Particle Swarm Optimization Methods for Pattern Recognition and Image Processing

by

Mahamed G. H. Omran

Submitted in partial fulfillment of the requirements for the degree Philosophiae Doctor in the Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

Pretoria

November 2004

Particle Swarm Optimization Methods for Pattern Recognition and Image Processing

by
Mahamed G. H. Omran

Abstract

Pattern recognition has as its objective to classify objects into different categories and classes. It is a fundamental component of artificial intelligence and computer vision. This thesis investigates the application of an efficient optimization method, known as Particle Swarm Optimization (PSO), to the field of pattern recognition and image processing. First a clustering method that is based on PSO is proposed. The application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images is investigated. A new automatic image generation tool tailored specifically for the verification and comparison of various unsupervised image classification algorithms is then developed. A dynamic clustering algorithm which automatically determines the "optimum" number of clusters and simultaneously clusters the data set with minimal user interference is then developed. Finally, PSO-based approaches are proposed to tackle the color image quantization and spectral unmixing problems. In all the proposed approaches, the influence of PSO parameters on the performance of the proposed algorithms is evaluated.

Key terms: Clustering, Color Image Quantization, Dynamic Clustering, Image Processing, Image Segmentation, Optimization Methods, Particle Swarm Optimization, Pattern Recognition, Spectral Unmixing, Unsupervised Image Classification.

Thesis supervisor: Prof. A. P. Engelbrecht

Thesis co-supervisor: Dr. Ayed Salman
Department of Computer Engineering, Kuwait University, Kuwait

Department of Computer Science

Degree: Philosophiae Doctor

"Obstacles are those frightening things you see when you take your eyes off your goal."

Henry Ford

"You will recognize your own path when you come upon it, because you will suddenly have all the energy and imagination you will ever need."

Jerry Gillies

Acknowledgments

I address my sincere gratitude to God as whenever I faced any difficulty I used to pray to God to help me and He always was there protecting and saving me.

Then, I would like to express my warm thanks to Professor Andries Engelbrecht, who spared no effort in supporting me with all care and patience. I enjoyed working with him, making every moment I spent in the research work as enjoyable as can be imagined.

I would like also to thank my co-supervisor Dr. Ayed Salman from Kuwait University for his continuous guidance, encouragement and patience throughout the PhD journey. I will never forget the long hours we spent together discussing various ideas and methods.

Last but not least, I would love to thank my family for their support and care, especially my mother Aysha and my father Ghassib. May God bless and protect them both hoping that God will help me to repay them part of what they really deserve. I also thank my two sisters Ala'a and Esra'a for their help in preparing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As humans, it is easy (even for a child) to recognize letters, objects, numbers, voices of friends, etc. However, making a computer solve these types of problems is a very difficult task. Pattern recognition is the science with the objective to classify objects into different categories and classes. It is a fundamental component of artificial intelligence and computer vision. Pattern recognition methods are used in various areas such as science, engineering, business, medicine, etc. Interest in pattern recognition is fast growing in order to deal with the prohibitive amount of information we encounter in our daily life. Automation is desperately needed to handle this information explosion. This thesis investigates the application of an efficient optimization method, known as Particle Swarm Optimization, to the field of pattern recognition and image processing. PSOs solve optimization problems by simulating the social behavior of bird flocks.

1.1 Motivation

There are many difficult problems in the field of pattern recognition and image processing. These problems are the focus of much active research in order to find efficient approaches to address them. However, the outcome of the research is still unsatisfactory.

Local search approaches were generally used to solve difficult problems in the field of pattern recognition and image processing. However, the selected set of problems in this thesis are NP-hard and combinatorial. Hence, evolutionary algorithms are generally more suitable to solve these difficult problems because they are population-based stochastic approaches. Thus, evolutionary algorithms can avoid being trapped in a local optimum and can often find a global optimal solution. A PSO is a population-based stochastic optimization algorithm modeled after the simulation of the social behavior of bird flocks. PSO is easy to implement and has been successfully applied to solve a wide range of optimization problems [Hu 2004]. Thus, due to its simplicity and efficiency in navigating large search spaces for optimal solutions, PSOs are used in this research to develop efficient, robust and flexible algorithms to solve a selective set of difficult problems in the field of pattern recognition and image processing. Out of these problems, data clustering is elaborately tackled in this thesis specifically image data. The motivation for the focus on data clustering is the fact that data clustering is an important process in pattern recognition and machine learning. Actually, clustering is a primary goal of pattern recognition. Furthermore, it is a central process in Artificial Intelligence. In addition, clustering algorithms are used in many applications, such as image segmentation, vector and color image quantization, spectral unmixing, data mining, compression, etc. Therefore, finding an efficient clustering algorithm is very important for researchers in many different disciplines.

1.2 Objectives

The primary objectives of this thesis can be summarized as follows:

- To show that the PSO can be successfully used to solve difficult problems in pattern recognition and image processing.

- To develop an efficient clustering algorithm based on PSO.
- To develop a tool that can aid researchers in the unsupervised image classification field to test their algorithms, compare different clustering algorithms and generate benchmarks.
- To develop an efficient dynamic clustering algorithm that can find the "optimum" number of clusters in a data set with minimum user interference.
- To develop a PSO-based approach to tackle the color image quantization problem.
- To develop an efficient end-members selection method based on PSO for spectral unmixing of multispectral imagery data.

1.3 Methodology

Algorithms proposed in this thesis are first presented and discussed. Experimental results were then generally obtained using various synthetic images with well-known characteristics in order to show the accuracy and efficiency of the proposed algorithms.

In addition, natural images from different areas such as medical images and remotely sensed satellite images were also used to show the wide applicability of the proposed approaches.

The results of *state-of-the-art* algorithms when applied to the same test images were also reported to show the relative performance of the proposed approaches when compared to other well-known approaches.

For the task of unsupervised image classification, attempts were made to find the best values for the PSO parameters.

Due to the stochastic nature of the proposed algorithms, all the presented results are averages and standard deviations over several simulations. However, due to the computational expensive nature of the simulations, results were generally taken over 10 or 20 runs.

1.4 Contributions

The main contributions of this thesis are:

- The development of an efficient clustering algorithm based on the PSO that performs better than *state-of-the-art* clustering algorithms when applied to the problem of unsupervised image classification.
- The development of a simple tool for synthetic image generation and verification. This tool can be used as a preliminary test to compare different unsupervised image classification algorithms. In addition, it can be used to generate a set of benchmark images that can be used by the researchers in the field of unsupervised image classification.
- The development of an efficient dynamic clustering algorithm based on the PSO that is able to simultaneously cluster a data set and find the "optimum" number of clusters in the data set.
- The development of an efficient color image quantization algorithm based on the PSO which is capable of generating high quality quantized images.
- The development of an efficient end-members selection method for spectral unmixing of multispectral satellite imagery data which is based on the PSO. The efficiency of the algorithm is demonstrated by applying it to test imagery from various platforms.

1.5 Thesis Outline

Chapter 2 briefly reviews the subject of optimization. This is followed by a brief discussion of traditional and stochastic optimization methods. Evolutionary Algorithms (EAs) (with more emphasis on Genetic Algorithms (GAs)) are then presented. This is followed by an elaborated discussion of particle swarm optimization and its various modifications. PSO is a model from the swarm intelligence paradigm. Therefore in order to provide a complete coverage of swarm intelligence background, a brief overview of another swarm intelligence model, Ant Colony Systems, is given.

Chapter 3 reviews the problems addressed in this thesis in sufficient detail. First the clustering problem is defined and different clustering concepts and approaches are presented. This is followed by defining image segmentation in addition to presenting various image segmentation methods. A survey of color image quantization and its approaches is then presented. This is followed by a brief introduction to spectral unmixing.

Chapter 4 presents a clustering method that is based on PSO. The algorithm finds the centroids of a user specified number of clusters, where each cluster groups together similar patterns. The application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images is investigated. To illustrate its wide applicability, the proposed algorithm is then applied to synthetic, MRI and satellite images.

Chapter 5 presents a new automatic image generation tool tailored specifically for the verification and comparison of different unsupervised image classification algorithms. The usefulness of the tool is demonstrated in this chapter with reference to the well-known K-means clustering algorithm and the PSO-based clustering algorithm proposed in the chapter 4.

Chapter 6 presents a new dynamic clustering approach based on PSO. This approach is applied to unsupervised image classification. The proposed approach automatically determines the "optimum" number of clusters and simultaneously clusters the data set with minimal user interference. The proposed approach is then applied to synthetic, natural and multispectral images. A genetic algorithm and a random search version of dynamic clustering are presented and compared to the particle swarm version.

Chapter 7 presents PSO-based approaches to tackle the color image quantization and spectral unmixing problems. The proposed approaches are then applied on different image sets to show their applicability and they are compared with other *state-of-the-art* approaches.

Chapter 8 highlights the conclusions of this thesis and discusses directions for future research.

The appendices present a definition of frequently used terms and symbols and a list of publications derived from the work introduced in this thesis.

Chapter 2

Optimization and Optimization Methods

This chapter provides a brief overview of optimization. This is followed by a brief discussion of traditional and stochastic optimization methods. Evolutionary algorithms (with more emphasis on genetic algorithms) are then presented. This is followed by an elaborated discussion of particle swarm optimization and its various modifications. A brief overview of ant colony systems is then given.

2.1 Optimization

The objective of optimization is to seek values for a set of parameters that maximize or minimize objective functions subject to certain constraints [Rardin 1998; Van den Bergh 2002]. A choice of values for the set of parameters that satisfy all constraints is called a *feasible solution*. Feasible solutions with objective function value(s) as good as the values of any other feasible solutions are called *optimal solutions* [Rardin 1998]. An example of an optimization problem is the arrangement of the transistors in a computer chip in such a way that the resulting layout occupies the smallest area and that as few as possible components are used. Optimization techniques are used on a daily base for industrial planning, resource allocation, scheduling, decision making, etc. Furthermore, optimization techniques are widely used in many fields such as business, industry, engineering and computer science. Research in the optimization field is very active and new optimization methods are being developed regularly [Chinneck 2000].

Optimization encompasses both maximization and minimization problems. Any maximization problem can be converted into a minimization problem by taking the negative of the objective function, and *vice versa*. Hence, the terms optimization, maximization and minimization are used interchangeably in this thesis. In general, the problems tackled in this thesis are minimization problems. Therefore, the remainder of the discussion focuses on minimization problems.

The minimization problem can be defined as follows [Pardalos *et al*. 2002]

Given $f : \boldsymbol{S} \rightarrow \Re$ where $\boldsymbol{S} \subseteq \Re^{N_d}$ and N_d is the dimension of the search space $\boldsymbol{S}$

find $\boldsymbol{x}^* \in \boldsymbol{S}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \boldsymbol{S}$ (2.1)

The variable $\boldsymbol{x}^*$ is called the *global minimizer* (or simply the *minimizer*) of f and $f(\boldsymbol{x}^*)$ is called the *global minimum* (or simply the *minimum*) value of f. This can be illustrated as given in Figure 2.1 where $\boldsymbol{x}^*$ is a global minimizer of f. The process of finding the global optimal solution is known as *global optimization* [Gray *et al*. 1997]. A true global optimization algorithm will find $\boldsymbol{x}^*$ regardless of the selected starting point $\boldsymbol{x}_0 \in \boldsymbol{S}$ [Van den Bergh 2002]. Global optimization problems are generally very difficult and are categorized under the class of nonlinear programming (NLP) [Gray *et al*. 1997].

Examples of global optimization problems are [Gray *et al*. 1997]:

- Combinatorial problems: where a linear or nonlinear function is defined over a finite but very large set of solutions, for example, network problems and scheduling [Pardalos *et al*. 2002]. The problems addressed in this thesis belong to this category.
- General unconstrained problems: where a nonlinear function is defined over an unconstrained set of real values.
- General constrained problems: where a nonlinear function is defined over a constrained set of real values.

Evolutionary algorithms (discussed in Sections 2.4-2.5) have been successfully applied to the above problems to find approximate solutions [Gray *et al*. 1997]. More details about global optimization can be found in Pardalos *et al*. [2002], Floudas and Pardalos [1992] and Horst *et al*. [2000].

In Figure 2.1, $\boldsymbol{x}_{\boldsymbol{B}}^*$ is called the *local minimizer* of region $\boldsymbol{B}$ because $f(\boldsymbol{x}_{\boldsymbol{B}}^*)$ is the smallest value within a local neighborhood, $\boldsymbol{B}$. Mathematically speaking, the variable $\boldsymbol{x}_{\boldsymbol{B}}^*$ is a local minimizer of the region $\boldsymbol{B}$ if

$$f(\boldsymbol{x}_{\boldsymbol{B}}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \boldsymbol{B} \tag{2.2}$$

where $\boldsymbol{B} \subset \boldsymbol{S}$. Every global minimizer is a local minimizer, but a local minimizer is not necessarily a global minimizer.

Generally, a local optimization method is guaranteed to find the local minimizer $\boldsymbol{x}_{\boldsymbol{B}}^*$ of the region $\boldsymbol{B}$ if a starting point $\boldsymbol{x}_0$ is used with $\boldsymbol{x}_0 \in \boldsymbol{B}$. An optimization algorithm that converges to a local minimizer, regardless of the selected starting point $\boldsymbol{x}_0 \in \boldsymbol{S}$, is called a *globally convergent* algorithm [Van den Bergh

2002]. There are many local optimization algorithms in the literature. For more detail the reader is referred to Aarts and Lenstra [2003] and Korte and Vygen [2002].

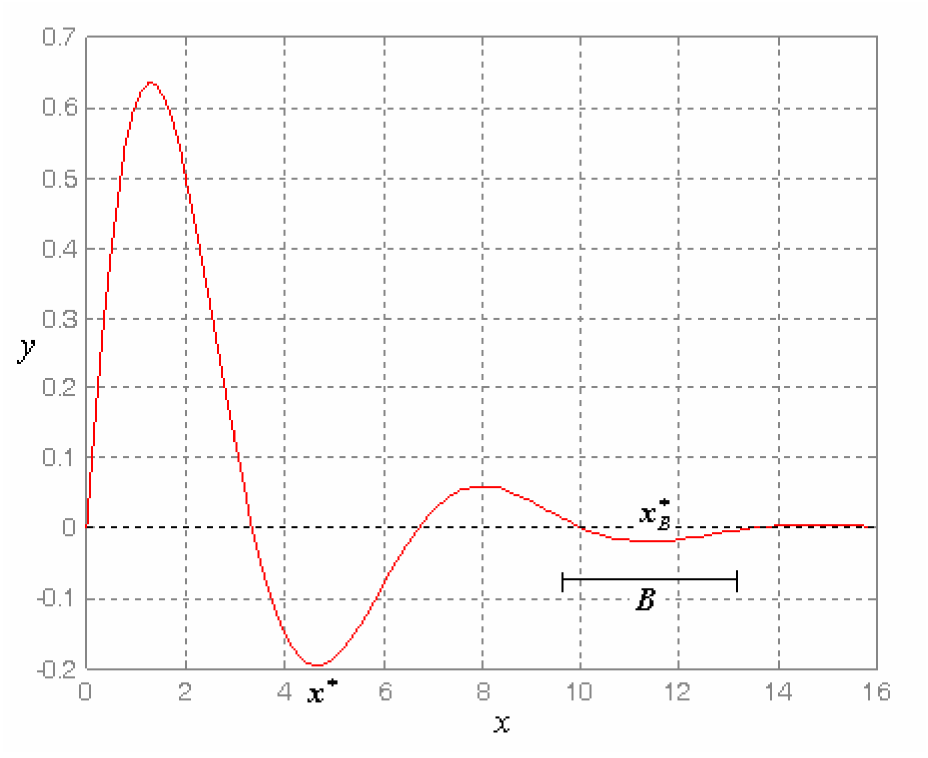


Figure 2.1: Example of a global minimizer x^* as well as a local minimizer x_B^*

2.2 Traditional Optimization Algorithms

Traditional optimization algorithms use exact methods to find the best solution. The idea is that if a problem can be solved, then the algorithm should find the global best solution. One exact method is the *brute force* (or *exhaustive*) search method where the algorithm tries every solution in the search space so that the global optimal solution is guaranteed to be found. Obviously, as the search space increases the cost of brute force algorithms increases. Therefore, brute force algorithms are not appropriate for the class of problems known as NP-hard problems. The time to exhaustively search an

NP-hard problem increases exponentially with problem size. Other exact methods include linear programming, divide and conquer and dynamic programming. More details about exact methods can be found in Michalewicz and Fogel [2000].

2.3 Stochastic Algorithms

Stochastic search algorithms are used to find near-optimal solutions for NP-hard problems in polynomial time. This is achieved by assuming that good solutions are close to each other in the search space. This assumption is valid for most real world problems [Løvberg 2002; Spall 2003]. Since the objective of a stochastic algorithm is to find a near-optimal solution, stochastic algorithms may fail to find a global optimal solution. While an exact algorithm generates a solution only after the run is completed, a stochastic algorithm can be stopped any time during the run and generate the best solution found so far [Løvberg 2002].

Stochastic search algorithms have several advantages compared to other algorithms [Venter and Sobieszczanski-Sobieski 2002]:

- Stochastic search algorithms are generally easy to implement.
- They can be used efficiently in a multiprocessor environment.
- They do not require the problem definition function to be continuous.
- They generally can find optimal or near-optimal solutions.
- They are suitable for discrete and combinatorial problems.

Three major stochastic algorithms are Hill-Climbing [Michalewicz and Fogel 2000], Simulated Annealing [Van Laarhoven and Aarts 1987] and Tabu search [Glover 1989;

Glover 1990]. In Hill-Climbing, a potential solution is randomly chosen. The algorithm then searches the neighborhood of the current solution for a better solution. If a better solution is found, then it is set as the new potential solution. This process is repeated until no more improvement can be made. Simulated annealing is similar to Hill-Climbing in the sense that a potential solution is randomly chosen. A small value is then added to the current solution to generate a new solution. If the new solution is better than the original one then the solution moves to the new location. Otherwise, the solution will move to the new location with a probability that decreases as the run progresses [Salman 1999]. Tabu search is a heuristic search algorithm where a tabu list memory of previously visited solutions is maintained in order to improve the performance of the search process. The tabu list is used to "guide the movement from one solution to the next one to avoid cycling" [Gabarro 2000], thus, avoid being trapped in a local optimum. Tabu search starts with a randomly chosen current solution. A set of test solutions are generated via moves from the current solution. The best test solution is set as the current solution if it is not in the tabu list, or if it is in the tabu list, but satisfies an aspiration criterion. A test solution satisfies an aspiration criterion if it is in the tabu list and it is the best solution found so far [Chu and Roddick 2003]. This process is repeated until a stopping criterion is satisfied.

2.4 Evolutionary Algorithms

Evolutionary algorithms (EAs) are general-purpose stochastic search methods simulating natural selection and evolution in the biological world. EAs differ from other optimization methods, such as Hill-Climbing and Simulated Annealing, in the fact that EAs maintain a population of potential (or candidate) solutions to a problem, and not just one solution [Engelbrecht 2002; Salman 1999].

Generally, all EAs work as follows: a population of individuals is initialized where each individual represents a potential solution to the problem at hand. The quality of each solution is evaluated using a *fitness function*. A selection process is applied during each iteration of an EA in order to form a new population. The selection process is biased toward the fitter individuals to ensure that they will be part of the new population. Individuals are altered using unary transformation (mutation) and higher order transformation (crossover). This procedure is repeated until convergence is reached. The best solution found is expected to be a *near-optimum* solution [Michalewicz 1996]. A general pseudo-code for an EA is shown in Figure 2.2 [Gray *et al*. 1997].

```
Initialize the population
Evaluate the fitness of each individual in the population
Repeat
        Apply selection on the population to form a new population
        Alter the individuals in the population using evolutionary operators
        Evaluate the fitness of each individual in the population
Until some convergence criteria are satisfied
```

Figure 2.2: General pseudo-code for EAs

The unary and higher order transformations are called *evolutionary operators*. The two most frequently evolutionary operators are:

- *Mutation*, which modifies an individual by a small random change to generate a new individual [Michalewicz 1996]. This change can be done by inverting the value of a binary digit in the case of binary representations, or by adding (or subtracting) a small number to (or from) selected values in the case of floating point representations. The main objective of mutation is to add some diversity by introducing more genetic material into the population in order to avoid being trapped in a local optimum. Generally, mutation is applied using a low probability. However, some problems (e.g. problems using floating point representations) require using mutation with high probability [Salman 1999]. A preferred strategy is to start with high probability of mutation and dreasing it over time.

- *Recombination (or Crossover)*, where parts from two (or more) individuals are combined together to generate new individuals [Michalewicz 1996]. The main objective of crossover is to explore new areas in the search space [Salman 1999].

There are four major evolutionary techniques:

- *Genetic Programming* (GP) [Koza 1992] which is used to search for the fittest program to solve a specific problem. Individuals are represented as trees and the focus is on genotypic evaluation.
- *Evolutionary Programming* (EP) [Fogel 1994] which is generally used to optimize real-valued continuous functions. EP uses selection and mutation operators; it does not use the recombination operator. The focus is on phenotypic evaluation and not on genotypic evaluation.
- *Evolutionary Strategies* (ES) [Bäck *et al.* 1991] which is used to optimize real-valued continuous functions. ES uses selection, crossover and mutation operators. ES optimizes both the population and the optimization process, by evolving strategy parameters. Hence, ES is evolution of evolution.

- *Genetic Algorithms* (GA) [Goldberg 1989] which is generally used to optimize general combinatorial problems [Gray *et al*. 1997]. The GA is a commonly used algorithm and has been used for comparison purposes in this thesis. The focus in GA is on genetic evolution using both mutation and crossover, although the original GAs developed by Holland [1962] used only crossover. Since later chapters make use of GAs, a detailed explanation of GAs is given in Section 2.5.

Due to its population-based nature, EAs can avoid being trapped in a local optimum and consequently can often find global optimal solutions. Thus, EAs can be viewed as global optimization algorithms. However, it should be noted that EAs may fail to converge to a global optimum [Gray *et al*. 1997].

EAs have successfully been applied to a wide variety of optimization problems, for example: image processing, pattern recognition, scheduling, engineering design, etc. [Gray *et al* 1997; Goldberg 1989].

2.5 Genetic Algorithms

Genetic Algorithms (GAs) are evolutionary algorithms that use selection, crossover and mutation operators. GAs were first proposed by Holland [1962; 1975] and were inspired by Darwinian evolution and Mendelian genetics [Salman 1999]. GAs follow the same algorithm presented in Figure 2.2. GAs are one of the most popular evolutionary algorithms and have been widely used to solve difficult optimization problems. GAs have been successfully applied in many areas such as pattern recognition, image processing, machine learning, etc. [Goldberg 1989]. In many cases GAs perform better than EP and ESs. However, EP and ESs usually converge better than GAs for real valued function optimization [Weiss 2003]. Individuals in GAs are called *chromosomes*. Each chromosome consists of a string of cells called *genes*. The value of each gene is called *allele*. The major parameters of GAs are discussed in Sections 2.5.1-2.5.5. In Section 2.5.6, a brief discussion about a problem that may be encountered in GAs is discussed.

2.5.1 Solution Representation

Binary representation is often used in GAs where each gene has a value of either 0 or 1. Other presentations have been proposed, for example, floating point representations [Janikow and Michalewicz 1991], integer representations [Bramlette 1991], gray-coded representations [Whitley and Rana 1998] and matrix representation [Michalewicz 1996]. More detail about representation schemes can be found in Goldberg [1989]. Generally, non-binary representations require different evolutionary operators for each representation while uniform operators can be used with binary representation for any problem [Van den Bergh 2002]. However, according to Michalewicz [1991], floating point representations are faster, more consistent and have higher precision than binary representations.

2.5.2 Fitness Function

A key element in GAs is the selection of a fitness function that accurately quantifies the quality of candidate solutions. A good fitness function enables the chromosomes to effectively solve a specific problem. Both the fitness function and solution representation are problem dependent parameters. A poor selection of these two parameters will drastically affect the performance of GAs. One problem related to fitness functions that may occur when GAs are used to optimize combinatorial problems is the existence of points in the search space that do not map to feasible solutions. One solution to this problem is the addition of a *penalty function* term to the original fitness function so that chromosomes representing infeasible solutions will have a low fitness score, and as such, will disappear from the population [Fletcher 2000].

2.5.3 Selection

Another key element of GAs is the selection operator which is used to select chromosomes (called *parents*) for mating in order to generate new chromosomes (called *offspring*). In addition, the selection operator can be used to select elitist individuals. The selection process is usually biased toward fitter chromosomes. Selection methods are used as mechanisms to focus the search on apparently more profitable regions in the search space [Angeline, Using Selection 1998]. Examples of well-known selection approaches are:

- *Roulette wheel selection:* Parent chromosomes are probabilistically selected based on their fitness. The fitter the chromosome, the higher the probability that it may be chosen for mating. Consider a roulette wheel where each chromosome in the population occupies a slot with slot size proportional to the chromosome's fitness [Gray *et al.* 1997]. When the wheel is randomly spun, the chromosome corresponding to the slot where the wheel stopped is selected as the first parent. This process is repeated to find the second parent. Clearly, since fitter chromosomes have larger slots, they have better chance to be chosen in the selection process [Goldberg 1989].

- *Rank selection:* Roulette wheel selection suffers from the problem that highly fit individuals may dominate in the selection process. When one or a few chromosomes have a very high fitness compared to the fitness of other chromosomes, the lower fit chromosomes will have a very slim chance to be selected for mating. This will increase selection pressure, which will cause diversity to decrease rapidly resulting in premature convergence. To reduce this problem, rank selection sorts the chromosomes according to their fitness and base selection on the rank order of the chromosomes, and not on the absolute fitness values. The worst (i.e. least fit) chromosome has rank of 1, the second worst chromosome has rank of 2, and so on. Rank selection still prefers the best chromosomes; however, there is no domination as in the case of roulette wheel selection. Hence, using this approach all chromosomes will have a good chance to be selected. However, this approach may have a slower convergence rate than the roulette wheel approach [Gray *et al*. 1997].
- *Tournament selection:* In this more commonly used approach [Goldberg 1989], a set of chromosomes are randomly chosen. The fittest chromosome from the set is then placed in a mating pool. This process is repeated until the mating pool contains a sufficient number of chromosomes to start the mating process.
- *Elitism:* In this approach, the fittest chromosome, or a user-specified number of best chromosomes, is copied into the new population. The remaining chromosomes are then chosen using any selection operator. Since the best solution is never lost, the performance of GA can significantly be improved [Gray *et al*. 1997].

2.5.4 Crossover

Crossover is "the main explorative operator in GAs" [Salman 1999]. Crossover occurs with a user-specified probability, called the *crossover probability* p_c. p_c is problem dependent with typical values in the range between 0.4 and 0.8 [Weiss 2003]. The four main crossover operators are:

- *Single point crossover:* In this approach, a position is randomly selected at which the parents are divided into two parts. The parts of the two parents are then swapped to generate two new offspring.

  Example 2.1

  Parent A: 11001**010**

  Parent B: **01110**011

  Offspring A: 11001011

  Offspring B: **01110010**

- *Two point crossover:* In this approach, two positions are randomly selected. The middle parts of the two parents are then swapped to generate two new offspring.

  Example 2.2

  Parent A: 11**001**010

  Parent B: **01**110**011**

  Offspring A: 11110010

  Offspring B: **01001011**

- *Uniform crossover:* In this approach, alleles are copied from either the first parent or the second parent with some probability, usually set to 0.5.

  Example 2.3

  Parent A: 1**10**010**10**

  Parent B: **0**11**100**11

  Offspring A: 11101011

  Offspring B: **01010010**

- *Arithmetic crossover:* In this approach, which is used for floating point representations, offspring is calculated as the arithmetic mean of the parents [Michalewicz 1996; Krink and Løvbjerg 2002], i.e.

$$\boldsymbol{x}_{\text{offspring A}} = r\,\boldsymbol{x}_{\text{parent A}} + (1-r)\boldsymbol{x}_{\text{parent B}} \tag{2.3}$$

$$\boldsymbol{x}_{\text{offspring B}} = r\,\boldsymbol{x}_{\text{parent B}} + (1-r)\boldsymbol{x}_{\text{parent A}} \tag{2.4}$$

  where $r \sim U(0,1)$.

2.5.5 Mutation

In GAs, mutation is considered to be a background operator, mainly used to explore new areas in the search space and to add diversity (contrary to selection and crossover which reduces diversity) to the population of chromosomes in order to prevent being trapped in a local optimum. Mutation is applied to the offspring chromosomes after crossover is performed. In a binary coded GA, mutation is done by inverting the value of each gene in the chromosome according to a user-specified probability, which is called the *mutation probability*, p_m. This probability is problem dependent. Mutation occurs infrequently both in nature and in GAs [Løvberg 2002], hence, a typical value for p_m is 0.01 [Weiss 2003]. However, a better value for p_m is the inverse of the number of genes in a chromosome (i.e. chromosome size) [Goldberg 1989].

One mutation scheme used with floating point representations is the non-uniform mutation [Michalewicz 1996]. The j^{th} element of chromosome $\boldsymbol{x}$ is mutated as follows:

$x_j = x_j + \Delta x_j$, where

$$\Delta x_j = \begin{cases} +(Z_{\max} - x_j)(1 - r^{(1-t/t_{\max})^b}) & \text{if a random bit is 0} \\ -(x_j - Z_{\min})(1 - r^{(1-t/t_{\max})^b}) & \text{if a random bit is 1} \end{cases} \tag{2.5}$$

where $Z_{\min}$ and $Z_{\max}$ are the lower and upper bound of the search space, $r \sim U(0,1)$, t is the current iteration, $t_{\max}$ is the total number of iterations and b is a user-specified parameter determining the degree of iteration number dependency (in this thesis, b was set to 5 as suggested by Michalewicz [1996]). Thus, the amount of mutation decreases as the run progresses.

Kennedy and Spears [1998] observed that a GA using either mutation or crossover performed better than a GA using both crossover and mutation operators when applied to a set of random problems (especially for problems with a large multimodality).

2.5.6 The Premature Convergence Problem

Genetic algorithms suffer from the *premature suboptimal convergence* (simply *premature convergence* or *stagnation*) which occurs when some poor individuals attract the population - due to a local optimum or bad initialization - preventing further exploration of the search space [Dorigo *et al*. 1999]. One of the causes of this problem is that a very fit chromosome is generally sure to be selected for mating, and since offspring resemble their parents, chromosomes become too similar (i.e. population loses diversity). Hence, the population will often converge before reaching the global optimal solution, resulting in premature convergence. Premature convergence can be prevented by:

- Using subpopulations: The population of chromosomes is divided into separate subpopulations. Each subpopulation is evolved independent of the other subpopulations for a user-specified number of generations. Then, a number of chromosomes are exchanged between the subpopulations. This process helps in increasing diversity and thus preventing premature convergence.
- Re-initializing some chromosomes: A few chromosomes are re-initialized from time to time in order to add diversity to the population.
- Increase the mutation probability: As already discussed, mutation aids in exploring new areas in the search space and increases diversity. Therefore, increasing p_m will help in preventing premature convergence.

In general, any mechanism that can increase diversity will help in preventing premature convergence.

2.6 Particle Swarm Optimization

A particle swarm optimizer (PSO) is a population-based stochastic optimization algorithm modeled after the simulation of the social behavior of bird flocks [Kennedy and Eberhart 1995; Kennedy and Eberhart 2001]. PSO is similar to EAs in the sense that both approaches are population-based and each individual has a fitness function. Furthermore, the adjustments of the individuals in PSO are relatively similar to the arithmetic crossover operator used in EAs [Coello Coello and Lechuga 2002]. However, PSO is influenced by the simulation of social behavior rather than the survival of the fittest [Shi and Eberhart 2001]. Another major difference is that, in PSO, each individual benefits from its history whereas no such mechanism exists in EAs [Coello Coello and Lechuga 2002]. PSO is easy to implement and has been successfully applied to solve a wide range of optimization problems such as continuous nonlinear and discrete optimization problems [Kennedy and Eberhart 1995; Kennedy and Eberhart 2001; Eberhart and Shi, Comparison 1998].

2.6.1 The PSO Algorithm

In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Each particle represents a candidate solution to the optimization problem. The position of a particle is influenced by the best position visited by itself (i.e. its own experience) and the position of the best particle in its neighborhood (i.e. the experience of neighboring particles). When the neighborhood of a particle is the entire swarm, the best position in the neighborhood is referred to as the global best particle, and the resulting algorithm is referred to as a *gbest* PSO. When smaller neighborhoods are used, the algorithm is generally referred to as a *lbest* PSO [Shi and Eberhart, Parameter 1998]. The performance of each particle (i.e. how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem.

Each particle in the swarm is represented by the following characteristics:

$\boldsymbol{x}_i$: The *current position* of the particle;

$\boldsymbol{v}_i$: The *current velocity* of the particle;

$\boldsymbol{y}_i$: The *personal best position* of the particle.

$\hat{\boldsymbol{y}}_i$: The *neighborhood best position* of the particle.

The personal best position of particle *i* is the best position (i.e. the one resulting in the best fitness value) visited by particle *i* so far. Let *f* denote the objective function. Then the personal best of a particle at time step *t* is updated as

$$\boldsymbol{y}_i(t+1) = \begin{cases} \boldsymbol{y}_i(t) & \text{if } f(\boldsymbol{x}_i(t+1)) \geq f(\boldsymbol{y}_i(t)) \\ \boldsymbol{x}_i(t+1) & \text{if } f(\boldsymbol{x}_i(t+1)) < f(\boldsymbol{y}_i(t)) \end{cases} \tag{2.6}$$

For the *gbest* model, the best particle is determined from the entire swarm by selecting the best personal best position. If the position of the global best particle is denoted by the vector $\hat{\boldsymbol{y}}$, then

$$\hat{\boldsymbol{y}}(t) \in \{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_s\} = \min\{f(\boldsymbol{y}_0(t)), f(\boldsymbol{y}_1(t)), \ldots, f(\boldsymbol{y}_s(t))\} \tag{2.7}$$

where s denotes the size of the swarm.

The velocity update step is specified for each dimension $j \in 1,\ldots,N_d$, hence, $v_{i,j}$ represents the jth element of the velocity vector of the ith particle. Thus the velocity of particle i is updated using the following equation:

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t)) \tag{2.8}$$

where w is the inertia weight, c_1 and c_2 are the acceleration constants, and $r_{1,j}(t)$, $r_{2,j}(t) \sim U(0,1)$. Equation (2.8) consists of three components, namely

- The *inertia weight* term, w, which was first introduced by Shi and Eberhart [A modified 1998]. This term serves as a memory of previous velocities. The inertia weight controls the impact of the previous velocity: a large inertia weight favors exploration, while a small inertia weight favors exploitation [Shi and Eberhart, Parameter 1998].
- The *cognitive component*, $\boldsymbol{y}_i(t) - \boldsymbol{x}_i$, which represents the particle's own experience as to where the best solution is.
- The *social component*, $\hat{\boldsymbol{y}}(t) - \boldsymbol{x}_i(t)$, which represents the belief of the entire swarm as to where the best solution is.

According to Van den Bergh [2002], the relation between the inertia weight and acceleration constants should satisfy the following equation in order to have guaranteed convergence:

$$\frac{c_1 + c_2}{2} - 1 < w \tag{2.9}$$

Otherwise, the PSO particles may exhibit divergent of cyclic behavior. For a thorough study of the relationship between the inertia weight and acceleration constants, the reader is advised to refer to Ozcan and Mohan [1998], Clerc and Kennedy [2001], Van den Bergh [2002], Zheng *et al*. [2003], Yasuda *et al*. [2003] and Trelea [2003].

Velocity updates can also be clamped with a user defined maximum velocity, $V_{\max}$, which would prevent them from exploding, thereby causing premature convergence [Eberhart *et al*. 1996].

The position of particle i, $\boldsymbol{x}_i$, is then updated using the following equation:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{2.10}$$

The PSO updates the particles in the swarm using equations (2.8) and (2.10). This process is repeated until a specified number of iterations is exceeded, or velocity updates are close to zero. The quality of particles is measured using a fitness function which reflects the optimality of a particular solution. Figure 2.3 summarizes the basic PSO algorithm.

2.6.2 The *lbest* Model

For the *lbest* model, a swarm is divided into overlapping neighborhoods of particles. For each neighborhood N_i, the best particle is determined, with position $\hat{\boldsymbol{y}}_i$. This particle is referred to as the *neighborhood best* particle. Let the indices of the particles wrap around at s and the neighborhood size is l. Then the update equations are:

```
For each particle $i \in 1,\ldots,s$ do
  Randomly initialize $\boldsymbol{x}_i$
  Randomly initialize $\boldsymbol{v}_i$ (or just set $\boldsymbol{v}_i$ to zero)
  Set $\boldsymbol{y}_i = \boldsymbol{x}_i$
endfor
Repeat
  For each particle $i \in 1,\ldots,s$ do
    Evaluate the fitness of particle $i$, $f(\boldsymbol{x}_i)$
    Update $\boldsymbol{y}_i$ using equation (2.6)
    Update $\hat{\boldsymbol{y}}$ using equation (2.7)
    For each dimension $j \in 1,\ldots,N_d$ do
      Apply velocity update using equation (2.8)
    endloop
    Apply position update using equation (2.10)
  endloop
Until some convergence criteria is satisfied
```

Figure 2.3: General pseudo-code for PSO

$$N_i = \{\boldsymbol{y}_{i-l}(t), \boldsymbol{y}_{i-l+1}(t), \ldots, \boldsymbol{y}_{i-1}(t), \boldsymbol{y}_i(t), \boldsymbol{y}_{i+1}(t), \ldots, \boldsymbol{y}_{i+l-1}(t), \boldsymbol{y}_{i+l}(t)\} \tag{2.11}$$

$$\hat{\boldsymbol{y}}_i(t+1) \in \{N_i \mid f(\hat{\boldsymbol{y}}_i(t+1)) = \min\{f(\boldsymbol{y}_i(t))\}, \forall \boldsymbol{y}_i \in N_i\} \tag{2.12}$$

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_{i,j}(t) - x_{i,j}(t)) \tag{2.13}$$

The position update equation is the same as given in equation (2.10). Neighbors represent the social factor in PSO. Neighborhoods are usually determined using particle indices, however, topological neighborhoods can also be used [Suganthan 1999]. It is clear that *gbest* is a special case of *lbest* with $l = s$; that is, the neighborhood is the entire swarm. While the *lbest* approach results in a larger diversity, it is still slower than the *gbest* approach.

2.6.3 PSO Neighborhood topologies

Different neighborhood topologies have been investigated [Kennedy 1999; Kennedy and Mendes 2002]. Two common neighborhood topologies are the *star* (or *wheel*) and *ring* (or *circle*) topologies. For the star topology one particle is selected as a *hub*, which is connected to all other particles in the swarm. However, all the other particles are only connected to the hub. For the ring topology, particles are arranged in a ring. Each particle has some number of particles to its right and left as its neighborhood. Recently, Kennedy and Mendes [2002] proposed a new PSO model using a *Von Neumann* topology. For the Von Neumann topology, particles are connected using a grid network (2-dimensional lattice) where each particle is connected to its four neighbor particles (above, below, right and left particles). Figure 2.4 illustrates the different neighborhood topologies.

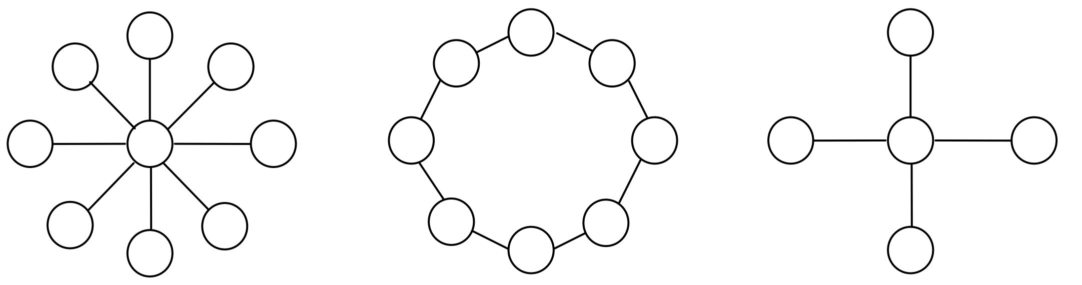

(a) Star topology (b) Ring Topology (c) Von Neumann Topology

Figure 2.4. A diagrammatic representation of neighborhood topologies

The choice of neighborhood topology has a profound effect on the propagation of the best solution found by the swarm. Using the *gbest* model the propagation is very fast (i.e. all the particles in the swarm will be affected by the best solution found in iteration t, immediately in iteration $t+1$). This fast propagation may result in the premature convergence problem discussed in Section 2.5.6. However, using the ring and Von Neumann topologies will slow down the convergence rate because the best solution found has to propagate through several neighborhoods before affecting all particles in the swarm. This slow propagation will enable the particles to explore more areas in the search space and thus decreases the chance of premature convergence.

2.6.4 The Binary PSO

Kennedy and Eberhart [1997] have adapted the PSO to search in binary spaces. For the binary PSO, the component values of $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ are restricted to the set {0, 1}. The velocity, $\boldsymbol{v}_i$, is interpreted as a probability to change a bit from 0 to 1, or from 1 to 0 when updating the position of particles. Therefore, the velocity vector remains continuous-valued. Since each $v_{i,j} \in \Re$, a mapping needs to be defined from $v_{i,j}$ to a probability in the range [0, 1]. This is done by using a sigmoid function to squash velocities into a [0, 1] range. The sigmoid function is defined as

$$sig(v) = \frac{1}{1+e^{-v}} \tag{2.14}$$

The equation for updating positions (equation (2.10)) is then replaced by the probabilistic update equation [Kennedy and Eberhart 1997]:

$$x_{i,j}(t+1) = \begin{cases} 0 & \text{if } r_{3,j}(t) \geq sig(v_{i,j}(t+1)) \\ 1 & \text{if } r_{3,j}(t) < sig(v_{i,j}(t+1)) \end{cases} \tag{2.15}$$

where $r_{3,j}(t) \sim U(0,1)$.

It can be observed from equation (2.15) that if $sig(v_{i,j}) = 0$ then $x_{i,j} = 0$. This situation occurs when $v_{i,j} < -10$. Furthermore, $sig(v_{i,j})$ will saturate when $v_{i,j} > 10$ [Van den Bergh 2002]. To avoid this problem, it is suggested to set $v_{i,j} \in [-4,4]$ and to use velocity clamping with $V_{\max} = 4$ [Kennedy and Eberhart 2001].

PSO has also been extended to deal with arbitrary discrete representation [Yoshida *et al.* 1999; Fukuyama and Yoshida 2001; Venter and Sobieszczanski-Sobieski 2002; Al-kazemi and Mohan 2000; Mohan and Al-Kazemi 2001]. These extensions are generally achieved by rounding $x_{i,j}$ to its closest discrete value after applying position update equation (2.10) [Venter and Sobieszczanski-Sobieski 2002].

2.6.5 PSO vs. GA

A PSO is an inherently continuous algorithm where as a GA is an inherently discrete algorithm [Venter and Sobieszczanski-Sobieski 2002]. Experiments conducted by Veeramachaneni *et al*. [2003] showed that a PSO performed better than GAs when applied on some continuous optimization problems. Furthermore, according to Robinson *et al*. [2002], a PSO performed better than GAs when applied to the design of a difficult engineering problem, namely, profiled corrugated horn antenna design [Diaz and Milligan 1996]. In addition, a binary PSO was compared with a GA by Eberhart and Shi [Comparison 1998] and Kennedy and Spears [1998]. The results showed that binary PSO is generally faster, more robust and performs better than binary GAs, especially when the dimension of a problem increases.

Hybrid approaches combining PSO and GA were proposed by Veeramachaneni *et al*. [2003] to optimize the profiled corrugated horn antenna. The hybridization works by taking the population of one algorithm when it has made no fitness improvement and using it as the starting population for the other algorithm. Two versions were proposed: GA-PSO and PSO-GA. In GA-PSO, the GA population is used to initialize the PSO population. For PSO-GA, the PSO population is used to initialize the GA population. According to Veeramachaneni *et al*. [2003], PSO-GA performed slightly better than PSO. Both PSO and PSO-GA outperformed both GA and GA-PSO.

Some of the first applications of PSO were to train Neural Networks (NNs), including NNs with product units. Results have shown that PSO is better than GA and other training algorithms [Eberhart and Shi, Evolving 1998; Van den Bergh and Engelbrecht 2000; Ismail and Engelbrecht 2000].

According to Shi and Eberhart [1998], the PSO performance is insensitive to the population size (however, the population size should not be too small). This observation was verified by Løvberg [2002] and Krink *et al*. [2002]. Consequently, PSO with smaller swarm sizes perform comparably to GAs with larger populations. Furthermore, Shi and Eberhart observed that PSO scales efficiently. This observation was verified by Løvberg [2002].

2.6.6 PSO and Constrained Optimization

Most engineering problems are constrained problems. However, the basic PSO is only defined for unconstrained problems. One way to allow the PSO to optimize constrained problems is by adding a penalty function to the original fitness function (as discussed in Section 2.5.2). In this thesis, a constant penalty function (empirically set to 10^6) is added to the original fitness function for each particle with violated constraints. More recently, a modification to the basic PSO was proposed by Venter and Sobieszczanski-Sobieski [2002] to penalize particles with violated constraints. The idea is to reset the velocity of each particle with violated constraints. Therefore, these particles will only be affected by $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}$. According to Venter and Sobieszczanski-Sobieski [2002], this modification has a significant positive effect on the performance of PSO. Other PSO approaches dealing with constrained problems can be found in El-Gallad *et al*. [2001], Hu and Eberhart [Solving 2002], Schoofs and Naudts [2002], Parsopoulos and Vrahatis [2002], Coath *et al*. [2003] and Gaing [2003].

2.6.7 Drawbacks of PSO

PSO and other stochastic search algorithms have two major drawbacks [Løvberg 2002]. The first drawback of PSO, and other stochastic search algorithms, is that the swarm may prematurely converge (as discussed in Section 2.5.6). According to Angeline [Evolutionary 1998], although PSO finds good solutions much faster than other evolutionary algorithms, it usually can not improve the quality of the solutions as the number of iterations is increased. PSO usually suffers from premature convergence when strongly multi-modal problems are being optimized. The rationale behind this problem is that, for the *gbest* PSO, particles converge to a single point, which is on the line between the global best and the personal best positions. This point is not guaranteed to be even a local optimum. Proofs can be found in Van den Bergh [2002]. Another reason for this problem is the fast rate of information flow between particles, resulting in the creation of similar particles (with a loss in diversity) which increases the possibility of being trapped in local optima [Riget and Vesterstrøm 2002]. Several modifications of the PSO have been proposed to address this problem. Two of these modifications have already been discussed, namely, the inertia weight and the *lbest* model. Other modifications are discussed in the next section.

The second drawback is that stochastic approaches have problem-dependent performance. This dependency usually results from the parameter settings of each algorithm. Thus, using different parameter settings for one stochastic search algorithm result in high performance variances. In general, no single parameter setting exists which can be applied to all problems. This problem is magnified in PSO where modifying a PSO parameter may result in a proportionally large effect [Løvberg 2002]. For example, increasing the value of the inertia weight, *w*, will increase the speed of the particles resulting in more exploration (global search) and less exploitation (local search). On the other hand, decreasing the value of *w* will decrease the speed of the particle resulting in more exploitation and less exploration. Thus finding the best value for *w* is not an easy task and it may differ from one problem to another. Therefore, it can be concluded that the PSO performance is problem-dependent.

One solution to the problem-dependent performance of PSO is to use self-adaptive parameters. In self-adaptation, the algorithm parameters are adjusted based on the feedback from the search process [Løvberg 2002]. Bäck [1992] has successfully applied self-adaptation on GAs. Self-adaptation has been successfully applied to PSO by Clerc [1999], Shi and Eberhart [2001], Hu and Eberhart [Adaptive 2002], Ratnaweera *et al.* [2003] and Tsou and MacNish [2003], Yasuda *et al.* [2003] and Zhang *et al.* [2003].

The problem-dependent performance problem can be addressed through *hybridization*. Hybridization refers to combining different approaches to benefit from the advantages of each approach [Løvberg 2002]. Hybridization has been successfully applied to PSO by Angeline [1998], Løvberg [2002], Krink and Løvbjerg [2002], Veeramachaneni *et al.* [2003], Reynolds *et al.* [2003], Higashi and Iba [2003] and Esquivel and Coello Coello [2003].

2.6.8 Improvements to PSO

The improvements presented in this section are mainly trying to address the problem of premature convergence associated with the original PSO. These improvements usually try to solve this problem by increasing the diversity of solutions in the swarm.

Constriction Factor

Clerc [1999] and Clerc and Kennedy [2001] proposed using a *constriction factor* to ensure convergence. The constriction factor can be used to choose values for w, c_1 and c_2 to ensure that the PSO converges. The modified velocity update equation is defined as follows:

$$v_{i,j}(t+1) = \chi(v_{i,j}(t) + c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) + c_2 r_{2,j}(t)(\hat{y}_j(t) - x_{i,j}(t))), \qquad (2.16)$$

where χ is the constriction factor defined as follows:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|},$$

and $\varphi = c_1 + c_2$, $\varphi > 4$.

Eberhart and Shi [2000] showed imperically that using both the constriction factor and velocity clamping generally improves both the performance and the convergence rate of the PSO.

Guaranteed Convergence PSO (GCPSO)

The original versions of PSO as given in Section 2.6.1, may prematurely converge when $\boldsymbol{x}_i = \boldsymbol{y}_i = \hat{\boldsymbol{y}}$, since the velocity update equation will depend only on the term $\boldsymbol{wv}_i(t)$ [Van den Bergh and Engelbrecht 2002; Van den Bergh 2002]. To overcome this problem, a new version of PSO with guaranteed local convergence was introduced by Van den Bergh [2002], namely GCPSO. In GCPSO, the global best particle with index τ is updated using a different velocity update equation, namely

$$v_{\tau,j}(t+1) = -x_{\tau,j}(t) + \hat{y}_j(t) + wv_{\tau,j}(t) + \rho(t)(1-2r_{2,j}(t)) \tag{2.17}$$

which results in a position update equation of

$$x_{\tau,j}(t+1) = \hat{y}_j(t) + w\boldsymbol{v}_{\tau,j}(t) + \rho(t)(1-2r_{2,j}(t)) \tag{2.18}$$

The term $-\boldsymbol{x}_\tau$ resets the particle's position to the global best position $\hat{\boldsymbol{y}}$; $w\boldsymbol{v}_\tau(t)$ signifies a search direction, and $\rho(t)(1-2r_{2,j}(t))$ adds a random search term to the equation. The term $\rho(t)$ defines the area in which a better solution is searched.

The value of $\rho(0)$ is initialized to 1.0, with $\rho(t+1)$ defined as

$$\rho(t+1) = \begin{cases} 2\rho(\mathrm{t}) & \text{if } \#successes > s_c \\ 0.5\rho(\mathrm{t}) & \text{if } \#failures > f_c \\ \rho(\mathrm{t}) & \text{otherwise} \end{cases} \tag{2.19}$$

A *failure* occurs when $f(\hat{\boldsymbol{y}}(t)) \geq f(\hat{\boldsymbol{y}}(t\text{-}1))$ (in the case of a minimization problem) and the variable *#failures* is subsequently incremented (i.e. no apparent progress has been made). A *success* then occurs when $f(\hat{\boldsymbol{y}}(t)) < f(\hat{\boldsymbol{y}}(t\text{-}1))$. Van den Bergh [2002] suggests learning the control threshold values f_c and s_c dynamically. That is,

$$s_c(t+1) = \begin{cases} s_c(t)+1 & \text{if } \#failures(t+1) > f_c \\ s_c(t) & \text{otherwise} \end{cases} \tag{2.20}$$

$$f_c(t+1) = \begin{cases} f_c(t)+1 & \text{if } \#success(t+1) > s_c \\ f_c(t) & \text{otherwise} \end{cases} \tag{2.21}$$

This arrangement ensures that it is harder to reach a success state when multiple failures have been encountered. Likewise, when the algorithm starts to exhibit overly confident convergent behavior, it is forced to randomly search a smaller region of the search space surrounding the global best position. For equation (2.19) to be well defined, the following rules should be implemented:

$\#successes(t+1) > \#successes(t) \Rightarrow \#failures(t+1) = 0$

$\#failures(t+1) > \#failures(t) \Rightarrow \#successes(t+1) = 0$

Van den Bergh suggests repeating the algorithm until ρ becomes sufficiently small, or until stopping criteria are met. Stopping the algorithm once ρ reaches a lower bound is not advised, as it does not necessarily indicate that all particles have converged – other particles may still be exploring different parts of the search space.

It is important to note that, for the GCPSO algorithm, all particles except for the global best particle still follow equations (2.8) and (2.10). Only the global best particle follows the new velocity and position update equations.

According to Van den Bergh [2002] and Peer *et al*. [2003], GCPSO generally performs better than PSO when applied to benchmark problems. This improvement in performance is especially noticeable when PSO and GCPSO are applied to unimodal functions, but the performance of both algorithms was generally comparable for multi-modal functions [Van den Bergh 2002]. Furthermore, due to its fast rate of convergence, GCPSO is slightly more likely to be trapped in local optima [Van den Bergh 2002]. However, it has gauaranteed local convergence whereas the original PSO does not.

Multi-start PSO (MPSO)

Van den Bergh [2002] proposed MPSO which is an extension to GCPSO in order to make it a global search algorithm. MPSO works as follows:

1. Randomly initialize all the particles in the swarm.
2. Apply the GCPSO until convergence to a local optimum. Save the position of this local optimum.
3. Repeat Steps 1 and 2 until some stopping criteria are satisfied.

In Step 2, the GCPSO can be replaced by the original PSO. Several versions of MPSO were proposed by Van den Bergh [2002] based on the way used to determine the convergence of GCPSO. One good approach is to measure the rate of change in the objective function as follows:

$$f_{\text{ratio}} = \frac{f(\hat{\mathbf{y}}(t)) - f(\hat{\mathbf{y}}(t-1))}{f(\hat{\mathbf{y}}(t))}$$

If f_{ratio} is less than a user-specified threshold then a counter is incremented. The swarm is assumed to have converged if the counter reaches a certain threshold [Van den Bergh 2002]. According to Van den Bergh [2002], MPSO generally performed better than GCPSO in most of the tested cases. However, the performance of MPSO degrades significantly as the number of dimensions in the objective function increases [Van den Bergh 2002].

Attractive and Repulsive PSO (ARPSO)

ARPSO [Riget and Vesterstrøm 2002] alternates between two phases: attraction and repulsion based on a diversity measure. In the attraction phase, ARPSO uses PSO to allow fast information flow, as such particles attract each other and thus the diversity reduces. It was found that 95% of fitness improvements were achieved in this phase. This observation shows the importance of low diversity in fine tuning the solution. In the repulsion phase, particles are pushed away from the best solution found so far thereby increasing diversity. Based on the experiments conducted by Riget and Vesterstrøm [2002] ARPSO outperformed PSO and GA in most of the tested cases.

Selection

A hybrid approach combining PSO with a tournament selection method was proposed by Angeline [Using Selection 1998]. Each particle is ranked based on its performance against a randomly selected group of particles. For this purpose, a particle is awarded one point for each opponent in the tournament for which the particle has a better fitnss. The population is then sorted in decending order according to the points accumulated. The bottom half of the population is then replaced by the top half. This step reduces the diversity of the population. The results showed that the hybrid approach performed better than the PSO (without w and χ) for unimodal functions. However, the hybrid approach performed worse than the PSO for functions with many local optima. Therefore, it can be concluded that although the use of a selection method improves the exploitation capability of the PSO, it reduces its exploration capability [Ven den Bergh 2002]. Hence, using a selection method with PSO may result in premature convergence.

Breeding

Løvberg *et al.* [2001] proposed a modification to PSO by using an arithmetic crossover operator (discussed in Section 2.5.4), referred to as a *breeding* operator, in order to improve the convergence rate of PSO. Each particle in the swarm is assigned a user-defined breeding probability. Based on these probabilities, two parent particles are randomly selected to create offspring using the arithmetic crossover operator. Offspring replace the parent particles. The personal best position of each offspring particle is initialized to its current position (i.e. $\boldsymbol{y}_i = \boldsymbol{x}_i$), and its velocity is set as the sum of the two parent's velocities normalized to the original length of each parent velocity. The process is repeated until a new swarm of the same size has been generated. PSO with breeding generally performed better than the PSO when applied to multi-modal functions [Løvberg *et al*. 2001].

Mutation

Recently, Higashi and Iba [2003] proposed hybriding PSO with Gaussian mutation. Similarly, Esquivel and Coello Coello [2003] proposed hybridizing *lbest*- and *gbest*-PSO with a powerful diversity maintenance mechanism, namely, a non-uniform mutation operator discussed in section 2.5.5 to solve the premature convergence problem of PSO. According to Esquivel and Coello Coello [2003] the hybrid approach of *lbest* PSO and the non-uniform mutation operator outperformed PSO and GCPSO in all of the conducted experiments.

Dissipative PSO (DPSO)

DPSO was proposed by Xie *et al*. [2002] to add random mutation to PSO in order to prevent premature convergence. DPSO introduces negative entropy via the addition of randomness to the particles (after executing equation (2.8) and (2.10)) as follows:

If $(r_1(t) < c_v)$ **then** $v_{i,j}(t+1) = r_2(t)V_{\max}$

If $(r_3(t) < c_l)$ **then** $x_{i,j}(t+1) = R(t)$

where $r_1(t) \sim U(0,1)$, $r_2(t) \sim U(0,1)$ and $r_3(t) \sim U(0,1)$; c_v and c_v are chaotic factors in the range [0,1] and $R(t) \sim U(Z_{\min},Z_{\max})$ where $Z_{\min}$ and $Z_{\max}$ are the lower and upper bound of the search space. The results showed that DPSO performed better than PSO when applied to the benchmarks problems [Xie *et al.* 2002].

Differential Evolution PSO (DEPSO)

DEPSO [Zhang and Xie 2003] uses a differential evolution (DE) operator [Storn and Price 1997] to provide the mutations. A trait point $\ddot{\boldsymbol{y}}_i(t)$ is calculated as follows:

If $(r_1(t) < p_c$ **OR** $j = k_d)$ **then**

$$\ddot{y}_{i,j}(t) = \hat{y}_j(t) + \frac{(y_{1,j}(t) - y_{2,j}(t)) + (y_{3,j}(t) - y_{4,j}(t))}{2} \tag{2.23}$$

where $r_1(t) \sim U(0,1)$, $k_d \sim U(1,N_d)$, and $\boldsymbol{y}_1(t)$, $\boldsymbol{y}_2(t)$, $\boldsymbol{y}_3(t)$ and $\boldsymbol{y}_4(t)$ are randomly chosen from the set of personal best positions. Then, $\boldsymbol{y}_i(t) = \ddot{\boldsymbol{y}}_i(t)$, only if $f(\ddot{\boldsymbol{y}}_i(t)) < f(\boldsymbol{y}_i(t))$. The rationale behind mutating $\boldsymbol{y}_i(t)$ instead of $\boldsymbol{x}_i(t)$ is to avoid disorganization of the swarm.

DEPSO works by alternating between the original PSO and the DE operator such that equations (2.8) and (2.10) are used in the odd iterations and equation (2.23) is used is in the even iterations. According to Zhang and Xie [2003], DEPSO generally performed better than PSO, DE, GA, ES, DPSO and fuzzy-adaptive PSO when applied to the benchmark functions.

Craziness

To avoid premature convergence, Kennedy and Eberhart [1995] introduced the use of a craziness operator with PSO. However, they concluded that this operator may not be necessay. More recently, Venter and Sobieszczanski-Sobieski [2002] reintroduced the craziness operator to PSO. In each iteration, a few particles far from the center of the swarm are selected. The positions of these particles are then randomly changed while their velocities are initialized to the cognitive velocity component, i.e.

$$v_{i,j}(t+1) = c_1 r_{1,j}(t)(y_{i,j}(t) - x_{i,j}(t)) \qquad (2.24)$$

According to Venter and Sobieszczanski-Sobieski [2002], the proposed craziness operator does not seem to have a big influence on the performance of PSO.

The LifeCycle Model

A self-adaptive heuristic search algorithm, called LifeCycle, was proposed by Krink and Løvbjerg [2002]. LifeCycle is a hybrid approach combing PSO, GA and Hill-Climbing approaches. The motivation for LifeCycle is to gain the benefits of PSO, GA and Hill-Climbing in one algorithm. In LifeCycle, the individuals (representing potential solutions) start as PSO particles, then depending on their performance (in searching for solutions) can change into GA individuals, or Hill-Climbers. Then, they can return back to particles. This process is repeated until convergence. The LifeCycle was compared with PSO, GA and Hill-Climbing [Krink, and Løvbjerg 2002] and has generally shown good performance when applied to the benchmark problems. However, PSO performed better than (or comparable to) the LifeCycle in three out of five benchmark problems. Another hybrid approach proposed by Veeramachaneni *et* *al*. [2003] combining PSO and GA has already been discussed in Section 2.6.5. From the experimental results of Krink and Løvbjerg [2002] and Veeramachaneni *et al*. [2003], it can be observed that the original PSO performed well compared to their more complicated hybrid approaches.

Multi-Swarm (or subpopulation)

The idea of using several swarms instead of one swarm was applied to PSO by Løvberg *et al*. [2001] and Van den Bergh and Engelbrecht [2001]. The approach proposed by Løvberg *et al*. [2001] is an extension of PSO with the breeding operator discussed above. The idea is to divide the swarm into several swarms. Each swarm has its own global best particles. The only interaction between the swarms occurs when the breeding selects two particles to mate from different swarms. The results in Løvberg *et al*. [2001] showed that this approach did not improve the performance of PSO. The expected reasons are [Jensen and Kristensen 2002]:

- The authors split a swarm of 20 particles into six different swarms. Hence, each swarm contains a few particles. Swarms with few particles have little diversity and therefore little exploration power.
- No action has been taken to prevent swarms from being too similar to each other.

The above problems were addressed by Jensen and Kristensen [2002]. The modified approach works by using two swarms (each with a size of 20 particles) and keeping them away from each other either by randomly spreading the swarm (with the worst performance) over the search space or by adding a small mutation to the positions of the particles in this swarm. The approach using the mutation technique generally performed better than PSO when applied to the benchmark problems [Jensen and Kristensen 2002]. However, one drawback of this approach is the fact that the decision of whether two swarms are too close to each other is very problem dependent [Jensen and Kristensen 2002].

Self-Organized Criticality (SOC PSO)

In order to increase the population diversity to avoid premature convergence, Løvberg and Krink [2002] extended PSO with *Self Organized Criticality* (SOC). A measure, called *criticality*, of how close particles are to each other is used to relocate the particles and thus increase the diversity of the swarm. A particle with a high criticality disperses its criticality by increasing the criticality of a user-specified number of particles, *CL*, in its neighborhood by 1. Then, the particle reduces its own criticality value by *CL*. The particle then relocates itself. Two types of relocation were investigated: the first re-initializes the particle, while the second pushes the particle with high criticality a little further in the search space. According to Løvberg and Krink [2002], the first relocation approach produced better results when applied to the tested functions. SOC PSO outperformed PSO in one case out of the four cases used in the experiments. However, adding a tenth of the criticality value of a particle to its own inertia (w was set to 0.2) results in a significant improvement of the SOC PSO compared to PSO [Løvberg and Krink 2002].

Fitness-Distance Ratio based PSO (FDR-PSO)

Recently, Veeramachaneni *et al.* [2003] proposed a major modification to the way PSO operates by adding a new term to the velocity update equation. The new term allows each particle to move towards a particle in its neighborhood that has a better personal best position. The modified velocity update equation is defined as:

$$v_{i,j}(t+1) = wv_{i,j}(t) + \psi_1(y_{i,j}(t) - x_{i,j}(t)) + \psi_2(\hat{y}_j(t) - x_{i,j}(t)) + \psi_3(y_{\eta,j}(t) - x_{i,j}(t)) \qquad (2.25)$$

where ψ_1, ψ_2 and ψ_3 are user-specified parameters and each $y_{\eta,j}(t)$ is chosen by maximizing

$$\frac{f(\boldsymbol{x}_i(t)) - f(\boldsymbol{y}_\eta(t))}{\left|y_{\eta,j}(t) - x_{i,j}(t)\right|} \qquad (2.26)$$

where |.| represents the absolute value.

According to Veeramachaneni *et al.* [2003], FDR-PSO decreases the possibility of premature convergence and thus is less likely to be trapped in local optima. In addition, FDR-PSO (using $\psi_1 = \psi_2 = 1$ and $\psi_3 = 2$) outperformed PSO and several other variations of PSO, namely, ARPSO, DPSO, SOC PSO and Multi Swarm PSO [Løvberg *et al.* 2001], in different tested benchmark problems [Veeramachaneni *et al.* 2003].

2.7 Ant Systems

Another population-based stochastic approach is Ant Systems. Ant Systems were first introduced by Dorigo [1992] and Dorigo *et al.* [1991] to solve some difficult combinatorial optimization problems [Dorigo *et al.* 1999]. Ant systems were inspired by the observation of real ant colonies. In real ant colonies, ants communicate with each other indirectly through depositing a chemical substance, called *pheromone*. Ants use, for example, pheromones to find the shortest path to food. This indirect way of communication via pheromones is called *stigmergy* [Dorigo *et al*. 1999].

Using Ant Colony Optimization (ACO), a finite size colony of artificial ants cooperate with each other via stigmergy to find quality solutions to optimization problems. Good solutions result from the cooperation of the artificial ants. ACO was applied to a wide range of optimization problems such as the traveling salesman problem, and routing and load balancing in packet switched networks with encouraging results [Dorigo *et al*. 1999]. More details about Ant Systems and their applications can be found in Bonabeau *et al*. [1999] and Dorigo and Di Caro [1999]. Ant systems and their applications are outside the scope of this thesis.

2.8 Conclusions

This chapter provided a short overview of optimization and optimization methods with a special emphasis on PSO. From the discussed methods, PSO (and GA for comparison purposes) is used in this thesis to optimize a set of problems in the field of pattern recognition and image processing. These problems are introduced in the next chapter.

Chapter 3

Problem Definition

This chapter reviews the problems addressed in this thesis in sufficient detail. First the clustering problem is defined and different clustering concepts and approaches are discussed. This is followed by defining image segmentation in addition to presenting various image segmentation methods. A survey of color image quantization and approaches to quantization are then presented. This is followed by a brief introduction to spectral unmixing.

3.1 The Clustering Problem

Data clustering is the process of identifying natural groupings or clusters within multidimensional data based on some similarity measure (e.g. Euclidean distance) [Jain *et al*. 1999; Jain *et al*. 2000]. It is an important process in pattern recognition and machine learning [Hamerly and Elkan 2002]. Furthermore, data clustering is a central process in Artificial Intelligence (AI) [Hamerly 2003]. Clustering algorithms are used in many applications, such as image segmentation [Coleman and Andrews 1979; Jain and Dubes 1988; Turi 2001], vector and color image quantization [Kaukoranta *et al*. 1998; Baek *et al*. 1998; Xiang 1997], data mining [Judd *et al*. 1998], compression [Abbas and Fahmy 1994], machine learning [Carpineto and Romano 1996], etc. A cluster is usually identified by a cluster center (or *centroid*) [Lee and Antonsson 2000]. Data clustering is a difficult problem in unsupervised pattern recognition as the clusters in data may have different shapes and sizes [Jain *et al*. 2000].

3.1.1 Definitions

The following terms are used in this thesis:

- A *pattern* (or *feature vector*), $\boldsymbol{z}$, is a single object or data point used by the clustering algorithm [Jain *et al*. 1999].
- A *feature* (or *attribute*) is an individual component of a pattern [Jain *et al*. 1999].
- A *cluster* is a set of similar patterns, and patterns from different clusters are not similar [Everitt 1974].
- *Hard* (or *Crisp*) clustering algorithms assign each pattern to one and only one cluster.
- *Fuzzy* clustering algorithms assign each pattern to each cluster with some degree of membership.
- A *distance measure* is a metric used to evaluate the similarity of patterns [Jain *et al*. 1999].

The clustering problem can be formally defined as follows (Veenman *et al*. 2003): Given a data set $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_p, \ldots, \boldsymbol{z}_{N_p}\}$ where $\boldsymbol{z}_p$ is a pattern in the N_d-dimensional feature space, and N_p is the number of patterns in $\boldsymbol{Z}$, then the clustering of $\boldsymbol{Z}$ is the partitioning of $\boldsymbol{Z}$ into K clusters $\{\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_K\}$ satisfying the following conditions:

- Each pattern should be assigned to a cluster, i.e.

  $\cup_{k=1}^{K} \boldsymbol{C}_k = \boldsymbol{Z}$

- Each cluster has at least one pattern assigned to it, i.e.

$C_k \neq \phi, \quad k = 1, \ldots, K$

- Each pattern is assigned to one and only one cluster (in case of hard clustering only), i.e.

  $C_k \cap C_{kk} = \phi$ where $k \neq kk$

3.1.2 Similarity Measures

As previously mentioned, clustering is the process of identifying natural groupings or clusters within multidimensional data based on some similarity measure. Hence, similarity measures are fundamental components in most clustering algorithms [Jain *et al.* 1999].

The most popular way to evaluate a similarity measure is the use of distance measures. The most widely used distance measure is the Euclidean distance defined as

$$d(\mathbf{z}_u, \mathbf{z}_w) = \sqrt{\sum_{j=1}^{N_d} (z_{u,j} - z_{w,j})^2} = \|\mathbf{z}_u - \mathbf{z}_w\| \tag{3.1}$$

Euclidean distance is a special case (when $\alpha = 2$) of the Minkowski metric [Jain *et al.* 1999] defined as

$$d^{\alpha}(\mathbf{z}_u, \mathbf{z}_w) = (\sum_{j=1}^{N_d} (z_{u,j} - z_{w,j})^{\alpha})^{1/\alpha} = \|\mathbf{z}_u - \mathbf{z}_w\|^{\alpha} \tag{3.2}$$

When $\alpha = 1$, the measure is referred to as the Manhattan distance [Hamerly 2003].

Clustering data of high dimensionality using the Minkowski metric is usually not efficient because the distance between the patterns increases with increase in dimensionality. Hence, the concepts of near and far become weaker [Hamerly 2003]. Furthermore, for the Minkowski metric, the largest-scaled feature tends to dominate the other features. This can be solved by normalizing the features to a common range [Jain *et al.* 1999]. One way to do this is by using the cosine distance (or vector dot product) which is the sum of the product of each component from two vectors defined as

$$< \boldsymbol{z}_u , \boldsymbol{z}_w > = \frac{\sum_{j=1}^{N_d} z_{u,j} z_{w,j}}{\|\boldsymbol{z}_u\| \|\boldsymbol{z}_w\|} \tag{3.3}$$

where $< \boldsymbol{z}_u , \boldsymbol{z}_w > \in [-1,1]$.

The cosine distance is actually not a distance but rather a similarity metric. In other words, the cosine distance measures the difference in the angle between two vectors not the difference in the magnitude between two vectors. The cosine distance is suitable for clustering data of high dimensionality [Hamerly 2003].

Another distance measure is the Mahalanobis distance defined as

$$d_{\mathrm{M}}(\boldsymbol{z}_u, \boldsymbol{z}_w) = (\boldsymbol{z}_u - \boldsymbol{z}_w)\Sigma^{-1}(\boldsymbol{z}_u - \boldsymbol{z}_w)^{\mathrm{T}} \tag{3.4}$$

where Σ is the covariance matrix of the patterns. The Mahalanobis distance gives different features different weights based on their variances and pairwise linear correlations. Thus, this metric implicitly assumes that the densities of the classes are multivariate Gaussian [Jain *et al*. 1999].

3.1.3 Clustering Techniques

Most clustering algorithms are based on two popular techniques known as *hierarchical* and *partitional* clustering [Frigui and Krishnapuram 1999; Leung *et al*. 2000]. In the following, an overview of both techniques is presented with an elaborate discussion of popular hierarchical and partitional clustering algorithms.

3.1.3.1 Hierarchical Clustering Techniques

Algorithms in this category generate a cluster tree (or *dendrogram*) by using heuristic splitting or merging techniques [Hamerly 2003]. A cluster tree is defined as "a tree showing a sequence of clustering with each clustering being a partition of the data set" [Leung *et al*. 2000]. Algorithms that use splitting to generate the cluster tree are called *divisive*. On the other hand, the more popular algorithms that use merging to generate the cluster tree are called *agglomerative*. Divisive hierarchical algorithms start with all the patterns assigned to a single cluster. Then, splitting is applied to a cluster in each stage until each cluster consists of one pattern. Contrary to divisive hierarchical algorithms, agglomerative hierarchical algorithms start with each pattern assigned to one cluster. Then, the two most similar clusters are merged together. This step is repeated until all the patterns are assigned to a single cluster [Turi 2001]. Several agglomerative hierarchical algorithms were proposed in the literature which differ in the way that the two most similar clusters are calculated. The two most popular agglomerative hierarchical algorithms are the *single link* [Sneath and Sokal 1973] and *complete link* [Anderberg 1973] algorithms. Single link algorithms merge the clusters whose distance between their closest patterns is the smallest. Complete link algorithms, on the other hand, merge the clusters whose distance between their most distant patterns is the smallest [Turi 2001]. In general, complete link algorithms generate compact clusters while single link algorithms generate elongated clusters. Thus, complete link algorithms are generally more useful than single link algorithms [Jain *et al.* 1999]. Another less popular agglomerative hierarchical algorithm is the *centroid* method [Anderberg 1973]. The centroid algorithm merges the clusters whose distance between their centroids is the smallest. One disadvantage of the centroid algorithm is that the characteristic of a very small cluster is lost when merged with a very large cluster [Turi 2001]. More details about traditional hierarchical clustering techniques can be found in Everitt [1974].

Recently, a hierarchical clustering approach to simulate the human visual system by modeling the blurring effect of lateral retinal interconnections based on scale space theory has been proposed by Leung *et al.* [2000]. The following paragraph provides the reader with a good idea about this approach as described by Leung *et al.* [2000]:

> "In this approach, a data set is considered as an image with each light point located at a datum position. As we blur this image, smaller light blobs merge into larger ones until the whole image becomes one light blob at a low level of resolution. By identifying each blob with a cluster, the blurring process generates a family of clustering along the hierarchy."

According to Leung *et al.* [2000], this approach has several advantages, including:

- it is not sensitive to initialization,
- it is robust in the presence of noise in the data set, and
- it generates clustering that is similar to that perceived by human eyes.

In general, hierarchical clustering techniques have the following advantages [Frigui and Krishnapuram 1999]:

- the number of clusters need not to be specified *a priori*, and
- they are independent of the initial conditions.

However, hierarchical clustering techniques generally suffer from the following drawbacks:

- They are computationally expensive (time complexity is $O(N_p^2 \log N_p)$ and space complexity is $O(N_p^2)$ [Turi 2001]). Hence, they are not suitable for very large data sets.
- They are static, i.e. patterns assigned to a cluster cannot move to another cluster.
- They may fail to separate overlapping clusters due to a lack of information about the global shape or size of the clusters.

3.1.3.2 Partitional Clustering Techniques

Partitional clustering algorithms divide the data set into a specified number of clusters. These algorithms try to minimize certain criteria (e.g. a square error function) and can therefore be treated as optimization problems. However, these optimization problems are generally NP-hard and combinatorial [Leung *et al*. 2000]. The advantages of hierarchical algorithms are the disadvantages of the partitional algorithms and *vice versa*. Because of their advantages, partitional clustering techniques are more popular than hierarchical techniques in pattern recognition [Jain *et al*. 2000], hence, this thesis concentrates on partitional techniques.

Partitional clustering algorithms are generally iterative algorithms that converge to local optima [Hamerly and Elkan 2002]. Employing the general form of iterative clustering used by Hamerly and Elkan [2002], the steps of an iterative clustering algorithm are:

```
1. Randomly initialize the $K$ cluster centroids
2. **Repeat**
  (a) **For** each pattern, $\boldsymbol{z}_p$, in the data set **do**
        Compute its membership $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p)$ to each centroid $\boldsymbol{m}_k$ and its weight $w(\boldsymbol{z}_p)$
      **endloop**

  (b) Recalculate the $K$ cluster centroids, using
        $\boldsymbol{m}_k = \frac{\sum_{\forall z_p} u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) w(\boldsymbol{z}_p) \boldsymbol{z}_p}{\sum_{\forall z_p} u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) w(\boldsymbol{z}_p)}$    (3.5)
  **until** a stopping criterion is satisfied.
```

In the above algorithm, $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p)$ is the membership function which quantifies the membership of pattern $\boldsymbol{z}_p$ to cluster k. The membership function, $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p)$, must satisfy the following constraints:

1) $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) \geq 0,\ p = 1,\ldots, N_p$ and $k = 1,\ldots, K$

2) $\sum_{k=1}^{K} u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) = 1,\ p = 1,\ldots, N_p$

Crisp clustering algorithms use a *hard* membership function (i.e. $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) \in \{0,1\}$), while fuzzy clustering algorithms use a *soft* member function (i.e. $u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) \in [0,1]$) [Hamerly and Elkan 2002].

The weight function, $w(\boldsymbol{z}_p)$, in equation (3.5) defines how much influence pattern $\boldsymbol{z}_p$ has in recomputing the centroids in the next iteration, where $w(\boldsymbol{z}_p) > 0$ [Hamerly and Elkan 2002]. The weight function was proposed by Zhang [2000].

Different stopping criteria can be used in an iterative clustering algorithm, for example:

- stop when the change in centroid values are smaller than a user-specified value,
- stop when the quantization error is small enough, or
- stop when a maximum number of iterations has been exceeded.

In the following, popular iterative clustering algorithms are described by defining the membership and weight functions in equation (3.5).

The K-means Algorithm

The most widely used partitional algorithm is the iterative K-means approach [Forgy 1965]. The objective function that the K-means optimizes is

$$J_{\text{K-means}} = \sum_{k=1}^{K} \sum_{\forall \boldsymbol{z}_p \in \boldsymbol{C}_k} d^2(\boldsymbol{z}_p, \boldsymbol{m}_k) \tag{3.6}$$

Hence, the K-means algorithm minimizes the intra-cluster distance [Hamerly and Elkan 2002]. The K-means algorithm starts with K centroids (initial values for the centroids are randomly selected or derived from *a priori* information). Then, each pattern in the data set is assigned to the closest cluster (i.e. closest centroid). Finally, the centroids are recalculated according to the associated patterns. This process is repeated until convergence is achieved.

The membership and weight functions for K-means are defined as

$$u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) = \begin{cases} 1 & \text{if } d^2(\boldsymbol{z}_p, \boldsymbol{m}_k) = \arg\min_k \{d^2(\boldsymbol{z}_p, \boldsymbol{m}_k)\} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$w(\boldsymbol{z}_p) = 1 \tag{3.8}$$

Hence, K-means has a hard membership function. Furthermore, K-means has a constant weight function, thus, all patterns have equal importance [Hamerly and Elkan 2002].

The K-means algorithm has the following main advantages [Turi 2001]:

- it is very easy to implement, and
- its time complexity is O(N_p) making it suitable for very large data sets.

However, the K-means algorithm has the following drawbacks [Davies 1997]:

- the algorithm is data-dependent,

- it is a greedy algorithm that depends on the initial conditions, which may cause the algorithm to converge to suboptimal solutions, and
- the user needs to specify the number of clusters in advance.

The K-medoids algorithm is similar to K-means with one major difference, namely, the centroids are taken from the data itself [Hamerly 2003]. The objective of K-medoids is to find the most centrally located patterns within the clusters [Halkidi *et al.* 2001]. These patterns are called *medoids*. Finding a single medoid requires $O(N_p^2)$. Hence, K-medoids is not suitable for moderately large data sets.

The Fuzzy C-means Algorithm

A fuzzy version of K-means, called Fuzzy C-means (FCM) (sometimes called fuzzy K-means), was proposed by Bezdek [1980; 1981]. FCM is based on a fuzzy extension of the least-square error criterion. The advantage of FCM over K-means is that FCM assigns each pattern to each cluster with some degree of membership (i.e. fuzzy clustering). This is more suitable for real applications where there are some overlaps between the clusters in the data set. The objective function that the FCM optimizes is

$$J_{\text{FCM}} = \sum_{k=1}^{K} \sum_{p=1}^{N_p} u_{k,p}^{q} d^2(\boldsymbol{z}_p, \boldsymbol{m}_k) \tag{3.9}$$

where q is the fuzziness exponent, with $q \geq 1$. Increasing the value of q will make the algorithm more fuzzy; $u_{k,p}$ is the membership value for the p^{th} pattern in the k^{th} cluster satisfying the following constraints:

1) $u_{k,p} \geq 0$, $p = 1,\ldots, N_p$ and $k = 1,\ldots, K$

2) $\sum_{k=1}^{K} u_{k,p} = 1$, $p = 1,\ldots, N_p$

The membership and weight functions for FCM are defined as [Hamerly and Elkan 2002]

$$u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) = \frac{\left\|\boldsymbol{z}_p - \boldsymbol{m}_k\right\|^{-2/(q-1)}}{\sum_{k=1}^{K}\left\|\boldsymbol{z}_p - \boldsymbol{m}_k\right\|^{-2/(q-1)}} \tag{3.10}$$

$$w(\boldsymbol{z}_p) = 1 \tag{3.11}$$

Hence, FCM has a soft membership function and a constant weight function. In general, FCM performs better than K-means [Hamerly 2003] and it is less affected by the presence of uncertainty in the data [Liew *et al*. 2000]. However, as in K-means it requires the user to specify the number of clusters in the data set. In addition, it may converge to local optima [Jain *et al*. 1999].

Krishnapuram and Keller [1993; 1996] proposed a possibilistic clustering algorithm, called *possibilistic* C-means. Possibilistic clustering is similar to fuzzy clustering; the main difference is that in possibilistic clustering the membership values may not sum to one [Turi 2001]. Possibilistic C-means works well in the presence of noise in the data set. However, it has several drawbacks, namely [Turi 2001],

- it is likely to generate coincident clusters,
- it requires the user to specify the number of clusters in advance,
- it converges to local optima, and
- it depends on initial conditions.

The Gaussian Expectation-Maximization Algorithm

Another popular clustering algorithm is the Expectation-Maximization (EM) algorithm [McLachlan and Krishnan 1997; Rendner and Walker 1984; Bishop 1995]. EM is used for parameter estimation in the presence of some unknown data [Hamerly 2003]. EM partitions the data set into clusters by determining a mixture of Gaussians fitting the data set. Each Gaussian has a mean and covariance matrix [Alldrin *et al.* 2003]. The objective function that the EM optimizes as defined by Hamerly and Elkan [2002] is

$$J_{\text{EM}} = -\sum_{p=1}^{N_p} \log\left(\sum_{k=1}^{K} p(\boldsymbol{z}_p \mid \boldsymbol{m}_k)\, p(\boldsymbol{m}_k)\right) \tag{3.12}$$

where $p(\boldsymbol{z}_p \mid \boldsymbol{m}_k)$ is the probability of $\boldsymbol{z}_p$ given that it is generated by a Gaussian distribution with centroid $\boldsymbol{m}_k$, and $p(\boldsymbol{m}_k)$ is the prior probability of centroid $\boldsymbol{m}_k$.

The membership and weight functions for EM are defined as [Hamerly and Elkan 2002]

$$u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) = \frac{p(\boldsymbol{z}_p \mid \boldsymbol{m}_k)\, p(\boldsymbol{m}_k)}{p(\boldsymbol{z}_p)} \tag{3.13}$$

$$w(\boldsymbol{z}_p) = 1 \tag{3.14}$$

Hence, EM has a soft membership function and a constant weight function. The algorithm starts with an initial estimate of the parameters. Then, an *expectation* step is applied where the known data values are used to compute the expected values of the unknown data [Hamerly 2003]. This is followed by a *maximization* step where the known and expected values of the data are used to generate a new estimate of the parameters. The expectation and maximization steps are repeated until convergence.

Results from Veenman *et al*. [2002] and Hamerly [2003] showed that K-means performs comparably to EM. Furthermore, Aldrin *et al*. [2003] stated that EM fails on high-dimensional data sets due to numerical precision problems. They also observed that Gaussians often collapsed to delta functions [Alldrin *et al*. 2003]. In addition, EM depends on the initial estimate of the parameters [Hamerly 2003; Turi 2001] and it requires the user to specify the number of clusters in advance. Moreover, EM assumes that the density of each cluster is Gaussian which may not always be true [Ng *et al*. 2001].

The K-harmonic Means Algorithm

Recently, Zhang and colleagues [1999; 2000] proposed a novel algorithm called K-harmonic means (KHM), with promising results. In KHM, the harmonic mean of the distance of each cluster center to every pattern is computed. The cluster centroids are then updated accordingly. The objective function that the KHM optimizes is

$$J_{\text{KHM}} = \sum_{p=1}^{N_p} \frac{K}{\sum_{k=1}^{K} \frac{1}{\left\| \boldsymbol{z}_p - \boldsymbol{m}_k \right\|^{\alpha}}} \tag{3.15}$$

where α is a user-specified parameter, typically $\alpha \geq 2$.

The membership and weight functions for KHM are [Hamerly and Elkan 2002]

$$u(\boldsymbol{m}_k \mid \boldsymbol{z}_p) = \frac{\left\| \boldsymbol{z}_p - \boldsymbol{m}_k \right\|^{-\alpha-2}}{\sum_{k=1}^{K} \left\| \boldsymbol{z}_p - \boldsymbol{m}_k \right\|^{-\alpha-2}} \tag{3.16}$$

$$w(\boldsymbol{z}_p) = \frac{\sum_{k=1}^{K} \left\| \boldsymbol{z}_p - \boldsymbol{m}_k \right\|^{-\alpha-2}}{\left(\sum_{k=1}^{K} \left\| \boldsymbol{z}_p - \boldsymbol{m}_k \right\|^{-\alpha} \right)^2} \tag{3.17}$$

Hence, KHM has a soft membership function and a varying weight function. KHM assigns higher weights for patterns that are far from all the centroids to help the centroids in covering the data [Hamerly and Elkan 2002].

Contrary to K-means, KHM is less sensitive to initial conditions and does not have the problem of collapsing Gaussians exhibited by EM [Alldrin *et al*. 2003]. Experiments conducted by Zhang *et al*. [1999], Zhang [2000] and Hamerly and Elkan [2002] showed that KHM outperformed K-means, FCM (according to Hamerly and Elkan [2002]) and EM.

Hybrid 2

Hamerly and Elkan [2002] proposed a variation of KHM, called Hybrid 2 (H2), which uses the soft membership function of KHM (i.e. equation (3.16)) and the constant weight function of K-means (i.e. equation (3.8)). Hamerly and Elkan [2002] showed that H2 outperformed K-means, FCM and EM. However, KHM, in general, performed slightly better than H2.

K-means, FCM, EM, KHM and H2 are linear time algorithms (i.e. their time complexity is O(N_p)) making them suitable for very large data sets. According to

Hamerly [2003], FCM, KHM and H2 - all use soft membership functions - are the best available clustering algorithms.

Non-iterative Partitional Algorithms

Another category of unsupervised partitional algorithms includes the non-iterative algorithms. The most widely used non-iterative algorithm is MacQueen's K-means algorithm [MacQueen 1967]. This algorithm works in two phases: the first phase finds the centroids of the clusters, and the second clusters the patterns. Competitive Learning (CL) updates the centroids sequentially by moving the closest centroid toward the pattern being classified [Scheunders, A Comparison 1997]. These algorithms suffer the drawback of being dependent on the order in which the data points are presented. To overcome this problem, data points are presented in a random order [Davies 1997]. In general, iterative algorithms are more effective than non-iterative algorithms, since they are less dependent on the order in which data points are presented.

3.1.3.3 Other Clustering Techniques

Another type of clustering algorithms includes the *Nearest Neighbor* clustering algorithm proposed by Lu and Fu [1978]. For each unclassified pattern, the algorithm finds the nearest classified pattern whose distance from the unclassified pattern is less than a pre-specified threshold. The unclassified pattern is then assigned to the cluster of the classified pattern. This process is repeated until all the patterns become classified or no further assignments can occur [Jain *et al*. 1999].

Recently, a new type of clustering algorithms called *spectral* clustering algorithms [Ng *et al*. 2001; Bach and Jordan 2003] has been proposed by computer vision researchers and graph theorists. Spectral clustering is based on spectral graph theory [Chung 1997] where a graph representing the data (the graph is analogous to a matrix of the distance between the patterns in the data set) is searched by the spectral clustering algorithm for globally optimal cuts [Hamerly 2003]. One major advantage of spectral clustering is that it can generate arbitrary-shaped clusters. However, spectral clustering suffers from two major drawbacks [Hamerly 2003]:

- It is computationally expensive (its time complexity is $O(N_p^3 + N_d\, N_p^2)$). Hence, they are not suitable for moderately large data sets.
- It requires the user to specify a kernel width parameter which has a profound effect on the result of the spectral clustering algorithm. Choosing a good value for this parameter is usually difficult.

The *mean shift* algorithm [Comaniciu and Meer 2002] also automatically finds the number of clusters in a data set and can work with arbitrary shaped clusters. The mean shift algorithm starts with a number of kernel estimators in the input space. These estimators are then repeatedly moved towards areas of higher density. When all the kernels reached stability, all the kernels that are near to each other are grouped together. The data is then segmented based on where each kernel started.

The mean shift algorithm has the following problems, [Hamerly 2003]:

- it has to find a way to group kernels and patterns, and

- as in spectral clustering, the mean shift algorithm requires the user to specify a kernel width parameter which has a profound effect on the result of the algorithm.

3.1.4 Clustering Validation Techniques

The main objective of cluster validation is to evaluate clustering results in order to find the best partitiong of a data set [Halkidi *et al*. 2001]. Hence, cluster validity approaches are used to quantitatively evaluate the result of a clustering algorithm [Halkidi *et al*. 2001]. These approaches have representative indices, called *validity indices*. The traditional approach to determine the "optimum" number of clusters is to run the algorithm repetitively using different input values and to select the partitioning of data resulting in the best validity measure [Halkidi and Vazirgiannis 2001].

Two criteria that have been widely considered sufficient in measuring the quality of data partitioning, are [Halkidi *et al*. 2001]

- *Compactness*: patterns in one cluster should be similar to each other and different from patterns in other clusters. The variance of patterns in a cluster gives an indication of compactness.
- *Separation*: clusters should be well-separated from each other. The Euclidean distance between cluster centroids gives an indication of cluster separation.

There are several validity indices; a thorough survey of validity indices can be found in Halkidi *et al*. [2001]. In the following, some representative indices are discussed.

Dunn [1974] proposed a well known cluster validity index that identifies compact and well separated clusters. The main goal of Dunn's index is to maximize inter-cluster distances (i.e. separation) while minimizing intra-cluster distances (i.e. increase compactness). The Dunn index is defined as

$$D = \min_{k=1,\ldots,K}\left\{ \min_{kk=k+1,\ldots,K}\left(\frac{dist(\boldsymbol{C}_k,\boldsymbol{C}_{kk})}{\max_{a=1,\ldots,K} diam(\boldsymbol{C}_a)} \right) \right\} \tag{3.18}$$

where $dist(\boldsymbol{C}_k,\boldsymbol{C}_{kk})$ is the dissimilarity function between two clusters $\boldsymbol{C}_k$ and $\boldsymbol{C}_{kk}$ defined as

$$dist(\boldsymbol{C}_k,\boldsymbol{C}_{kk}) = \min_{\boldsymbol{u}\in\boldsymbol{C}_k,\boldsymbol{w}\in\boldsymbol{C}_{kk}} d(\boldsymbol{u},\boldsymbol{w}),$$

where $d(\boldsymbol{u}, \boldsymbol{w})$ is the Euclidean distance between $\boldsymbol{u}$ and $\boldsymbol{v}$; $diam(\boldsymbol{C})$ is the diameter of a cluster, defined as

$$diam(\boldsymbol{C}) = \max_{\boldsymbol{u},\boldsymbol{w}\in\boldsymbol{C}} d(\boldsymbol{u},\boldsymbol{w})$$

An "optimal" value of K is the one that maximizes the Dunn's index. Dunn's index suffers from the following problems [Halkidi *et al*. 2001]:

- it is computationally expensive, and
- it is sensitive to the presence of noise.

Several Dunn-like indices were proposed in Pal and Biswas [1997] to reduce the sensitivity to the presence of noise.

Another well known index, proposed by Davies and Bouldin [1979], minimizes the average similarity between each cluster and the one most similar to it. The Davies and Bouldin index is defined as

$$DB = \frac{1}{K}\sum_{k=1}^{K} \max_{\substack{kk=1,\dots,K \\ k \neq kk}} \left(\frac{diam(\boldsymbol{C}_k) + diam(\boldsymbol{C}_{kk})}{dist(\boldsymbol{C}_k, \boldsymbol{C}_{kk})} \right) \tag{3.19}$$

An "optimal" value of K is the one that minimizes the DB index.

Recently, Turi [2001] proposed an index incorporating a multiplier function (to penalize the selection of a small number of clusters) to the ratio between intra-cluster and inter-cluster distances, with some promising results. The index is defined as

$$V = (c \times \mathrm{N}(2,1) + 1) \times \frac{\text{intra}}{\text{inter}} \tag{3.20}$$

where c is a user specified parameter and N(2,1) is a Gaussian distribution with mean 2 and standard deviation of 1. The "intra" term is the average of all the distances between each data point and its cluster centroid, defined as

$$\text{intra} = \frac{1}{N_p}\sum_{k=1}^{K}\sum_{\forall \boldsymbol{u} \in \boldsymbol{C}_k} \left\| \boldsymbol{u} - \boldsymbol{m}_k \right\|^2$$

This term is used to measure the compactness of the clusters. The "inter" term is the minimum distance between the cluster centroids, defined as

$$\text{inter} = min\{\left\| \boldsymbol{m}_k - \boldsymbol{m}_{kk} \right\|^2\}, \forall\, k = 1,\dots,K-1 \text{ and } kk = k+1,\dots,K.$$

This term is used to measure the separation of the clusters. An "optimal" value of K is the one that minimizes the V index.

According to Turi [2001], this index performed better than both Dunn's index and the index of Davies and Bouldin on the tested cases.

Two recent validity indices are *S_Dbw* [Halkidi and Vazirgiannis 2001] and *CDbw* [Halkidi and Vazirgiannis 2002]. *S_Dbw* measures the compactness of a data set by the cluster variance, whereas separation is measured by the density between clusters. The *S_Dbw* index is defined as

$$S\_Dbw = scat(K) + Dens\_bw(K) \tag{3.21}$$

The first term is the average scattering of the clusters which is a measure of compactness of the clusters, defined as

$$scat(K) = \frac{1}{K}\sum_{k=1}^{K} \|\sigma(\boldsymbol{C}_k)\| / \|\sigma(\boldsymbol{Z})\|$$

where $\sigma(\boldsymbol{C}_k)$ is the variance of cluster $\boldsymbol{C}_k$ and $\sigma(\boldsymbol{Z})$ is the variance of data set $\boldsymbol{Z}$; $||\boldsymbol{z}||$ is defined as $||\boldsymbol{z}|| = (\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z})^{1/2}$, where $\boldsymbol{z}$ is a vector.

The second term in equation (3.21) evaluates the density of the area between the two clusters in relation to the density of the two clusters. Thus, the second term is a measure of the separation of the clusters, defined as

$$Dens\_bw(K) = \frac{1}{K(K-1)}\sum_{k=1}^{K}\left[\sum_{\substack{kk=1\\ k\neq kk}}^{K} \frac{density(\boldsymbol{b}_{k,kk})}{max\{density(\boldsymbol{C}_k), density(\boldsymbol{C}_{kk})\}}\right]$$

where $\boldsymbol{b}_{k,kk}$ is the middle point of the line segment defined by $\boldsymbol{m}_k$ and $\boldsymbol{m}_{kk}$. The term $density(\boldsymbol{b})$ is defined as

$$density(\boldsymbol{b}) = \sum_{ll=1}^{n_{k,kk}} f(\boldsymbol{z}_{ll}, \boldsymbol{b})$$

where $n_{k,kk}$ is the total number of patterns in clusters $\boldsymbol{C}_k$ and $\boldsymbol{C}_{kk}$ (i.e. $n_{k,kk} = n_k + n_{kk}$). The function $f(\boldsymbol{z},\boldsymbol{b})$ is defined as

$$f(\boldsymbol{z},\boldsymbol{b}) = \begin{cases} 0 & \text{if } d(\boldsymbol{z},\boldsymbol{b}) > \sigma \\ 1 & \text{otherwise} \end{cases}$$

where

$$\sigma = \frac{1}{K}\sqrt{\sum_{k=1}^{K}\|\sigma(\boldsymbol{C}_k)\|}$$

An "optimal" value of K is the one that minimizes the *S_Dbw* index. Halkidi and Vazirgiannis [2001] showed that, in tested cases, *S_Dbw* successfully found the "optimal" number of clusters whereas other well-known indices often failed to do so. However, *S_Dbw* does not work properly for arbitrary shaped clusters.

To address this problem, Halkidi and Vazirgiannis [2002] proposed a multi-representative validity index, *CDbw*, in which each cluster is represented by a user-specified number of points, instead of one representative as is done in *S_Dbw*. Furthermore, *CDbw* uses intra-cluster density to measure the compactness of a data set, and uses the density between clusters to measure their separation.

More recently, Veenman *et al.* [2002; 2003] proposed a validity index that minimizes the intra-cluster variability while constraining the intra-cluster variability of the union of the two clusters. The sum of squared error is used to minimize the intra-cluster variability while a minimum variance for the union of two clusters is used to implement the joint intra-cluster variability. The index is defined as

$$IV = min\sum_{k=1}^{K} n_k Var(\boldsymbol{C}_k) \tag{3.22}$$

where n_k is the number of patterns in cluster $\boldsymbol{C}_k$ and

$$Var(\boldsymbol{C}_k) = \frac{1}{n_k}\sum_{z_p \in \boldsymbol{C}_k}\|\boldsymbol{z}_p - \boldsymbol{m}_k\|^2$$

such that

$$Var(\boldsymbol{C}_k \cup \boldsymbol{C}_{kk}) \geq \sigma^2_{\max}, \quad \forall \boldsymbol{C}_k, \boldsymbol{C}_{kk}, k \neq kk$$

where $\sigma^2_{\max}$ is a user-specified parameter. This parameter has a profound effect on the final result.

The above validity indices are suitable for hard clustering. Validity indices have been developed for fuzzy clustering. The interested reader is referred to Halkidi *et al*. [2001] for more information.

3.1.5 Determining the Number of Clusters

Most clustering algorithms require the number of clusters to be specified in advance [Lee and Antonsson 2000; Hamerly and Elkan 2003]. Finding the "optimum" number of clusters in a data set is usually a challenge since it requires *a priori* knowledge, and/or ground truth about the data, which is not always available. The problem of finding the optimum number of clusters in a data set has been the subject of several research efforts [Halkidi *et al*. 2001; Theodoridis and Koutroubas 1999], however, despite the amount of research in this area, the outcome is still unsatisfactory [Rosenberger and Chehdi 2000]. In the literature, many approaches to dynamically find the number of clusters in a data set were proposed. In this section, several dynamic clustering approaches are presented and discussed.

ISODATA (Iterative Self-Organizing Data Analysis Technique), proposed by Ball and Hall [1967], is an enhancement of the K-means algorithm (K-means is sometimes referred to as *basic* ISODATA [Turi 2001]). ISODATA is an iterative procedure that assigns each pattern to its closest centroids (as in K-means). However, ISODATA has the ability to merge two clusters if the distance between their centroids is below a user-specified threshold. Furthermore, ISODATA can split elongated clusters into two clusters based on another user-specified threshold. Hence, a major advantage of ISODATA compared to K-means is the ability to determine the number of clusters in a data set. However, ISODATA requires the user to specify the values of several parameters (e.g. the merging and splitting thresholds). These parameters have a profound effect on the performance of ISODATA making the result subjective [Turi 2001].

Dynamic Optimal Cluster-seek (DYNOC) [Tou 1979] is a dynamic clustering algorithm which is similar to ISODATA. DYNOC maximizes the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance. This is done by an iterative procedure with the added capability of splitting and merging. However, as in ISODATA, DYNOC requires the user to specify a value for a parameter that determines whether splitting is needed [Turi 2001].

Snob [Wallace 1984; Wallace and Dowe 1994] uses various methods to assign objects to clusters in an intelligent manner [Turi 2001]. After each assignment, a means of model selection called the Wallace Information Measure (also known as the Minimum Message Length) [Wallace and Boulton 1968; Oliver and Hand 1994] is calculated and based on this calculation the assignment is accepted or rejected. Snob can split/merge and move points between clusters, thereby allowing it to determine the number of clusters in a data set.

Bischof *et al.* [1999] proposed an algorithm based on K-means which uses a similar concept to the Wallace Information Measure called the Minimum Description Length [Rissanen 1978] framework. The algorithm starts with a large value for K and proceeds to remove centroids when this removal results in a reduction of the description length. K-means is used between the steps that reduce K.

Modified Linde-Buzo-Gray (MLBG), proposed by Rosenberger and Chehdi [2000], improves K-means by automatically finding the number of clusters in data set by using intermediate results. MLBG is an iterative procedure that starts with K clusters. In each iteration, a cluster, $\boldsymbol{C}_k$, maximizing an intra-cluster distance measure is chosen for splitting. Two centroids are generated from the splitting process. The first centroid, $\boldsymbol{m}_1$, is initialized to the centroid of the original cluster, $\boldsymbol{C}_k$. The second cluster centroid, $\boldsymbol{m}_2$, is chosen to be the pattern in $\boldsymbol{C}_k$ which is the most distant from $\boldsymbol{m}_1$. K-means is then applied on the new K+1 centroids. The new set of centroids is accepted if it satisfies an evaluation criterion based on a dispersion measure. This process is repeated until no valid partition of the data can be obtained. One of the main problems with MLBG is that it requires the user to specify the values of four parameters, which have a profound effect on the resultant number of clusters.

Pelleg and Moore [2000] proposed another K-means based algorithm, called X-means that uses model selection. X-means starts by setting the number of clusters, K, to be the minimum number of clusters in the data set (e.g. K = 1). Then, K-means is applied on the K clusters. This is followed by a splitting process based on the Bayesian Information Criterion (BIC) [Kass and Wasserman 1995] defined as

$$BIC(\boldsymbol{C} \mid \boldsymbol{Z}) = \hat{l}(\boldsymbol{Z} \mid \boldsymbol{C}) - \frac{K(N_d + 1)}{2} \log N_p \tag{3.23}$$

where $\hat{l}(\boldsymbol{Z} \mid \boldsymbol{C})$ is the log-likelihood of the data set $\boldsymbol{Z}$ according to model $\boldsymbol{C}$. If the splitting process improves the BIC score the resulting split is accepted, otherwise it is rejected. Other scoring functions can also be used.

These two steps are repeated until a user-specified upper bound of K is reached. X-means searches over the range of values of K and reports the value with the best BIC score.

Recently, Huang [2002] proposed SYNERACT as an alternative approach to ISODATA. SYNERACT combines K-means with hierarchical descending approaches to overcome the drawbacks of K-means mentioned previously. Three concepts used by SYNERACT are:

- a hyperplane to split up a cluster into two smaller clusters and compute their centroids,
- iterative clustering to assign pixels into available clusters, and
- a binary tree to store clusters generated from the splitting process.

According to Huang [2002], SYNERACT is faster than and almost as accurate as ISODATA. Furthermore, it does not require the number of clusters and initial location of centroids to be specified in advance. However, SYNERACT requires the user to specify the values of two parameters that affect the splitting process.

Veenman *et al.* [2002] proposed a partitional clustering algorithm that finds the number of clusters in a data set by minimizing the clustering validity index defined in equation (3.22). This algorithm starts by initializing the number of clusters equal to the number of patterns in the data set. Then, iteratively, the clusters are split or merged according to a series of tests based on the validity index. According to Veenman *et al.* [2002], the proposed approach performed better than both K-means and EM algorithms. However, the approach suffers from the following drawbacks, namely

- it is computationally expensive, and
- it requires the user to specify a parameter for the validity index (already discussed in Section 3.1.4) which has a significant effect on the final results (although the authors provide a method to help the user in finding a good value for this parameter).

More recently, Hamerly and Elkan [2003] proposed another approach based on K-means, called G-means. G-means starts with a small value for *K*, and with each iteration splits up the clusters whose data do not fit a Gaussian distribution. Between each round of splitting, K-means is applied to the entire data set in order to refine the current solution. According to Hamerly and Elkan [2003], G-means works better than X-means, however, it works only for data having spherical and/or elliptical clusters. G-means is not designed to work for arbitrary-shaped clusters [Hamerly 2003].

Gath and Geva [1989] proposed an unsupervised fuzzy clustering algorithm based on a combination of FCM and fuzzy maximum likelihood estimation. The algorithm starts by initializing *K* to a user-specified lower bound of the number of clusters in the data set (e.g. $K = 1$). A modified FCM (that uses an unsupervised learning process to initialize the *K* centroids) is first applied to cluster the data. Using the resulting centroids, a fuzzy maximum likelihood estimation algorithm is then applied. The fuzzy maximum likelihood estimation algorithm uses an "exponential" distance measure based on maximum likelihood estimation [Bezdek 1981] instead of the Euclidean distance measure, because the exponential distance measure is more suitable for hyper-ellipsoidal clusters. The quality of the resulting clusters is then evaluated using a clustering validity index that is mainly based on a hyper-volume criterion which measures the compactness of a cluster. *K* is then incremented and the algorithm is repeated until a user-specified upper bound of *K* is reached. The value of *K* resulting in the best value of the validity index is considered to be the "optimal" number of clusters in the data set. Gath and Geva [1989] stated that their algorithm works well in cases of large variability of cluster shapes. However, the algorithm becomes more sensitive to local optima as the complexity increases. Furthermore, because of the exponential function, floating point overflows may occur [Su 2002].

Lorette *et al.* [2000] proposed an algorithm based on fuzzy clustering to dynamically determine the number of clusters in a data set. In this thesis, the proposed algorithm is referred as the Unsupervised Fuzzy Clustering (UFC) algorithm. A new objective function was proposed for this purpose, defined as

$$J_{\text{UFC}} = \sum_{k=1}^{K}\sum_{p=1}^{N_p} u_{k,p}^{q} d^2(\boldsymbol{z}_p, \boldsymbol{m}_k) - \beta \sum_{k=1}^{K} p_k \log(p_k) \tag{3.24}$$

where q is the fuzziness exponent, $u_{k,p}$ is the membership value for the pth pattern in the kth cluster, β is a parameter that decreases as the run progresses, and p_k is the *a priori* probability of cluster $\boldsymbol{C}_k$ defined as

$$p_k = \frac{1}{N_p} \sum_{p=1}^{N_p} u_{k,p} \tag{3.25}$$

The first term of equation (3.24) is the objective function of FCM which is minimized when each cluster consists of one pattern. The second term is an entropy term that is minimized when all the patterns are assigned to one cluster. Lorette *et al.* [2000] use this objective function to derive new update equations for the membership and centroid parameters.

The algorithm starts with a large number of clusters. Then, the membership values and centroids are updated using the new update equations. This is followed by applying equation (3.25) to update the *a priori* probabilities. If $p_k < \varepsilon$ then cluster k is discarded; ε is a user-specified parameter. This procedure is repeated until convergence. The drawback of this approach is that it requires the parameter ε to be specified in advance. The performance of the algorithm is sensitive to the value of ε.

Similar to UFC, Boujemaa [2000] proposed an algorithm, based on a generalization of the competitive agglomeration clustering algorithm introduced by Frigui and Krishnapuram [1997].

The fuzzy algorithms discussed above modify the objective function of FCM. In general, these approaches are sensitive to initialization and other parameters [Frigui and Krishnapuram 1999]. Frigui and Krishnapuram [1999] proposed a robust competitive clustering algorithm based on the process of competitive agglomeration. The algorithm starts with a large number of small clusters. Then, during the execution of the algorithm, adjacent clusters compete for patterns. Clusters losing the competition will eventually disappear [Frigui and Krishnapuram 1999]. However, this algorithm also requires the user to specify a parameter that has a significant effect on the generated result.

3.1.6 Clustering using Self-Organizing Maps

Kohonen's Self Organizing Maps (SOM) [Kohonen 1995] can be used to automatically find the number of clusters in a data set. The objective of SOM is to find regularities in a data set without any external supervision [Pandya and Macy 1996]. SOM is a single-layered unsupervised artificial neural network where input patterns are associated with output nodes via weights that are iteratively modified until a stopping criterion is met [Jain *et al.* 1999]. SOM combines competitive learning (in which different nodes in the Kohonen network compete to be the winner when an input pattern is presented) with a topological structuring of nodes, such that adjacent nodes tend to have similar weight vectors (this is done via lateral feedback)

[Mehrotra *et al.* 1997; Pandya and Macy 1996]. A general pseudo-code of SOM [Pandya and Macy 1996] is shown in Figure 3.1.

```
Let $\eta(t)$ be the learning rate parameter and $\Delta_w(t)$ be the neighborhood function
Randomly initialize the weight vectors, $\boldsymbol{w}_k(0)$
Initialize the learning rate $\eta(0)$ and the neighborhood function $\Delta_w(0)$
Repeat
  For each input pattern $\boldsymbol{z}_p$ do
    Select the node whose weight vector is closest (in terms of Euclidean distance) to $\boldsymbol{z}_p$ as the winning node
    Use competitive learning to train the weight vectors such that all the nodes within the neighborhood of the winning node are moved toward $\boldsymbol{z}_p$:
      $\boldsymbol{w}_k(t+1) = \begin{cases} \boldsymbol{w}_k(t) + \eta(t)[\boldsymbol{z}_p - \boldsymbol{w}_k(t)] & k \in \Delta_w(t) \\ \boldsymbol{w}_k(t) & \text{otherwise} \end{cases}$
  Endloop
  Linearly decrease $\eta(t)$ and reduce $\Delta_w(t)$
Until some convergence criteria are satisfied
```

Figure 3.1: General pseudo-code for SOM

In Figure 3.1, $\eta(t)$ starts relatively large (e.g. close to 1) then linearly decreases until it reaches a small user-specified value. The neighborhood function $\Delta_w(t)$ defines the neighborhood size surrounding the winning node. A large value of $\Delta_w(t)$ is used at the beginning of the training. This value is then reduced as the training progresses in order to get sharper clusters [Pandya and Macy 1996]. A typical neighborhood arrangement is the rectangular lattice shown in Figure 3.2 [Pandya and Macy 1996].

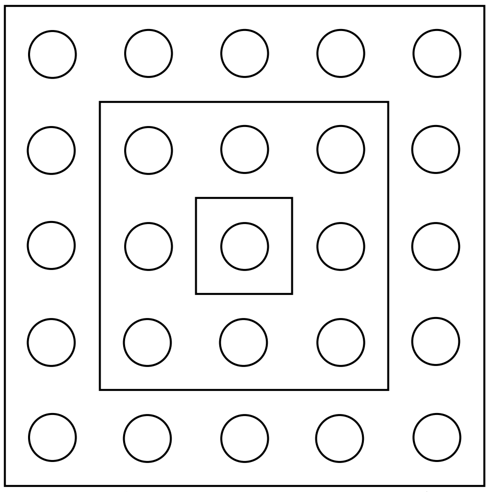


Figure 3.2: Rectangular Lattice arrangement of neighborhoods

SOM suffers from the following drawbacks [Jain *et al*. 1999]:

- It depends on the initial conditions.
- Its performance is affected by the learning rate parameter and the neighborhood function.
- It works well with hyper-spherical clusters only.
- It uses a fixed number of output nodes.
- It depends on the order in which the data points are presented. To overcome this problem, the choice of data points can be randomized during each iteration [Pandya and Macy 1996].

3.1.7 Clustering using Stochastic Algorithms

Simulated annealing (discussed in Section 2.3) has been used for clustering [Klein and Dubes 1989]. In general, a simulated annealing based clustering algorithm works as shown in Figure 3.3 [Jain *et al*. 1999].

```
An initial partition $P_0$ of the data set is randomly chosen
Repeat
  A neighbor of $P_0$ is chosen
  If the new partition is better than $P_0$ then
    move to the new partition
  Else
    move to the new partition with a probability that decreases as the algorithm
    progresses.
Until a stopping criterion is satisfied
```

Figure 3.3: General simulated annealing based clustering algorithm

One problem with simulated annealing is that it is very slow in finding an optimal solution [Jain *et al*. 1999].

Tabu search (discussed in Section 2.3) has also been used for hard clustering [Al-Sultan 1995] and fuzzy clustering [Delgado *et al*. 1997] with encouraging results. A hybrid approach combining both K-means and tabu search that performs better than both K-means and tabu search was proposed by Frnti *et al*. [1998]. Recently, Chu and Roddick [2003] proposed a hybrid approach combining both tabu search and simulated annealing that outperforms the hybrid proposed by Frnti *et al*. [1998].

However, the performance of simulated annealing and tabu search depends on the selection of several control parameters [Jain *et al*. 1999].

Most clustering approaches discussed so far perform local search to find a solution to a clustering problem. Evolutionary algorithms (discussed in Section 2.4) which perform global search have also been used for clustering [Jain *et al*. 1999]. Raghavan and Birchand [1979] used GAs to minimize the squared error of a clustering solution. In this approach, each chromosome represents a partition of N_p patterns into K clusters. Hence, the size of each chromosome is N_p. This representation has a major drawback in that it increases the search space by a factor of $K!$. The crossover operator may also result in inferior offspring [Jain *et al*. 1999].

Babu and Murty [1993] proposed a hybrid approach combining K-means and GAs that performed better than the GA. In this approach, a GA is only used to feed K-means with good initial centroids [Jain *et al*. 1999].

Recently, Maulik and Bandyopadhyay [2000] proposed a GA-based clustering where each chromosome represents K centroids. Hence, a floating point representation is used. The fitness function is defined as the inverse of the objective function of K-means (refer to equation (3.6)). The GA-based clustering algorithm is summarized in Figure 3.4.

According to Maulik and Bandyopadhyay [2000], this approach outperformed K-means on the tested cases. One drawback of this approach is that it requires the user to specify the number of clusters in advance.

```
1. Initialize each chromosome to contain $K$ randomly chosen centroids from the
   data set
2. For $t = 1$ to $t_{\max}$
(a) For each chromosome $i$
    (i) Assign each pattern to the cluster with the closest centroid
    (ii) Recalculate the $K$ cluster centroids of chromosome $i$ as the means of their
         patterns
    (iii) Calculate the fitness of chromosome $i$

 (b) Apply roulette wheel selection
 (c) Apply single point crossover with probability $p_c$
 (d) Apply mutation with probability $p_m$. The mutation operator is defined as
     $\boldsymbol{x} = \boldsymbol{x} \pm (r + \gamma)\boldsymbol{x}$
     where $r \sim U(0,1)$ and $\gamma$ is a user-specified parameter such that $\gamma \in (0,1)$
```

Figure 3.4: General pseudo-code for GA-based clustering algorithm

Lee and Antonsson [2000] used an evolution strategy (ES) to dynamically cluster a data set. The proposed ES implemented variable length individuals to search for both the centroids and the number of clusters. Each individual represents a set of centroids. The length of each individual is randomly chosen from a user-specified range of cluster numbers. The centroids of each individual are then randomly initialized. Mutation is applied to the individuals by adding/subtracting a Gaussian random variable with zero mean and unit standard deviation. Two point crossover is also used as a "length changing operator". A (10+60) ES selection is used where 10 is the number of parents and 60 is the number of offspring generated in each generation. The best ten individuals from the set of parents and offspring are used for the next generation. A modification of the mean square error is used as the fitness function, defined as

$$J_{\text{ES}} = \sqrt{K+1}\sum_{k=1}^{K}\sum_{\forall z_p \in C_k} d(\boldsymbol{z}_p, \boldsymbol{m}_k) \tag{3.26}$$

The modification occurs by multiplying the mean square error by a constant corresponding to the square root of the number of clusters. This constant is used to penalize a large value of *K*. According to Lee and Antonsson [2000], the results are promising. However, the proposed algorithm needs to be compared with other dynamic clustering approaches and its performance needs to be investigated as the dimension increases.

In general, evolutionary approaches have several advantages, namely [Jain *et al.* 1999]:

- they are global search approaches,
- they are suitable for parallel processing, and
- they can work with a discontinuous criterion function.

However, evolutionary approaches generally suffer from the following drawbacks [Jain *et al.* 1999]:

- they require the user to specify the values of a set of parameters (e.g. population size, p_c, p_m, etc.) for each specific problem, and

- the execution time of EAs is significantly higher than the execution time of other traditional clustering algorithms (e.g. K-means and FCM), especially when applied to large data sets.

3.1.8 Unsupervised Image Classification

Image classification is the process of identifying groups of similar image primitives [Puzicha *et al*. 2000]. These image primitives can be pixels, regions, line elements and so on, depending on the problem encountered.

There are two main approaches to image classification: supervised and unsupervised. In the supervised approach, the number and the numerical characteristics (e.g. mean and variance) of the classes in the image are known in advance (by the analyst) and used in the training step, which is followed by the classification step. There are several popular supervised algorithms such as the minimum-distance-to-mean, parallelepiped and the Gaussian maximum likelihood classifiers [Lillesand and Kiefer 1994]. In the unsupervised approach the classes are unknown and the approach starts by partitioning the image data into groups (or clusters), according to a similarity measure, which can be compared with reference to data by an analyst and used to segment the image.

Therefore, unsupervised classification is a special case of the general clustering problem where the data set is an image (or a set of images) and the patterns are the pixels of the image(s).

In general, the unsupervised approach has several advantages over the supervised approach, namely [Davies 1997]

- For unsupervised approaches, there is no need for an analyst to specify in advance all the classes in the image data set. The clustering algorithm automatically finds distinct clusters, which dramatically reduces the work of the analyst.
- The characteristics of the objects being classified can vary with time; the unsupervised approach is an excellent way to monitor these changes.
- Some characteristics of objects may not be known in advance. The unsupervised approach automatically flags these characteristics.

3.2 Image Segmentation using Clustering

Image segmentation is a fundamental process in several image processing and computer vision applications. It can be considered as the first low-level processing step in image processing and pattern recognition [Cheng *et al.* 2001]. Image segmentation is defined as the process of dividing an image into disjoint homogenous regions. These homogenous regions should represent objects or parts of them [Lucchese and Mitra 2001]. The homogeneity of the regions is measured using some image property (e.g. pixel intensity) [Jain *et al.* 1999]. Image segmentation can be formally defined as follows:

Given an image I and a homogeneity predicate P. The segmentation of image I is the partitioning of I into K regions, $\{R_1, R_2,\ldots,R_K\}$, satisfying the following conditions:

- Each pixel in the image should be assigned to a region, i.e.

  $\cup_{k=1}^{K} R_k = I$

- Each pixel is assigned to one and only one region, i.e.

$R_k \cap R_{kk} = \phi$ where $k \neq kk$

- Each region satisfies homogeneity predicate P, i.e.

  $P(R_k) = \text{True}, \ \forall k = 1, \ldots, K$

- Two different regions can not satisfy P, i.e.

  $P(R_k \cup R_{kk}) = \text{False}$ where $k \neq kk$

There are many techniques for image segmentation in the literature; details can be found in Fu and Mui [1981], Pal and Pal [1993], Cheng *et al.* [2001], Lucchese and Mitra [2001] and Turi [2001]. In general, these techniques can be categorized into thresholding, edge-based, region growing and clustering techniques [Turi 2001]. Each of these categories are discussed in the following sections.

3.2.1 Thresholding Techniques

Thresholding [Gonzalez and Woods 1992; Jain *et al.* 1995] is the simplest image segmentation technique. In its simplest version an image is divided into two segments: object and background by specifying a threshold. A pixel above the threshold is assigned to one segment and a pixel below the threshold is assigned to the other segment. For more sophisticated images multiple thresholds can be used.

3.2.2 Edge-based Techniques

In edge-based techniques [Gonzalez and Woods 1992; Jain *et al.* 1995; Kwok and Constantinides 1997], segmentation is achieved by finding the edges of the regions.

This is usually accomplished by moving a mask (e.g. a 3×3 window) over the image to detect local changes in the image intensity.

3.2.3 Region growing Techniques

In region growing [Gonzalez and Woods 1992; Jain *et al*. 1995; Fuh *et al*. 2000], a set of seed pixels are chosen. Neighboring pixels of a seed are agglomerated if they satisfy a homogeneity criterion. This is repeated until no more pixels can be added to the region. This approach has some problems [Turi 2001]:

- The selection of the seed pixels which is not a straightforward task.
- The selection of the homogeneity criterion.

Region splitting and merging divide the image into regions. A region is then split if it does not satisfy a homogeneity condition. Regions can also be merged if their merging results in a region that satisfies some condition. This is repeated until no more splitting and merging can occur [Gonzalez and Woods 1992].

3.2.4 Clustering Techniques

Image segmentation can be treated as a clustering problem where features describing each pixel correspond to a pattern and an image region (i.e. segment) corresponds to a cluster [Jain *et al*. 1999]. This similarity is obvious by comparing the clustering problem definition (refer to section 3.1.1) and the image segmentation problem definition (refer to section 3.2). Therefore, clustering algorithms have been widely used to solve the problem of image segmentation (e.g. K-means [Tou and Gonzalez

1974], FCM [Trivedi and Bezdek 1986], ISODATA [Tou and Gonzalez 1974] and snob [Wallace and Dowe 1994]). However, it should be noted that the number of clusters is usually not known *a priori* in image segmentation. Therefore, clustering algorithms that do not require the user to specify the number of clusters are usually preferred.

In this thesis, the clustering problem and the image segmentation problem are considered to be similar. Thus, algorithms are proposed for both problems interchangeably. In the following, several representative clustering-based techniques are presented.

A hybrid approach combining agglomerative hierarchical clustering and region-based segmentation was proposed by Amadasun and King [1988]. The image is first divided into regions. Homogenous regions are specified and mean feature vectors are then determined for each homogenous region. The most similar mean feature vectors are merged. This process is repeated until the specified number of clusters is reached. One advantage of this approach is that it is computationally efficient, because hierarchical clustering is applied on the mean feature vectors instead of the image pixels. However, this approach has several drawbacks, namely [Turi 2001],

- it requires the user to specify the number of clusters in advance,
- it depends on the region size, and
- it depends on the used homogeneity criterion.

Clustering algorithms are usually applied to feature space, and as such they do not use any spatial information (e.g. the relative location of the patterns in the feature space). However, for image segmentation spatial information is important because pixels with similar features are usually found near each other in the spatial domain [Liew *et al.* 2000]. To address this issue, a generalization of K-means that is adaptive and includes spatial information was proposed by Pappas [1992]. In this approach, *a posteriori* probability function is defined which constrains the region intensity and imposes spatial continuity [Turi 2001]. The iterative algorithm alternates between maximizing the *a posteriori* probability function and calculating the cluster centroids. The cluster centroids are initially equal to the K-means cluster centroids. The centroids are updated by averaging them over a sliding window. The size of the sliding window is progressively decreases [Lucchese and Mitra 2001]. Chang *et al.* [1994] extends this algorithm to color image segmentation. Saber *et al.* [1996] extends the approach of Chang *et al.* by proposing a hybrid approach combining color image segmentation and edge linking. Chen *et al.* [1998] applied an approach similar to Pappas [1992] to biomedical images. A drawback of the generalization of K-means approaches is that they require the user to specify the number of clusters in advance [Turi 2001].

A color map image segmentation algorithm combining FCM and a supervised neural network was proposed by Wu *et al.* [1994]. FCM is first applied giving a set of prototypes satisfying some validation criteria. A neural network with supervised learning is then used to optimize these prototypes. The optimized prototypes are used to segment the image using the nearest neighbor rule [Turi 2001].

A fuzzy image clustering algorithm which incorporates spatial contextual information was proposed by Liew *et al.* [2000]. A dissimilarity measure which considers the eight neighboring pixels of each pixel was proposed. The dissimilarity measure is adaptive in the sense that the effect of the neighboring pixels is suppressed in nonhomogenous image regions. In addition, a merging process that merges clusters based on their closeness and their degree of overlap is also used to determine the "optimal" number of clusters. According to Liew *et al.* [2000], due to the incorporation of spatial information, this approach is faster, less sensitive to noise and more suitable for arbitrary shaped clusters than FCM.

Lim and Lee [1990] proposed a two-stage process called *thresholding and FCM*. In the first stage, a *coarse* segmentation is obtained by smoothing the histogram of each color component by a Gaussian convolution. Thresholds are set as the valleys of the smoothed histograms (the valleys are obtained using the first and second derivative of the smoothed histograms). A safe area around each threshold is determined. Each pixel outside these safe areas is assigned to a cluster according to its red, green and blue values. Cluster centroids are then calculated. In the second stage, a *fine* segmentation is obtained by assigning pixels in safe areas to their closest clusters as determined from the fuzzy membership functions. One advantage of this approach is that it dynamically determines the number of clusters. However, the number of clusters obtained is significantly affected by the smoothing function parameter and the size of the safe area [Turi 2001].

Color image segmentation using competitive learning based on the least-squares criterion was proposed by Uchiyama and Arbib [1994]. An image segmentation approach based on the mean shift algorithm was proposed by Comaniciu and Meer [1997]. Shi and Malik [1997] addressed image segmentation using clustering as a graph partitioning problem.

Zhang *et al.* [2001] proposed a hybrid approach combining hidden Markov random field (HMRF) and the EM algorithm to segment brain magnetic resonance (MR) images. A HMRF model is a stochastic process generated by a MRF. The HMRF state sequence can be observed through a field of observations [Zhang *et al.* 2001]. An advantage of HMRF is that it encodes spatial information, which is very useful in image segmentation since it reduces the sensitivity to the presence of noise. A parameter estimation method is required to approximate the parameters of the HMRF model. In this approach, the EM algorithm is used to estimate the parameters. One drawback of this approach is that it depends on the initial estimations of the parameters.

Recently, Veenman *et al.* [2003] proposed a cellular coevolutionary algorithm for image segmentation. The algorithm places agents in a two-dimensional grid representing an image. The agents move pixels between each other to improve the homogeneity of the regions. Neighboring agents form alliances if the union of their regions is homogenous. This approach does not require the user to specify the number of clusters in advance. However, it requires the user to specify a parameter (discussed in section 3.1.4, equation (3.22)) that has a profound effect on the performance of the algorithm.

3.3 Color Image Quantization

Color image quantization is the process of reducing the number of colors presented in a digital color image [Braquelaire and Brun 1997]. Color image quantization can be formally defined as follows [Velho *et al.* 1997]:

Given a set of $N_{S'}$ colors $\boldsymbol{S}' \subset \Re^{N_d}$. The color quantization is a map $f_q : \boldsymbol{S}' \rightarrow \boldsymbol{S}''$ where $\boldsymbol{S}''$ is a set of $N_{S''}$ colors such that $\boldsymbol{S}'' \subset \boldsymbol{S}'$ and $N_{S''} < N_{S'}$. The objective is to minimize the quantization error resulting from replacing a color $\boldsymbol{c} \in \boldsymbol{S}'$ with its quantized value $f_q(\boldsymbol{c}) \in \boldsymbol{S}''$.

Color image quantization is an important problem in the fields of image processing and computer graphics [Velho *et al*. 1997]:

- It can be used in lossy compression techniques [Velho *et al*. 1997];
- It is suitable for mobile and hand-held devices where memory is usually small [Rui *et al*. 2002];
- It is suitable for low-cost color display and printing devices where only small number of colors can be displayed or printed simultaneously [Scheunders, A Genetic 1997].
- Most graphics hardware use color lookup tables with a limited number of colors [Freisleben and Schrader 1997].

Color image quantization consists of two major steps:

- Creating a colormap (or palette) where a small set of colors (typically 8-256 [Scheunders, A Genetic 1997]) is chosen from the (2^{24}) possible combinations of red, green and blue (RGB).
- Mapping each color pixel in the color image to one of the colors in the colormap.

Therefore, the main objective of color image quantization is to map the set of colors in the original color image to a much smaller set of colors in the quantized image [Xiang and Joy 1994]. Furthermore, this mapping, as already mentioned, should minimize the difference between the original and the quantized images [Freisleben and Schrader 1997]. The color quantization problem is known to be NP-complete [Wu and Zhang 1991]. This means that it is not feasible to find the global optimal solution because this will require a prohibitive amount of time. To address this problem, several approximation techniques have been used. One popular approximation method is the use of a standard local search strategy such as K-means. K-means has already been applied to the color image quantization problem [Shafer and Kanade 1987; Celenk 1990]. However, as previously mentioned, K-means is a greedy algorithm which depends on the initial conditions, which may cause the algorithm to converge to suboptimal solutions. This drawback is magnified by the fact that the distribution of local optima is expected to be broad in the color image quantization problem due to the three dimensional color space. In addition, this local optimality is expected to affect the visual image quality. The local optimality issue can be addressed by using stochastic optimization schemes.

Several heuristic techniques have been proposed in the literature. These techniques can be categorized into two main categories: pre-clustering and post-clustering. The next subsections discuss each of these categories.

3.3.1 Pre-clustering approaches

Pre-clustering approaches divide the color into disjoint regions of similar colors. A representative color is then determined from each region. These representatives form the colormap. There are many fast algorithms in this category which are commonly used.

The median cut algorithm (MCA) [Heckbert 1982] is often used in image applications because of its simplicity [Freisleben and Schrader 1997]. MCA divides the color space repeatedly along the median into rectangular boxes until the desired number of colors is obtained.

The variance-based algorithm (VBA) [Wan 1990] also divides the color space into rectangular boxes. However, in VBA the box with the largest mean squared error between the colors in the box and their mean is split.

The octree quantization algorithm [Gervautz and Purgathofer 1990] repeatedly subdivides a cube into eight smaller cubes in a tree structure of degree eight. Then adjacent cubes with the least number of pixels are merged. This is repeated until the required number of colors is obtained [Dekker 1994]. Octree produces results similar to MCA, but with higher speed and smaller memory requirements [Freisleben and Schrader 1997].

Xiang and Joy [1994] proposed an agglomerative clustering method which starts with each image color as a separate cluster. Small clusters are then repeatedly clustered into larger clusters in a hierarchical way until the required number of colors is obtained. The abandoning of the fixed hierarchical division of the color space is a significant improvement over the octree approach [Xiang and Joy 1994].

A similar approach called *Color Image Quantization by Pairwise Clustering* was proposed by Velho *et al*. [1997]. In this approach, a relatively large set of colors is chosen. An image histogram is then created. Two clusters that minimize the quantization error are then selected and merged together. This process is repeated until the required number of colors is obtained. According to Velho *et al*. [1997], this approach performed better than MCA, VBA, octree, K-means and other popular quantization algorithms when applied to the two colored images used in their experiments.

Xiang [1997] proposed a color image quantization algorithm that minimizes the maximum distance between color pixels in each cluster (i.e. the intra-cluster distance). The algorithm starts by assigning all the pixels into one cluster. A pixel is then randomly chosen as the *head* of the cluster. A pixel that is the most distant from its cluster head is chosen as the head of a new cluster. Then, pixels nearer to the head of the new cluster move towards the new head forming the new cluster. This procedure is repeated until the desired number of clusters is obtained. The set of cluster heads forms the colormap.

A hybrid competitive learning (HCL) approach combining competitive learning and splitting of the color space was proposed by Scheunders [A Comparison 1997]. HCL starts by randomly choosing a pixel as a cluster centroid. Competitive learning is then applied resulting in assigning all the image pixels to one cluster surrounding the centroid. A splitting process is then conducted by creating another copy of the centroid; competitive learning is then applied on both centroids. This process is repeated until the desired number of clusters is obtained. According to Scheunders [A Comparison 1997], HCL is fast, completely independent of initial conditions and can obtain near global optimal results. When applied to commonly used images, HCL outperformed MCA, VBA and K-means, and performed comparably with competitive learning [Scheunders, A Comparison 1997; Scheunders, A Genetic 1997].

Braquelaire and Brun [1997] compared the various pre-clustering heuristics and suggested some optimizations of the algorithms and data structures used. Furthermore, they proposed a new color space called H_1 H_2 H_3 and argued that it improves the quantization heuristics. Finally, they proposed a new method which divides each cluster along the axis H_1, H_2 or H_3 of greatest variance. According to Braquelaire and Brun [1997], the proposed approach generates images with comparable quality to that obtained from better but slower methods in this category.

Recently, Cheng and Yang [2001] proposed a color image quantization algorithm based on color space dimensionality reduction. The algorithm repeatedly sub-divides the color histogram into smaller classes. The colors of each class are projected into a line. This line is defined by the mean color vector and the most distant color from the mean color. For each class, the vector generated from the projection of the colors into the line is then used to cluster the colors into two representative palette colors. This process is repeated until the desired number of representative colors is obtained. All color vectors in each class are then represented by their class mean. Finally, all these representative colors form the colormap. According to Cheng and Yang [2001], this algorithm performed better than MCA, and performed comparably to SOM when applied on commonly used images.

3.3.2 Post-clustering approaches

The main disadvantage of the pre-clustering approaches is that they only work with color spaces of simple geometric characteristics. On the other hand, post-clustering approaches can work with arbitrary shaped clusters. Post-clustering approaches perform clustering of the color space [Cheng and Yang 2001]. A post-clustering algorithm starts with an initial colormap. It then iteratively modifies the colormap to improve the approximation. The major disadvantage of post-clustering algorithms is the fact that it is time consuming [Freisleben and Schrader 1997].

The K-means algorithm is one of the most popular post-clustering algorithms. It starts with an initial set of colors (i.e. initial colormap). Then, each color pixel is assigned to the closest color in the colormap. The colors in the colormap are then recomputed as the centroids of the resulting clusters. This process is repeated until convergence. The K-means algorithm has been proven to converge to a local optimum

[Freisleben and Schrader 1997]. As previously mentioned, a major disadvantage of K-means is its dependency on initial conditions.

FCM [Balasubramanian and J. Allebach 1990] and Learning Vector Quantization [Kotropoulos *et al.* 1992] have also been used in the color image quantization. Scheunders and De Backer [1997] proposed a joint approach using both competitive learning and a dithering process to overcome the problem of contouring effects when using small colormaps.

Fiume and Quellette [1989] proposed an approach which uses simulated annealing for color image segmentation. Pre-clustering approaches were used to initialize the colormap.

SOMs (discussed in Section 3.1.6) were used by Dekker [1994] to quantize color images. The approach selects an initial colormap, and then modifies the colors in the colormap by moving them in the direction of the image color pixels. However, to reduce the execution time, only samples of the colors in the image are used. According to Dekker [1994], the algorithm performs better than MCA and octree.

Rui *et al*. [2002] presented an initialization and training method for SOM that reduces the computational load of SOM and at the same time generates reasonably good results.

A hybrid approach combining evolutionary algorithms with K-means has been proposed by Freisleben and Schrader [1997]. A population of individuals, each representing a colormap, are arbitrary initialized. Then, after each generation, the K-means algorithm (using a few iterations) is applied on each individual in the population. The standard error function of the Euclidean distance is chosen to be the fitness function of each individual. Based on the experiments conducted by Freisleben and Schrader [1997], this hybrid approach outperformed both MCA and octree algorithms.

Genetic C-means algorithm (GCMA) uses a similar idea where a hybrid approach combining a genetic algorithm with K-means was proposed by Scheunders [A Genetic 1997]. The fitness function of each individual in the population is set to be the mean square error (MSE), defined as

$$MSE = \frac{\sum_{k=1}^{K} \sum_{\forall z_p \in C_k} (\boldsymbol{z}_p - \boldsymbol{m}_k)^2}{N_p} \tag{3.27}$$

As in Freisleben and Schrader [1997], each chromosome represents a colormap. GCMA starts with a population of arbitrary initialized chromosomes. K-means is then applied to all the chromosomes to reduce the search space. A single-point crossover is then applied. This is followed by the application of mutation which randomly decides if a value of one is added to (or subtracted from) the gene's value (i.e. mutating the gene's value with ±1). All the chromosomes are then pairwise compared and the chromosome with the lowest MSE replaces the other chromosome. This process is repeated until a stopping criterion is satisfied. A faster version of this approach can be obtained by applying K-means to the best chromosome in each generation. For the remaining chromosomes, an approximation of K-means is used where a single iteration of K-means is applied on a randomly chosen subset of pixels. This process is repeated a user-specified number of times using different subsets. GCMA outperformed MCA, VBA, K-means, competitive learning and HCL when applied on commonly used images [Scheunders, A Comparison 1997; Scheunders, A Genetic 1997]. However, GCMA is computationally expensive.

Recently, a new approach using model based clustering trees was proposed by Murtagh *et al*. [2001]. The algorithm requires selecting a 3D color space (e.g. RGB) and specifying the order of color bands. For the first color band, the number of clusters is determined using BIC (discussed in section 3.1.5, equation (3.23)). The EM algorithm is used to estimate the model parameters and each pixel is then assigned to its most likely cluster. The second color band is then used to split each of the clusters generated from the previous step. The generated clusters are further subdivided using the third color band.

3.4 Spectral Unmixing

In remote sensing, classification is the main tool for extracting information about the surface cover type. Conventional classification methods assign each pixel to one class (or species). This class can represent water, vegetation, soil, etc. The classification methods generate a map showing the species with highest concentration. This map is known as the *thematic map*. A thematic map is useful when the pixels in the image represent pure species (i.e. each pixel represents the spectral signature of one species). Hence, thematic maps are suitable for imagery data with a small ground sampling distance (GSD) such as LANDSAT Thematic Mapper (GSD = 30 m). However, thematic maps are not as useful for large GSD imagery such as NOAA'a AVHRR (GSD = 1.1 km) because in this type of imagery pixels are usually not pure. Therefore, pixels need to be assigned to several classes along with their respective concentrations in that pixel's footprint. Spectral unmixing (or *mixture modeling*) is used to assign these classes and concentrations. Spectral unmixing generates a set of maps showing the proportions of all species present in each pixel footprint. These maps are called the *abundance images*. Hence, each abundance image shows the concentration of one species in a scene. Therefore, spectral unmixing provides a more complete and accurate classification than a thematic map generated by conventional classification methods.

Spectral unmixing can be used for the compression of multispectral imagery. Using spectral unmixing, the user can prioritize the species of interest in the compression process. This is done by first applying the spectral unmixing on the original images to generate the abundance images. The abundance images representing the species of interest are then prioritized by coding them with a relatively high bit rate. Other abundance images are coded using a relatively low bit rate. At the decoder, the species-prioritized reconstructed multispectral imagery is generated via a re-mixing process on the decoded abundance images [Saghri *et al.* 2002]. This approach is feasible if the spectral unmixing algorithm results in a small (negligible) residual error.

3.4.1 Linear Pixel Unmixing (or Linear Mixture Modeling)

Spectral unmixing is generally performed using a linear mixture modeling approach. In linear mixture modeling the spectral signature of each pixel vector is assumed to be a linear combination of a limited set of fundamental spectral components known as *end-members*. Hence, spectral unmixing can be formally defined as follows:

$$\boldsymbol{z}_p = \boldsymbol{EM}.\boldsymbol{f} + \boldsymbol{e} = f_1\boldsymbol{em}_1 + f_2\boldsymbol{em}_2 + \cdots + f_i\boldsymbol{em}_i + \cdots + f_{N_e}\boldsymbol{em}_{N_e} + \boldsymbol{e} \tag{3.28}$$

where the symbols are defined as follows:

$\boldsymbol{z}_p$	a pixel signature of N_b components
$\boldsymbol{EM}$	$N_b \times N_e$ matrix of end-members $\boldsymbol{em}_{1,\cdots,N_e}$
f_i	fractional component of end-member i (i.e. proportion of footprint covered by species i)
$\boldsymbol{f}$	vector of fractional components $(f_1, f_2, \cdots, f_i, \cdots, f_{N_e})^{\mathrm{T}}$
$\boldsymbol{em}_i$	end-member i of N_b components
$\boldsymbol{e}$	residual error vector of N_b components
N_b	number of spectral bands
N_e	number of components, $N_e \le N_b$

Provided that the number of end-members is less than or equal to the true spectral dimensionality of the scene, the solution via classical least-squares estimation is,

$$\boldsymbol{f} = (\boldsymbol{EM}^{\mathrm{T}}\boldsymbol{EM})^{-1}\boldsymbol{EM}^{\mathrm{T}}\boldsymbol{z}_p \tag{3.29}$$

Therefore, there are two requirements for linear spectral unmixing:

- the spectral signature of the end-members needs to be known, and
- the number of end-members has to be less than or equal to the true spectral dimensionality of the scene (i.e. the dimension of the feature space). This is known as the *condition of identifiability*.

The condition of identifiability restricts the application of the linear spectral unmixing when applied to multispectral imagery, because

- the end-members may not correspond to physically identifiable species on the ground, and
- the number of distinct species in the scene may be more than the true spectral dimensionality of the scene. For example, for Landsat TM with seven spectral bands (N_b =7), the true spectral dimension is at most five (N_e=5) based on principal component analysis.

3.4.2 Selection of the End-Members

There are many methods for end-member selection proposed in the literature [Settle and Drake 1993; Antoniades *et al*. 1995; Hlavka and Spanner 1995; Bateson and Curtiss 1996; Maselli 1998; Parra *et al*. 2000; Saghri *et al*. 2000]. In the following, several representative techniques are presented.

Mathematical techniques such as Gram-Schmidt orthogonalization and principal component analysis can be used to obtain orthogonal end-members which can be used to linearly unmix each pixel vector of the scene. There are several advantages for the mathematical techniques, namely

- they result in minimum residual error, and
- there is no human interaction time.

However, mathematical techniques suffer from the following drawbacks:

- They may generate end-members with negative components.
- They may not correspond to physical species in the scene.

Manual techniques can also be used to obtain end-members which can be used to linearly unmix each pixel vector of the scene. In manual techniques, the user will select the end-members directly from the scene, or from a library of end-members. The advantages of manual techniques are the disadvantages of mathematical techniques and *vice versa*.

Spectral screening is another way to obtain end-members. In this approach, a set of unique pixels are selected from the scene. The selection is based on a user-specified spectral angle threshold. The approach works as follows:

- The first pixel in the image is assumed to be unique and is added to the set of unique pixels.
- The pixels in the image are then sequentially scanned and each pixel whose spectral angle with respect to all the unique pixels in the set exceeds the user-specified spectral angle threshold, is added to the set of unique pixels.

Clearly, this technique suffers from two major drawbacks:

- the generated set of unique pixels depends on the order in which the pixels are scanned, and
- the generated set also depends on the spectral angle threshold.

To overcome the condition of identifiability, Maselli [1998] proposed a method of dynamic selection of an optimum end-member subset. In this technique, an optimum subset of all available end-members is selected for spectral unmixing of each pixel vector in the scene. Thus, although every pixel vector will not have a fractional component for each end-member, the ensemble of all pixel vectors in the scene will collectively have fractional contributions for each end-member.

For each pixel vector, a unique subset of the available end-members is selected which minimizes the residual error after decomposition of that pixel vector. To determine the N_e optimum end-members for pixel vector $\boldsymbol{z}_p$, the pixel vector is projected onto all available normalized end-members. The most efficient projection, which corresponds to the highest dot product value $c_{\max}$, indicates the first selected end-member $\boldsymbol{em}_{\max}$. It can be shown that this procedure is equivalent to finding the end-member with the smallest spectral angle with respect to $\boldsymbol{z}_p$ [Saghri *et al.* 2000]. The residual pixel signature, $\boldsymbol{r}_{z_p} = \boldsymbol{z}_p - c_{\max}.\boldsymbol{em}_{\max}$ is then used to identify the second end-member by repeating the projection onto all remaining end-members. The process continues up to the identification of a prefixed maximum N_e number of end-members from the total of N_m available end-members.

More recently, Saghri *et al.* [2000] proposed a method to obtain end-members from the scene with relatively small residual errors. In this method, the set of end-members are chosen from a thematic map resulting from a modified ISODATA. The modified ISODATA uses the spectral angle measure instead of the Euclidean distance measure to reduce the effect of shadows and sun angle effects. The end-members are then set as the centroids of the compact and well-populated clusters. Maselli's approach discussed above is then used to find the optimum end-member subset from the set of available end-members for each pixel in the scene. Linear spectral unmixing is then applied to generate the abundance images.

According to Saghri *et al.* [2000], the proposed approach has several advantages:

- the resulting end-members correspond to physically identifiable (and likely pure) species on the ground,

- the residual error is relatively small, and
- minimal human interaction time is required.

However, this approach has the drawback that it uses ISODATA which depends on initial conditions.

3.5 Conclusions

This chapter presented an overview of a set of problems from the field of pattern recognition and image processing. The clustering problem was defined and discussed, followed by image segmentation and color image quantization. Finally, spectral unmixing was overviewed. From the discussion presented in this chapter it can be observed that all these problems are difficult to solve and they need efficient optimization methods to solve them. In this thesis, the PSO is used to address these difficult problems. In the next chapter, a PSO-based clustering algorithm is proposed and compared with other *state-of-the-art* clustering algorithms.

Chapter 4

A PSO-based Clustering Algorithm with Application to Unsupervised Image Classification

A clustering method that is based on PSO is developed in this chapter. The algorithm finds the centroids of a user specified number of clusters, where each cluster groups together similar patterns. The application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images is investigated. To illustrate its wide applicability, the proposed algorithm is then applied to synthetic, MRI and satellite images. Experimental results show that the PSO clustering algorithm performs better than *state-of-the-art* clustering algorithms (namely, K-means, Fuzzy C-means, K-Harmonic means and Genetic Algorithms) in all measured criteria. The influence of different values of PSO control parameters on performance is illustrated. The performance of different versions of PSO is also investigated.

4.1 PSO-Based Clustering Algorithm

This section defines the terminology used throughout the rest of the chapter. A measure is given to quantify the quality of a clustering algorithm, after which the PSO-based clustering algorithm is introduced.

4.1.1 Measure of Quality

Different measures can be used to express the quality of a clustering algorithm. The most general measure of performance is the quantization error, defined as

$$J_e = \frac{\sum_{k=1}^{K}\left[\sum_{\forall \boldsymbol{z}_p \in \boldsymbol{C}_k} d(\boldsymbol{z}_p, \boldsymbol{m}_k)\right] \Big/ n_k}{K} \tag{4.1}$$

where $\boldsymbol{C}_k$ is the k^{th} cluster, and n_k is the number of pixels in $\boldsymbol{C}_k$

4.1.2 PSO-Based Clustering Algorithm

In the context of data clustering, a single particle represents the K cluster centroids. That is, each particle $\boldsymbol{x}_i$ is constructed as $\boldsymbol{x}_i = (\boldsymbol{m}_{i,1},\ldots,\boldsymbol{m}_{i,k},\ldots,\boldsymbol{m}_{i,K})$ where $\boldsymbol{m}_{i,k}$ refers to the k^{th} cluster centroid vector of the i^{th} particle. Therefore, a swarm represents a number of candidate data clusterings. The quality of each particle is measured using

$$f(\boldsymbol{x}_i, \boldsymbol{Z}_i) = w_1 \bar{d}_{\max}(\boldsymbol{Z}_i, \boldsymbol{x}_i) + w_2(z_{\max} - d_{\min}(\boldsymbol{x}_i)) \tag{4.2}$$

where $z_{\max}$ is the maximum value in the data set (i.e. in the context of digital images, $z_{\max} = 2^s - 1$ for an s-bit image); $\boldsymbol{Z}_i$ is a matrix representing the assignment of patterns to the clusters of particle i. Each element $z_{i,k,p}$ indicates if pattern $\boldsymbol{z}_p$ belongs to cluster $\boldsymbol{C}_k$ of particle i. The constants w_1 and w_2 are user-defined constants used to weigh the contribution of each of the sub-objectives. Also,

$$\bar{d}_{max}(\boldsymbol{Z}_i, \boldsymbol{x}_i) = \max_{k=1,\ldots,K}\left\{\sum_{\forall \boldsymbol{z}_p \in \boldsymbol{C}_{i,k}} d(\boldsymbol{z}_p, \boldsymbol{m}_{i,k}) / n_{i,k}\right\} \tag{4.3}$$

is the maximum average Euclidean distance of particles to their associated clusters, and

$$d_{min}(\boldsymbol{x}_i) = \min_{\forall k,kk,k \neq kk} \{d(\boldsymbol{m}_{i,k}, \boldsymbol{m}_{i,kk})\} \tag{4.4}$$

is the minimum Euclidean distance between any pair of clusters. In the above, $n_{i,k}$ is the number of patterns that belong to cluster $\boldsymbol{C}_{i,k}$ of particle *i*.

The fitness function in equation (4.2) has as objective to simultaneously minimize the intra-distance between patterns and their cluster centroids, as quantified by $\bar{d}_{\max}(\boldsymbol{Z}_i, \boldsymbol{x}_i)$, and to maximize the inter-distance between any pair of clusters, as quantified by, $d_{\min}(\boldsymbol{x}_i)$.

According to the definition of the fitness function, a small value of $f(\boldsymbol{x}_i, \boldsymbol{Z}_i)$ suggests compact and well-separated clusters (i.e. *good* clustering).

The fitness function is thus a multi-objective problem. Approaches to solve multi-objective problems have been developed mostly for evolutionary computation approaches [Coello Coello 1996]. Recently, approaches to multi-objective optimization using PSO have been developed by Hu and Eberhart [Multiobjective 2002], Fieldsend and Singh [2002] and Coello Coello and Lechuga [2002]. Since our scope is to illustrate the applicability of PSO to data clustering, and not on multi-objective optimization, a simple weighted approach is used to cope with multiple objectives. Different priorities are assigned to the subobjectives via appropriate initialization of the values of w_1 and w_2.

The PSO clustering algorithm is summarized in Figure 4.1.

```
1. Initialize each particle to contain $K$ randomly selected cluster centroids
2. For $t = 1$ to $t_{\max}$
        (a) For each particle $i$
                i. For each pattern $\boldsymbol{z}_p$
                        • calculate $d(\boldsymbol{z}_p, \boldsymbol{m}_{i,k})$ for all clusters $\boldsymbol{C}_{i,k}$ using equation (3.1)
                        • assign $\boldsymbol{z}_p$ to $\boldsymbol{C}_{i,k}$ where
                          $d(\boldsymbol{z}_p, \boldsymbol{m}_{i,k}) = \min_{\forall k=1,\ldots,K} \{d(\boldsymbol{z}_p, \boldsymbol{m}_{i,k})\}$        (4.5)
                ii. Calculate the fitness, $f(\boldsymbol{x}_i, \boldsymbol{Z}_i)$
        (b) Find the personal best position for each particle and the global best solution, $\hat{\boldsymbol{y}}(t)$
(c) Update the cluster centroids using equations (2.8) and (2.10)
```

Figure 4.1: The PSO clustering algorithm

As previously mentioned, an advantage of using PSO is that a parallel search for an optimal clustering is performed. This population-based search approach reduces the effect of the initial conditions, compared to K-means (as shown in Figure 4.4), especially for relatively large swarm sizes.

4.1.3 A Fast Implementation

Since most of the images used in this thesis are single band, gray scale images and since most clustering algorithms do not use spatial information, a fast implementation is used for this type of images in order to speedup the execution time of the algorithms used. The fast implementation works as follows:

1) The histogram of a single band, gray scale image is created by calculating the frequency of each gray level.
2) A data structure is used where each gray level is associated with a frequency value and a cluster label.
3) Depending on the algorithm used, perform all the calculations (e.g. Euclidean distance, calculation of centroids, fitness function, etc.) using the above data structure by multiplying each gray level by its frequency and using the cluster labels for clustering.

Using the above implementation, the execution time will be independent on the size of the image. However, the execution time will depend on the number of gray levels which is usually very small (e.g. 256 for 8-bit images and 1024 for 10-bit images). Furthermore, the number of gray levels is generally much less than the number of pixels. Hence, the execution time will reduce significantly while the results are the same. Therefore, this implementation is used in this thesis for single band, gray scale images.

4.2 Experimental Results

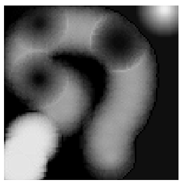
The PSO-based clustering algorithm has been applied to three types of imagery data, namely synthetic, MRI and LANDSAT 5 MSS (79 m GSD) images. These data sets have been selected to test the algorithms, and to compare them with other algorithms, on a range of problem types, as listed below:

Synthetic Image: Figure 4.2(a) shows a 100 × 100 8-bit gray scale image created to specifically show that the PSO algorithm does not get trapped in the local minimum. The image was created using two types of brushes, one brighter than the other.

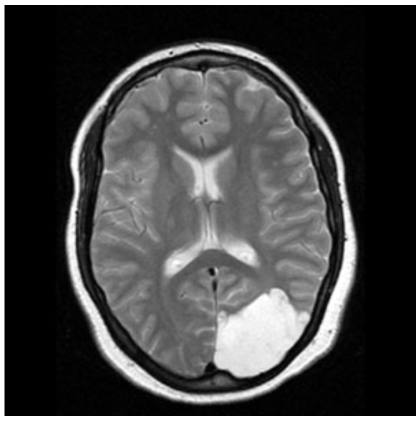
MRI Image: Figure 4.2(b) shows a 300 × 300 8-bit gray scale image of a human brain, intentionally chosen for its importance in medical image processing.

Remotely Sensed Imagery Data: Figure 4.2(c) shows band 4 of the four-channel multispectral test image set of the Lake Tahoe region in the US. Each channel is comprised of a 300 × 300, 8-bit per pixel (remapped from the original 6 bit) image. The test data are one of the North American Landscape Characterization (NALC) Landsat multispectral scanner data sets obtained from the U.S. Geological Survey (USGS).

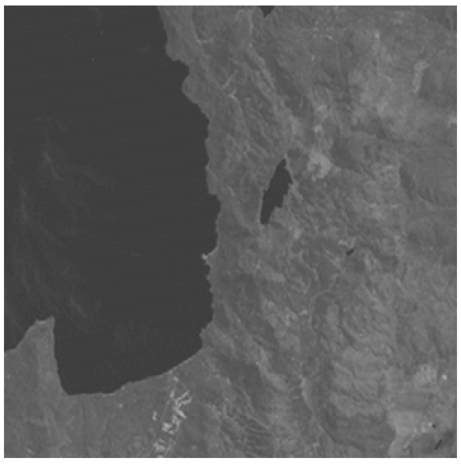
The rest of this section is organized as follows: Section 4.2.1 illustrates that the basic PSO can be used successfully as an unsupervised image classifier, using the original fitness function as defined in equation (4.2). Section 4.2.2 illustrates the performance under a new fitness function. Results of the *gbest* PSO are compared with that of GCPSO in section 4.2.3, using the new fitness function. Section 4.2.4 investigates the influence of the different PSO control parameters. The performance of PSO using the new fitness function is compared with *state-of-the-art* clustering approaches in Section 4.2.5. In section 4.2.6, the performance of different versions of PSO is investigated. A new non-parametric fitness function is presented in Section 4.2.7. In section 4.2.8, the PSO-based clustering algorithm is applied to multispectral imagery data. Finally, section 4.2.9 provides a discussion of applying PSO to data clustering.



(a) Synthetic image



(b) MRI Image of Human brain



(c) Band 4 of the Landsat MSS test image of Lake Tahoe

Figure 4.2: Data set consisting of synthetic, MRI and LANDSAT images

The results reported in this section are averages and standard deviations over 20 simulations. All comparisons are made with reference to J_e, $\bar{d}_{\max}$ and $d_{\min}$.

Furthermore, a total number of clusters of 3, 8 and 4 were used respectively for the synthetic, MRI and Tahoe images.

4.2.1 *gbest* PSO versus K-Means

This section presents results to compare the performance of the *gbest* PSO algorithm with that of the K-means algorithm for each of the images.

Table 4.1 summarizes the results for the three images. In all cases, for PSO, 50 particles were trained for 100 iterations; for K-means, 5000 iterations were used (that is, both algorithms have performed 5000 function evaluations). $V_{max} = 5$, $w = 0.72$ and $c_1 = c_2 = 1.49$. The chosen values of w, c_1, and c_2 are popular in the literature and ensure convergence [Van den Bergh 2002]. For the fitness function in equation (4.2), $w_1 = w_2 = 0.5$ to give each subobjective an equal contribution.

The results showed that, for the images used, K-means performed better than the PSO algorithm with reference to the quantization error J_e. However, J_e does not give an idea of the quality of the individual clusters. With respect to the minimization of intra-distances ($\bar{d}_{\max}$) and the maximization of inter-distances ($d_{\min}$), the PSO algorithm generally performed better than K-means clustering.

Figure 4.3 illustrates for the synthetic image how the fitness of PSO improves over time. For this figure, 10 particles have been used for a training phase of 100 iterations, $V_{max} = 5$, $w = 0.72$, $c_1 = c_2 = 1.49$, and $w_1 = w_2 = 0.5$. The fitness value, as measured using equation 4.2, improves from the initial 96.637 to 91.781.

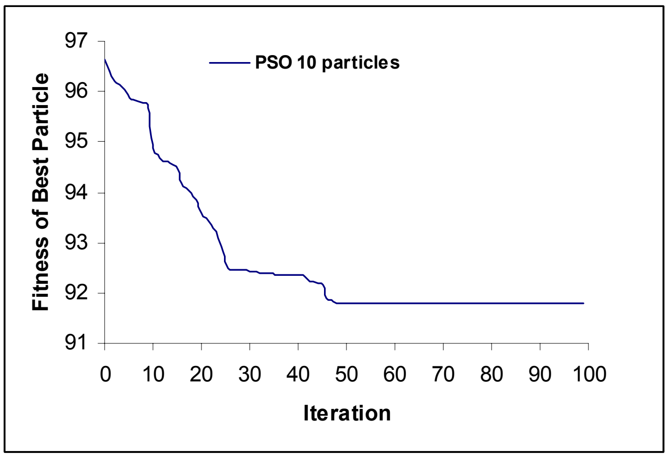


Figure 4.3: PSO Performance on Synthetic Image

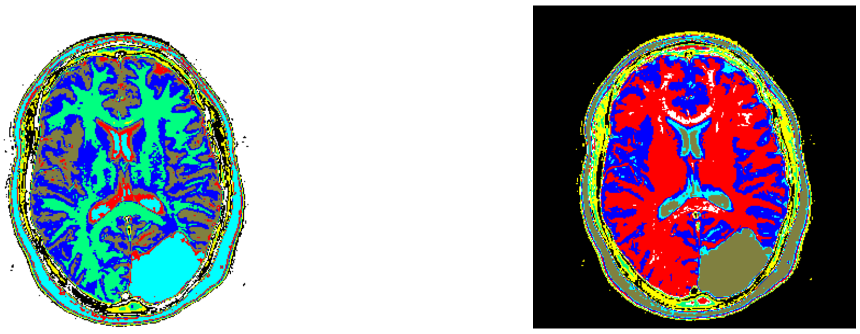
Figure 4.4(a) illustrates the segmented image of the synthetic image for the K-means algorithm, while Figure 4.4(b) illustrates the segmented image obtained from the PSO algorithm. These figures clearly illustrate that K-means was trapped in a local optimum. Three clusters were created using two brushers, the brighter brush were used to create the two spots in the upper right and lower left corner while the other brush were used to create the remaining shape. K-means could not classify the clusters correctly, since it failed to cluster the two spots as separate clusters. PSO, on the other hand, was not trapped in this local optimum and succeeded in showing the two spots as separate clusters. The segmented images for the MRI and the Tahoe images are given in Figures 4.5 and 4.6, respectively.

Table 4.1: Comparison between K-means and PSO

Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	**K-means**	20.212 ± 0.938	28.040 ± 2.778	78.498 ± 7.0629
	PSO	24.453 ± 0.209	27.157 ± 0.017	98.679 ± 0.023
MRI	**K-means**	7.370 ± 0.0428	13.214 ± 0.762	9.934 ± 7.309
	PSO	8.536 ± 0.584	10.129 ± 1.262	28.745 ± 2.949
Tahoe	**K-means**	1.664 ± 0.040	3.107 ± 0.168	4.527 ± 1.347
	PSO	7.215 ± 2.393	9.036 ± 3.363	25.777 ± 9.602

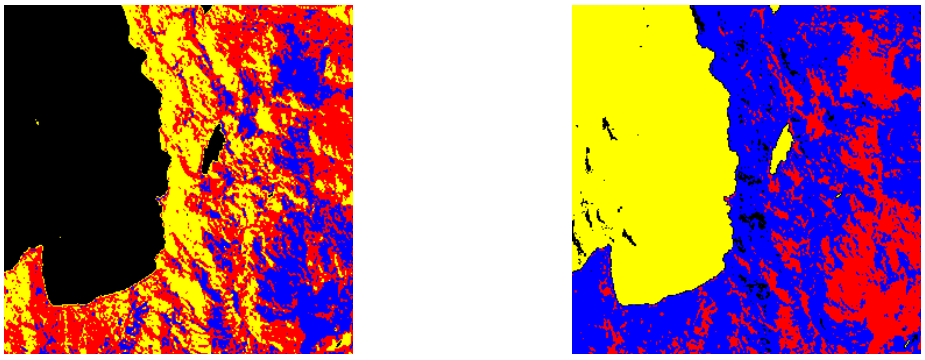
(a) K-means (b) PSO

Figure 4.4: The Segmented Synthetic Images



(a) K-means (b) PSO

Figure 4.5: The Segmented MRI Images



(a) K-means (b) PSO

Figure 4.6: The Segmented Lake Tahoe Images

4.2.2 Improved Fitness Function

The above experimental results have shown that the PSO clustering algorithm improves on the performance of the K-means algorithm in terms of inter- and intra-cluster distances. An improved fitness function which simply adds to the previous fitness function an additional sub-objective to also minimize the quantization error is presented in the following equation:

$$f(\boldsymbol{x}_i, \boldsymbol{Z}_i) = w_1 \bar{d}_{\max}(\boldsymbol{Z}_i, \boldsymbol{x}_i) + w_2(z_{\max} - d_{\min}(\boldsymbol{x}_i)) + w_3 J_{e,i} \tag{4.6}$$

In this section, the results of the *gbest* PSO shown in the previous section are compared with results using the new fitness function as defined in equation (4.6). All parameters are set as in the previous section. The only difference is that for the extended fitness function, $w_1 = w_2 = 0.3$, $w_3 = 0.4$ were used for the synthetic image, $w_1 = 0.2$, $w_2 = 0.5$, $w_3 = 0.3$ were used for the MRI image and $w_1 = w_2 = w_3 = 0.333333$ were used for the Tahoe image. These values were set empirically.

Table 4.2 compares the results for the two fitness functions. The new fitness function succeeded in significant improvements in the quantization error, J_e. The new fitness function also achieved significant improvements in minimizing the intra-cluster distances for the synthetic and Tahoe images, thus resulting in more compact clusters, and only marginally worse for the MRI image. These improvements were at the cost of loosing on maximization of the inter-cluster distances. However, this loss is acceptable because the *gbest* PSO using the new fitness function still performs better than the K-means algorithm in terms of the inter-cluster distance (compare the results in Table 4.1 and Table 4.2).

Due to the improved performance on the quantization error and intra-cluster distances, the rest of this chapter uses the 3-component fitness function as defined in equation (4.6).

Table 4.2: 2-component versus 3-component fitness function

Problem	**2-Component Fitness Function**			**3-Component Fitness Function**		
	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	24.453 ± 0.209	27.157 ± 0.017	98.679 ± 0.023	17.113 ± 0.548	24.781 ± 0.270	92.768 ± 4.043
MRI	8.536 ± 0.584	10.129 ± 1.262	28.745 ± 2.949	7.225 ± 0.552	12.206 ± 2.507	22.936 ± 8.311
Tahoe	7.215 ± 2.393	9.036 ± 3.363	25.777 ± 9.602	3.556 ± 0.140	4.688 ± 0.260	14.987 ± 0.425

4.2.3 *gbest* PSO versus GCPSO

This section compares the performance of the *gbest* PSO with the GCPSO. This is done for a low V_{max} = 5 and a high V_{max} = 255. All other parameters are as for section 4.2.2. Table 4.3 shows no significant difference in the performance between PSO and GCPSO. It is, however, important to note that too much clamping of the velocity updates have generally a negative influence on performance. In general, better results were obtained, for both the PSO and GCPSO with a large value of V_{max}.

Table 4.3: PSO versus GCPSO

Problem	**PSO**			**GCPSO**		
V_{max}=5	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	17.112672 ± 0.548096	24.781384 ± 0.270409	92.767925 ± 4.043086	17.116036 ± 0.547317	24.826868 ± 0.237154	92.845323 ± 4.056681
MRI	7.225384 ± 0.552381	12.205947 ± 2.506827	22.935786 ± 8.310654	7.239264 ± 0.475250	12.438016 ± 2.437064	23.377287 ± 6.722787
Tahoe	3.556281 ± 0.139881	4.688270 ± 0.259919	14.986923 ± 0.425077	3.542732 ± 0.109415	4.672483 ± 0.129913	15.007491 ± 0.621020
V_{max}=255						
Synthetic	17.004993 ± 0.086698	24.615665 ± 0.143658	93.478081± 0.276109	17.000393 ± 0.022893	24.672107 ± 0.174457	93.588530 ± 0.400137
MRI	7.640622 ± 0.514184	10.621452 ± 1.284735	24.948486 ± 3.446673	7.694498 ± 0.591383	10.543233 ± 1.038114	25.355967 ± 3.945929
Tahoe	3.523967 ± 0.172424	4.681492 ± 0.110739	14.664859 ± 1.177861	3.609807 ± 0.188862	4.757948 ± 0.227090	15.282949 ± 1.018218

4.2.4 Influence of PSO Parameters

The PSO have a number of parameters that have an influence on the performance of the algorithm. These parameters include V_{max}, the number of particles, the inertia weight and the acceleration constants. Additionally, the PSO-based clustering algorithm adds a weight to each sub-objective. This section investigates the influence of different values of these parameters.

Velocity Clamping

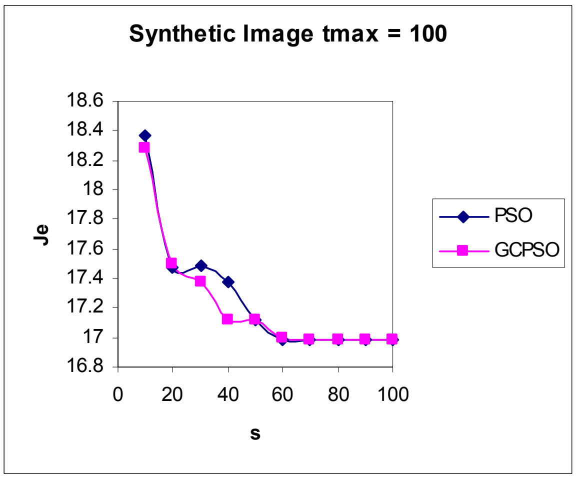
Table 4.3 shows that less clamping of velocity updates is more beneficial. This allows particles to make larger jumps in the search space.

Swarm Size

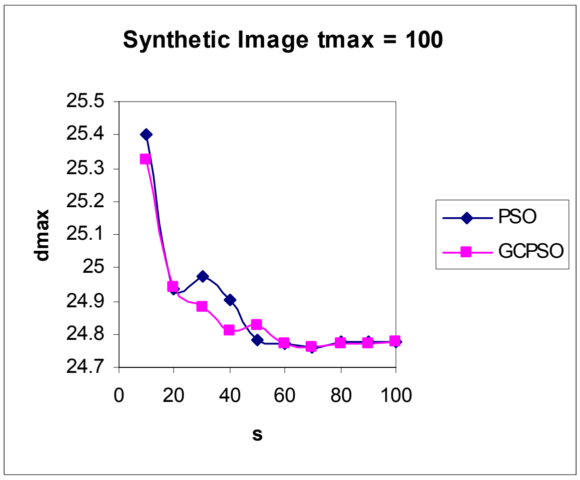
To investigate the effect of different swarm sizes on performance, both the PSO and GCPSO have been executed using 10 to 100 particles. All other parameters are as for section 4.2.2. Figure 4.7 shows the effect of the swarm size, *s*, on the synthetic image. It is clear from the figure that increasing the number of particles improves the performance of both algorithms. The same conclusion can be drawn for the MRI image as illustrated in Figure 4.8. However, it can be observed from Figure 4.7, that no significant improvement is achived for more than 60 particles. In general, an increase in the number of particles increases diversity, thereby limiting the effects of initial conditions and reducing the possibility of being trapped in local minima.

Inertia Weight

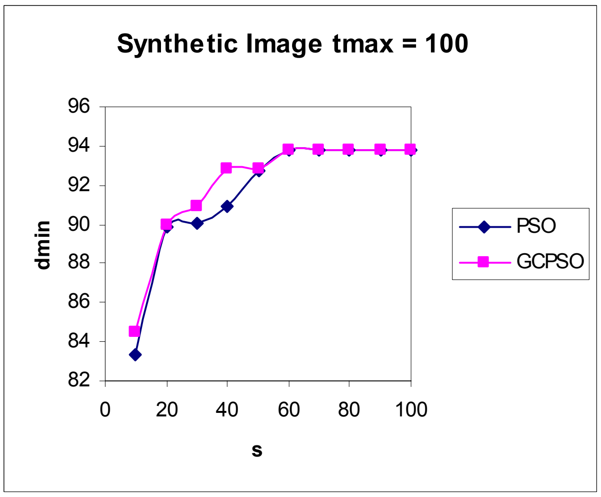
Given that all parameters are fixed at the values given in section 4.2.2, the inertia weight *w* was set to different values for both PSO and GCPSO. In addition, a dynamic inertia weight was used with an initial w =1.4, which linearly decreased to 0.8. The initial large value of *w* favors exploration in the early stages, with more exploitation in the later stages with the smaller values. Tables 4.4 and 4.5 summarize the results for the synthetic and MRI images respectively. For the synthetic image, the results illustrate no significant difference in performance, meaning that for the synthetic image, the PSO-based clustering algorithms are generally insensitive to the value of the inertia weight (provided that c_1 and c_2 are selected such that equation (2.9) is not violated). However, in the MRI image, it can be observed that w =0 yields the best results in terms of inter- and intra-cluster distances.



(a) Quantization error

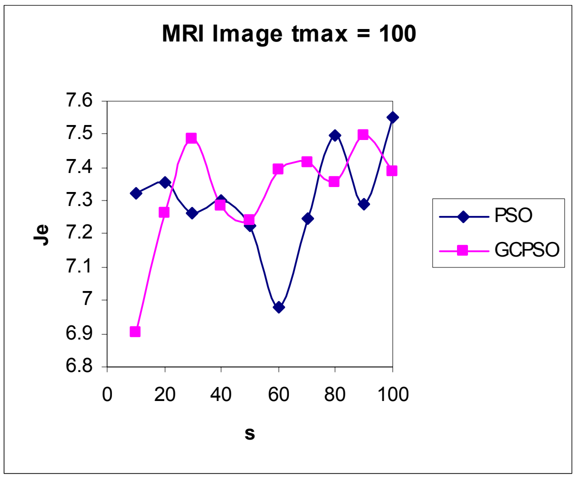


(b) Intra-cluster Distances

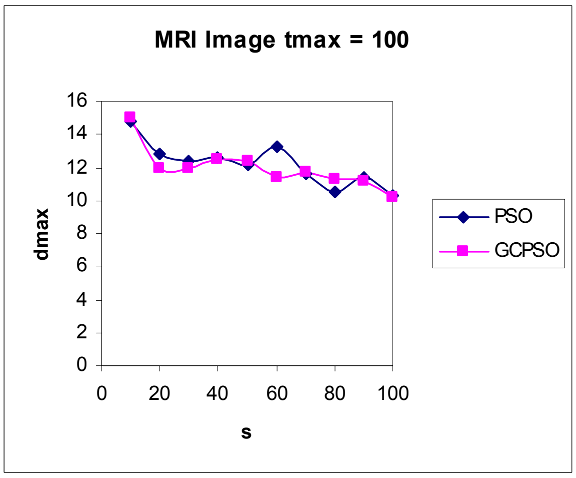


(c) Inter-cluster Distances

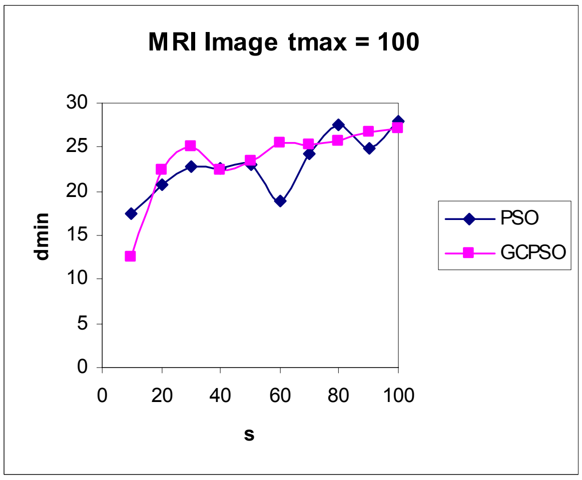
Figure 4.7: Effect of swarm size on synthetic image



(a) Quantization error



(b) Intra-cluster Distances



(c) Inter-cluster Distances

Figure 4.8: Effect of swarm size on MRI image

Table 4.4: Effect of inertia weight on the synthetic image

w	PSO			GCPSO		
	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
0.0	16.983429 ± 0.017011	24.581799 ± 0.165103	93.435221 ± 0.308601	16.986386 ± 0.016265	24.649368 ± 0.138223	93.559275 ± 0.254670
0.1	16.982362 ± 0.016074	24.645884 ± 0.137442	93.543795 ± 0.256700	16.985079 ± 0.016995	24.637893 ± 0.138894	93.538635 ± 0.257167
0.5	16.985826 ± 0.014711	24.664421 ± 0.144252	93.595394 ± 0.246110	16.987470 ± 0.028402	24.662973 ± 0.163768	93.58124 ± 0.281366
0.72	16.992102 ± 0.021756	24.670338 ± 0.150542	93.606400 ± 0.258548	16.995967 ± 0.039686	24.722414 ± 0.144572	93.680765 ± 0.253954
0.9	16.993759 ± 0.014680	24.650337 ± 0.140005	93.569595 ± 0.252781	17.040990 ± 0.168017	24.633802 ± 0.352785	93.495340 ± 0.584424
1.4 to 0.8	17.824495 ± 0.594291	24.433770 ± 1.558219	92.625088 ± 2.031224	17.481146 ± 0.504740	24.684407 ± 1.010815	93.223498 ± 1.490217

Table 4.5: Effect of inertia weight on the MRI image

w	PSO			GCPSO		
	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
0.0	7.538669 ± 0.312044	9.824915 ± 0.696940	28.212823 ± 2.300930	7.497944 ± 0.262656	9.731746 ± 0.608752	28.365827 ± 1.882164
0.1	7.511522 ± 0.281967	10.307791 ± 1.624499	27.150801 ± 3.227550	7.309289 ± 0.452103	10.228958 ± 1.354945	26.362349 ± 3.238452
0.5	7.612079 ± 0.524669	10.515242 ± 1.103493	26.996556 ± 2.161969	7.466388 ± 0.492750	10.348044 ± 1.454050	26.790056 ± 2.830860
0.72	7.57445 ± 0.382172	10.150214 ± 1.123441	27.393498 ± 3.260418	7.467591 ± 0.396310	10.184191 ± 0.955129	26.596493 ± 3.208689
0.9	7.847689 ± 0.529134	10.779765 ± 1.134843	26.268057 ± 3.595596	7.598518 ± 0.516938	10.916945 ± 1.534848	25.417859 ± 3.174232
1.4 to 0.8	8.354957 ± 0.686190	13.593536 ± 2.035889	21.625623 ± 4.507230	8.168068 ± 0.709875	12.722139 ± 1.850957	21.169304 ± 4.732452

Acceleration Coefficients

Given that all parameters are fixed at the values given in section 4.2.2, the influence of different values for the acceleration coefficients, c_1 and c_2, were evaluated for the synthetic and MRI images. Tables 4.6 and 4.7 summarize these results. For these choices of the acceleration coefficients, no single choice is superior to the others. While these tables indicate an independence to the value of the acceleration coefficients, it is important to note that convergence depends on the relationship between the inertia weight and the acceleration coefficients, as derived in Van den Bergh [2002] (also refer to equation (2.9)).

Table 4.6: Effect of acceleration coefficients on the synthetic image

W	PSO			GCPSO		
	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
$c_1 = 0.7$ $c_2 = 1.4$	16.989197 ± 0.011786	24.726716 ± 0.101239	93.698591 ± 0.184244	16.989355 ± 0.012473	24.708151 ± 0.120168	93.667144 ± 0.207355
$c_1 = 1.4$ $c_2 = 0.7$	16.991884 ± 0.016970	24.700627 ± 0.125603	93.658673 ± 0.208500	16.993095 ± 0.040042	24.685461 ± 0.165669	93.619162 ± 0.279258
$c_1 = 1.49$ $c_2 = 1.49$	16.987582 ± 0.009272	24.710933 ± 0.122622	93.672792 ± 0.206395	16.995967 ± 0.039686	24.722414 ± 0.144572	93.680765 ± 0.253954

Table 4.7: Effect of acceleration coefficients on the MRI image

W	PSO			GCPSO		
	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
$c_1 = 0.7$ $c_2 = 1.4$	7.599324 ± 0.289702	10.14501 ± 1.353091	26.977217 ± 3.467738	7.530823 ± 0.477134	10.201762 ± 0.986726	26.425638 ± 3.248949
$c_1 = 1.4$ $c_2 = 0.7$	7.528712 ± 0.439470	10.23899 ± 1.484245	27.747333 ± 2.850575	7.476468 ± 0.459432	10.159019 ± 1.085977	27.001444 ± 3.360799
$c_1 = 1.49$ $c_2 = 1.49$	7.499845 ± 0.416682	10.20391 ± 0.951100	26.629647 ± 2.652593	7.467591 ± 0.396310	10.184191 ± 0.955129	26.596493 ± 3.208689

Sub-objective Weight Values

Tables 4.8 and 4.9 summarize the effects of different values of the weights, w_1, w_2 and w_3, of the sub-objectives for the synthetic and MRI images respectively. The results show that increasing the value of a weight, improves the corresponding fitness term. However, it is not so clear which sub-objective weight value combination is best for the synthetic and MRI images. To eliminate tuning of these weight values, an alternative multi-objective approach can be followed [Coello Coello 1996; Hu and Eberhart, Multiobjective 2002; Fieldsend and Singh 2002; Coello Coello and Lechuga 2002], or a non-parametric fitness function can be used as proposed in section 4.2.7.

4.2.5 *gbest* PSO versus *state-of-the-art* clustering algorithms

This section compares the performance of the *gbest* PSO and GCPSO with K-means, FCM, KHM, H2 and a GA clustering algorithm. This is done for a high $V_{max} = 255$. All other parameters are as for section 4.2.2. In all cases, for PSO, GCPSO and GA, 50 particles were trained for 100 iterations; for the other algorithms 5000 iterations were used (i.e. all algorithms have performed 5000 function evaluations). For FCM, q was set to 2 since it is the commonly used value [Hoppner *et al.* 1999]. For KHM and H2, α was set to 2.5 and 4 respectively since these values produced the best results according to our preliminary tests. For the GA, a tournament size of 2 was used, a uniform crossover probability of 0.8 with mixing ratio of 0.5, and a mutation probability of 0.05. Only the best individual survived to the next generation. The results are summarized in Table 4.10. These results are also averages over 20 simulation runs. Table 4.10 shows that PSO and GCPSO generally outperformed K-means, FCM, KHM and H2 in $d_{\min}$ and $\overline{d}_{\max}$, while performing comparably with respect to J_e (for the synthetic image, PSO performs significantly better than K-means, FCM, KHM and H2 with respect to J_e). The PSO, GCPSO and GA showed similar performance, with no significant difference.

These results show that the PSO-based clustering algorithms are viable alternatives that merit further investigation.

Table 4.8: Effect of sub-objective weight values on synthetic image

			PSO			**GCPSO**		
w_1	w_2	w_3	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
0.3	0.3	0.3	17.068816 ± 0.157375	24.67201 ± 0.572276	93.59498 ± 0.984724	17.01028 ± 0.059817	24.74227 ± 0.258118	93.711385 ± 0.437418
0.8	0.1	0.1	17.590421 ± 0.353375	21.76629 ± 0.127098	88.89228 ± 0.143159	17.51434 ± 0.025242	21.72462 ± 0.018983	88.879342 ± 0.062452
0.1	0.8	0.1	18.827495 ± 0.558357	27.62398 ± 0.427120	97.71945 ± 0.202744	18.82712 ± 0.688529	27.52239 ± 0.282601	97.768398 ± 0.266885
0.1	0.1	0.8	16.962755 ± 0.003149	24.49574 ± 0.089611	93.22883 ± 0.135893	16.98372 ± 0.122501	24.54688 ± 0.434417	92.576271 ± 4.357444
0.1	0.45	0.45	17.550448 ± 0.184982	26.70792 ± 0.692239	96.02056 ± 0.757185	17.55782 ± 0.226305	26.59844 ± 0.907974	95.888089 ± 1.152158
0.45	0.45	0.1	18.134349 ± 0.669151	26.48904 ± 0.982256	96.46178 ± 1.495491	18.29490 ± 0.525467	26.79529 ± 0.800436	96.922471 ± 1.225336
0.45	0.1	0.45	17.219807 ± 0.110357	22.63196 ± 0.522369	90.15281 ± 0.887423	17.20169 ± 0.093969	22.70116 ± 0.469470	90.289690 ± 0.828522

Table 4.9: Effect of sub-objective weight values on MRI image

			PSO			**GCPSO**		
w_1	w_2	w_3	J_e	$\bar{d}_{\max}$	$d_{\min}$	J_e	$\bar{d}_{\max}$	$d_{\min}$
0.3	0.3	0.3	7.239181 ± 0.576141	10.235431 ± 1.201349	24.705469 ± 3.364803	7.194243 ± 0.573013	10.403608 ± 1.290794	23.814072 ± 3.748753
0.8	0.1	0.1	7.364818 ± 0.667141	9.683816 ± 0.865521	24.021787 ± 3.136552	7.248268 ± 0.474639	9.327774 ± 0.654454	23.103375 ± 4.970816
0.1	0.8	0.1	8.336001 ± 0.599431	11.256763 ± 1.908606	31.106734 ± 1.009284	8.468620 ± 0.626883	11.430190 ± 1.901736	30.712733 ± 1.336578
0.1	0.1	0.8	6.160486 ± 0.241060	15.282308 ± 2.300023	2.342706 ± 5.062570	6.088302 ± 0.328147	15.571290 ± 2.410393	1.659674 ± 4.381048
0.1	0.45	0.45	7.359711 ± 0.423120	10.826327 ± 1.229358	24.536828 ± 3.934388	7.303304 ± 0.439635	11.602263 ± 1.975870	22.939088 ± 3.614108
0.45	0.45	0.1	8.001817 ± 0.391616	9.885342 ± 0.803478	28.057459 ± 1.947362	7.901145 ± 0.420714	9.657340 ± 0.947210	29.236420 ± 1.741987
0.45	0.1	0.45	6.498429 ± 0.277205	11.392347 ± 2.178743	12.119429 ± 8.274427	6.402205 ± 0.363938	10.939902 ± 2.301587	14.422413 ± 6.916785

Table 4.10: Comparison between K-means, FCM, KHM, H2, GA and PSO for fitness function defined in equation (4.6)

Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	**K-means**	20.21225 ± 0.937836	28.04049 ± 2.7779388	78.4975 ± 7.0628718
	FCM	20.731920 ± 0.650023	28.559214 ± 2.221067	82.434116 ± 4.404686
	KHM(*p*=2.5)	20.168574 ± 0.0	23.362418 ± 0.0	86.307593 ± 0.000008
	H2 (*p*=4)	20.136423 ± 0.793973	26.686939 ± 3.011022	81.834143 ± 6.022036
	GA	17.004002 ± 0.035146	24.603018 ± 0.11527	93.492196 ± 0.2567
	PSO	16.988910 ± 0.023937	24.696055 ± 0.130334	93.632200 ± 0.248234
	GCPSO	16.995967 ± 0.039686	24.722414 ± 0.144572	93.680765 ± 0.253954
MRI	**K-means**	7.3703 ± 0.042809	13.214369 ± 0.761599	9.93435 ± 7.308529
	FCM	7.205987 ± 0.166418	10.851742 ± 0.960273	19.517755 ± 2.014138
	KHM(*p*=2.5)	7.53071± 0.129073	10.655988 ± 0.295526	24.270841 ± 2.04944
	H2 (*p*=4)	7.264114 ± 0.149919	10.926594 ± 0.737545	20.543530 ± 1.871984
	GA	7.038909 ± 0.508953	9.811888 ± 0.419176	25.954191 ± 2.993480
	PSO	7.594520 ± 0.449454	10.186097 ± 1.237529	26.705917 ± 3.008073
	GCPSO	7.555421 ± 0.409742	9.983189 ± 0.915289	27.313118 ± 3.342264
Tahoe	**K-means**	3.280730 ± 0.095188	5.234911 ± 0.312988	9.402616 ± 2.823284
	FCM	3.164670 ± 0.000004	4.999294 ± 0.000009	10.970607 ± 0.000015
	KHM(*p*=2.5)	3.830761 ± 0.000001	6.141770 ± 0.0	13.768387 ± 0.000002
	H2 (*p*=4)	3.197610 ± 0.000003	5.058015 ± 0.000007	11.052893 ± 0.000012
	GA	3.472897 ± 0.151868	4.645980 ± 0.105467	14.446860 ± 0.857770
	PSO	3.523967 ± 0.172424	4.681492 ± 0.110739	14.664859 ± 1.177861
	GCPSO	3.609807 ± 0.188862	4.757948 ± 0.227090	15.282949 ± 1.018218

4.2.6 Different Versions of PSO

This section investigates the use of different versions of PSO, namely:

- *lbest* PSO (with $l = 2$).
- *gbest-to-lbest* PSO, starting start with an *lbest* implementation of the PSO (with zero-radius neighborhood i.e. $l = 0$) and linearly increasing the neighborhood radius until a *gbest* implementation of the PSO is reached. This hybrid approach is used in order to initially explore more, thus, avoid being trapped in local optima, by using a *lbest* approach [Suganthan 1999]. The algorithm then attempts to converge onto the best solution found by the initial phase by using a *gbest* approach.
- *gbest*- and *lbest*- PSO with mutation proposed by Esquivel and Coello Coello [2003] (discussed in section 2.6.8). In this approach, the PSO parameters where set as specified by Esquivel and Coello Coello [2003] (i.e. $w = 0.3$, $c_1 = c_2 = 1.3$). In addition, p_m is initialized to 0.9, then linearly decreases with increase in the number of iterations [Coello Coello 2003].

Table 4.11 summarizes the results of *gbest* PSO, GCPSO, *lbest* PSO, *gbest-to-lbest* PSO, *gbest* PSO with mutation and *lbest* PSO with mutation. In all the experiments, 50 particles were trained for 100 iterations and $V_{max} = 255$. All the other parameters are as for section 4.2.2 for all the approaches except the approaches that use mutation. Although the results are generally comparable, it can be observed that the *gbest-to-lbest* PSO is slightly better than the others. An explanation for this observation is the fact that *gbest-to-lbest* PSO starts with high diversity (therefore more exploration), then as the run progresses, the diversity is reduced (to focus more exploitation). This observation shows the importance of high diversity at the beginning of the run in order to avoid premature convergence and the importance of low diversity at the end of the run in order to fine tune the solution.

Table 4.11: Comparison of different PSO versions

Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	*gbest* **PSO**	16.988910 ± 0.023937	24.696055 ± 0.130334	93.632200 ± 0.248234
	GCPSO	16.995967 ± 0.039686	24.722414 ± 0.144572	93.680765 ± 0.253954
	lbest **PSO**	16.991791 ± 0.003523	24.771597 ± 0.004171	93.775989 ± 0.0
	gbest-to-lbest **PSO**	16.988325 ± 0.000273	24.774668 ± 0.000371	93.775989 ± 0.0
	gbest **PSO with mutation**	16.985563 ± 0.006350	24.728548 ± 0.110016	93.698591 ± 0.184244
	lbest **PSO with mutation**	16.995550 ± 0.015914	24.684511 ± 0.135164	93.646993 ± 0.223429
MRI	**PSO**	7.594520 ± 0.449454	10.186097 ± 1.237529	26.705917 ± 3.008073
	GCPSO	7.555421 ± 0.409742	9.983189 ± 0.915289	27.313118 ± 3.342264
	lbest **PSO**	7.676197 ± 0.138833	9.500085 ± 0.567423	29.684682 ± 0.929038
	gbest-to-lbest **PSO**	7.663361 ± 0.142196	9.211712 ± 0.502518	30.138389 ± 0.878266
	gbest **PSO with mutation**	7.301802 ± 0.474767	9.573999 ± 0.581114	27.691924 ± 3.145707
	lbest **PSO with mutation**	7.657294 ± 0.277544	9.890083 ± 0.696923	28.731981 ± 1.938404
Tahoe	**PSO**	3.523967 ± 0.172424	4.681492 ± 0.110739	14.664859 ± 1.177861
	GCPSO	3.609807 ± 0.188862	4.757948 ± 0.227090	15.282949 ± 1.018218
	lbest **PSO**	3.527251 ± 0.212840	4.778272 ± 0.217206	15.619541 ± 1.179783
	gbest-to-lbest **PSO**	3.460024 ± 0.289942	4.826269 ± 0.238982	15.985762 ± 1.410871
	gbest **PSO with mutation**	3.592122± 0.180782	4.750996 ± 0.213625	15.252226 ± 0.987399
	lbest **PSO with mutation**	3.660723± 0.121711	4.793518 ± 0.144508	15.522304 ± 0.597297

4.2.7 A Non-parametric Fitness Function

The fitness function defined in equation (4.6) provides the user with the flexibility of prioritizing the fitness term of interest by modifying the corresponding weight. However, it requires the user to find the best combination of w_1, w_2 and w_3 for each image which is not an easy task. Therefore, a non-parametric fitness function without weights is defined as

$$f(\boldsymbol{x}_i, \boldsymbol{Z}_i) = \frac{\bar{d}_{\max}(\boldsymbol{Z}_i, \boldsymbol{x}_i) + J_{e,i}}{d_{\min}(\boldsymbol{Z}_i, \boldsymbol{x}_i)} \tag{4.7}$$

The advantage of equation (4.7) is that it works with any data set without any user intervention. Table 4.12 is a repeat of Table 4.10, but with the results of *gbest* PSO using the non-parametric fitness function (referred to as *PSO noweights*) added. In general, the PSO using the non-parametric fitness function performed better than K-Means, FCM, KHM and H2 in terms of $d_{\min}$ and $\bar{d}_{\max}$, while performing comparably with respect to J_e. In addition, the PSO using the non-parametric fitness function performed comparably with GA, PSO and GCPSO using the parametric fitness function (equation (4.6)). Hence, the non-parametric fitness function (equation (4.7)) can be used instead of the parametric fitness function (equation (4.6)), thereby eliminating the need for tuning w_1, w_2 and w_3.

However, since the difference between J_e and $\bar{d}_{\max}$ on the one hand and $d_{\min}$ on the other hand is quite large (as can be observed from the results of this section), the value of the fitness function is usually less than one, and may incorrectly indicate a good clustering for large values of J_e and $\bar{d}_{\max}$. The proposed non-parametric fitness function therefore has the problem that the largest criterion tends to dominate the other criteria. To address this biased behavior, the values of J_e and $\bar{d}_{\max}$ are normalized to the range [0,0.5], while the value of $d_{\min}$ are normalized to the range [0,1]. Therefore, J_e + $\bar{d}_{\max}$ and $d_{\min}$ contributes equally the fitness function. Table 4.13 compares the performance of the non-normalized, non-parametric fitness function (*PSO noweights*) with the normalized, non-parametric fitness function (*PSO normalized noweights*). From Table 4.13, it can be observed that both non-parametric fitness functions performed comparably. Hence, it can be concluded that it is not necessary to normalize the non-parametric fitness function.

4.2.8 Multispectral Imagery Data

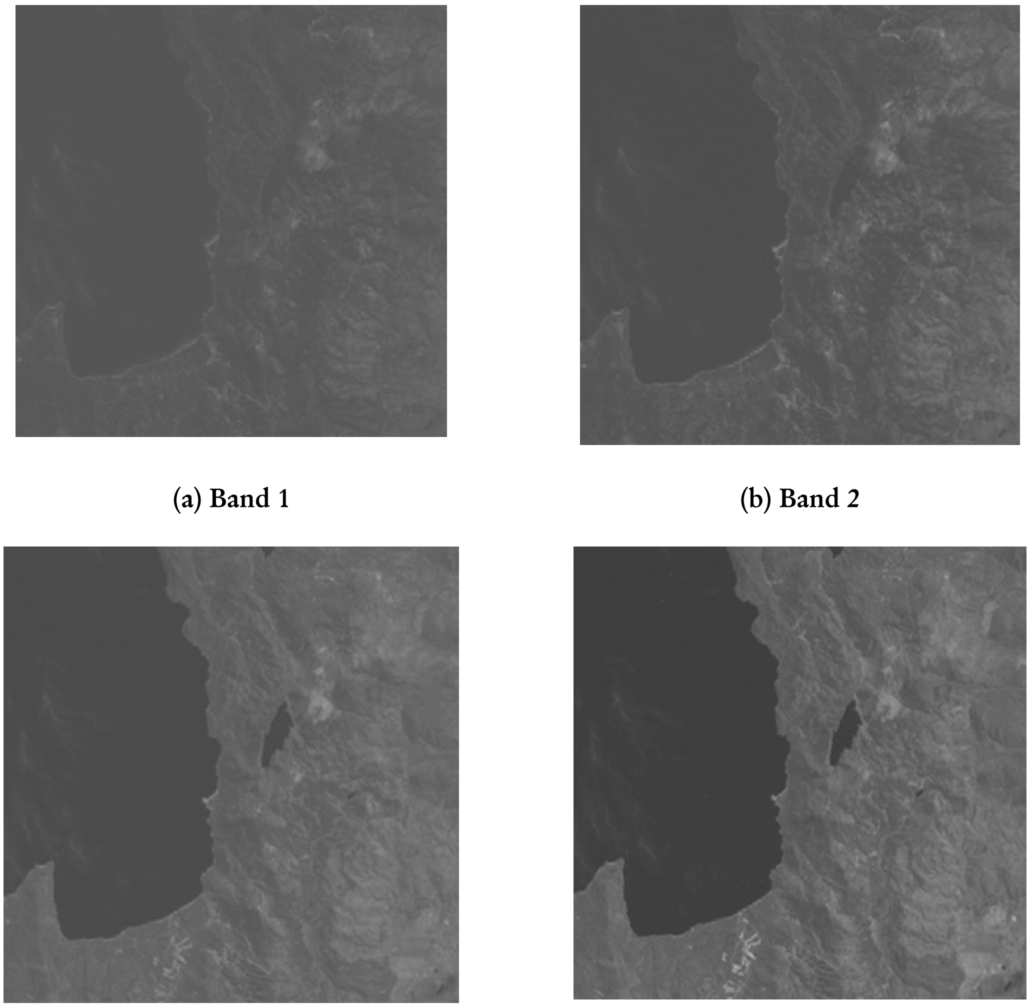
To illustrate the applicability of the proposed approach to multidimensional feature spaces, the PSO-based clustering algorithm was applied to the four-channel multispectral image set of the Lake Tahoe region in the US shown in Figure 4.9. Table 4.14 summarizes the results of applying K-means, *gbest* PSO and *lbest-to-gbest* PSO on the image set. In all the experiments, 50 particles were trained for 100 iterations and V_{max} = 255. All parameters are as for section 4.2.2. The results showed that both PSO approaches performed better than K-means in term of $d_{\min}$. However, *gbest* PSO performed comparably to K-means in terms of $\bar{d}_{\max}$, while *lbest-to-gbest* PSO performed comparably to K-means in terms of J_e. The segmented images (known as thematic maps) for the Tahoe image set are given in Figure 4.10.

Table 4.12: Comparison between K-means, FCM, KHM, H2, GA and PSO for fitness function defined in equation (4.7)

Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	**K-means**	20.21225 ± 0.937836	28.04049 ± 2.7779388	78.4975 ± 7.0628718
	FCM	20.731920 ± 0.650023	28.559214 ± 2.221067	82.434116 ± 4.404686
	KHM(*p*=2.5)	20.168574 ± 0.0	23.362418 ± 0.0	86.307593 ± 0.000008
	H2 (*p*=4)	20.136423 ± 0.793973	26.686939 ± 3.011022	81.834143 ± 6.022036
	GA	17.004002 ± 0.035146	24.603018 ± 0.11527	93.492196 ± 0.2567
	PSO	16.988910 ± 0.023937	24.696055 ± 0.130334	93.632200 ± 0.248234
	GCPSO	16.995967 ± 0.039686	24.722414 ± 0.144572	93.680765 ± 0.253954
	PSO noweights	17.284 ± 0.09	22.457 ± 0.414	90.06 ± 0.712
MRI	**K-means**	7.3703 ± 0.042809	13.214369 ± 0.761599	9.93435 ± 7.308529
	FCM	7.205987 ± 0.166418	10.851742 ± 0.960273	19.517755 ± 2.014138
	KHM(*p*=2.5)	7.53071± 0.129073	10.655988 ± 0.295526	24.270841 ± 2.04944
	H2 (*p*=4)	7.264114 ± 0.149919	10.926594 ± 0.737545	20.543530 ± 1.871984
	GA	7.038909 ± 0.508953	9.811888 ± 0.419176	25.954191 ± 2.993480
	PSO	7.594520 ± 0.449454	10.186097 ± 1.237529	26.705917 ± 3.008073
	GCPSO	7.555421 ± 0.409742	9.983189 ± 0.915289	27.313118 ± 3.342264
	PSO noweights	7.839 ± 0.238	9.197 ± 0.56	29.45 ± 1.481
Tahoe	**K-means**	3.280730 ± 0.095188	5.234911 ± 0.312988	9.402616 ± 2.823284
	FCM	3.164670 ± 0.000004	4.999294 ± 0.000009	10.970607 ± 0.000015
	KHM(*p*=2.5)	3.830761 ± 0.000001	6.141770 ± 0.0	13.768387 ± 0.000002
	H2 (*p*=4)	3.197610 ± 0.000003	5.058015 ± 0.000007	11.052893 ± 0.000012
	GA	3.472897 ± 0.151868	4.645980 ± 0.105467	14.446860 ± 0.857770
	PSO	3.523967 ± 0.172424	4.681492 ± 0.110739	14.664859 ± 1.177861
	GCPSO	3.609807 ± 0.188862	4.757948 ± 0.227090	15.282949 ± 1.018218
	PSO noweights	3.882 ± 0.274	5.036 ± 0.368	16.410 ± 1.231

Table 4.13: Comparison between different non-parametric fitness function				
Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Synthetic	**PSO noweights**	17.284 ± 0.09	22.457 ± 0.414	90.06 ± 0.712
	PSO normalized noweights	17.298567 ± 0.065019	22.387227 ± 0.295405	89.969316 ± 0.482432
MRI	**PSO noweights**	7.839 ± 0.238	9.197 ± 0.56	29.45 ± 1.481
	PSO normalized noweights	7.851594 ± 0.293330	9.182184 ± 0.534796	29.393441 ± 1.240797
Tahoe	**PSO noweights**	3.882 ± 0.274	5.036 ± 0.368	16.410 ± 1.231
	PSO normalized noweights	3.970922 ± 0.218675	5.141907 ± 0.312130	16.746504 ± 1.119426

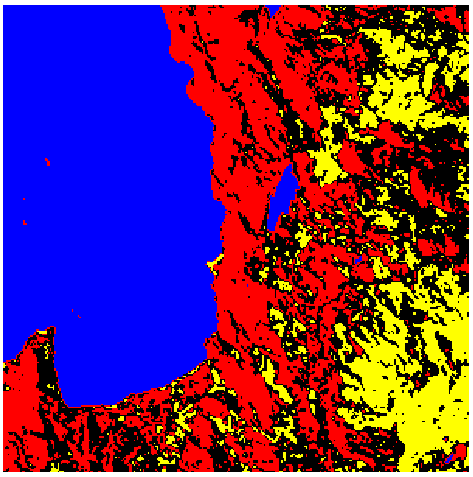
Table 4.14: Comparison between K-means, *gbest* PSO and *lbest-to-gbest* PSO when applied to multispectral image set				
Image		J_e	$\bar{d}_{\max}$	$d_{\min}$
Four-bands Lake Tahoe	**K-means**	7.281864 ± 0.001512	11.876593 ± 0.001526	17.675578 ± 0.008525
	PSO	8.005989 ± 0.812936	11.935493 ± 0.732004	19.937182 ± 3.468417
	gbest-to-lbest **PSO**	7.639596 ± 0.654930	12.173503 ± 0.740456	18.263982 ± 3.041869



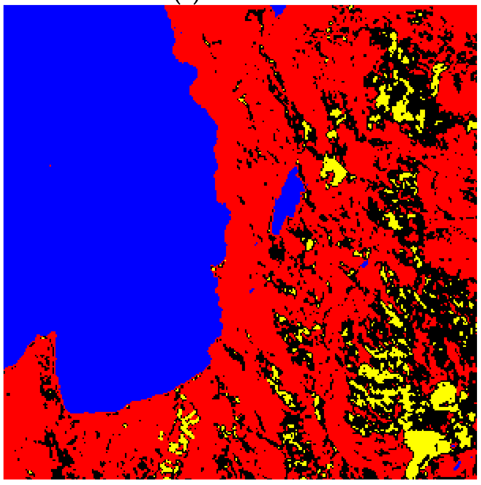
(a) Band 1 (b) Band 2

(c) Band 3 (d) Band 4

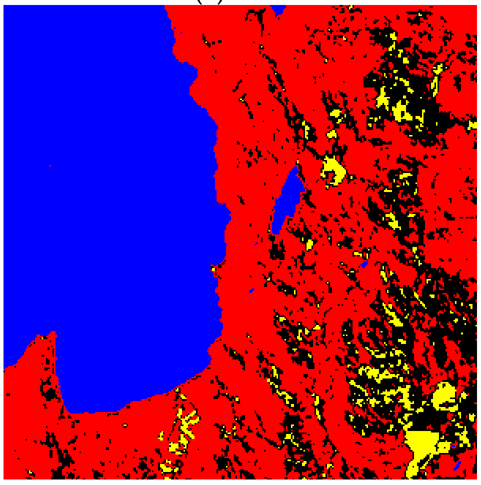
Figure 4.9: The Landsat MSS test images of Lake Tahoe



(a) K-means



(b) PSO



(c) *lbest-to-gbest* PSO

Figure 4.10: The Thematic Maps for Lake Tahoe Image Set

4.2.9 PSO for Data Clustering

The same algorithm presented in section 4.1.1 was used by Van der Merwe and Engelbrecht [2003] to cluster general data sets. It was applied on a set of multi-dimensional data (e.g. the Iris plant data base) using a fitness function consisting of J_e only. In general, the results show that the PSO-based clustering algorithm performs better than the K-means algorithm, which verify the results presented in this chapter. These results are expected since, as previously mentioned, K-means is a greedy algorithm which depends on the initial conditions, which may cause the algorithm to converge to suboptimal solutions. On the other hand, PSO is less sensitive to the effect of the initial conditions due to its population-based nature. Thus, PSO is more likely to find near-optimal solutions.

4.3 Conclusions

This chapter presented a new clustering approach using PSO. The PSO clustering algorithm has as objective to simultaneously minimize the quantization error and intra-cluster distances, and to maximize the inter-cluster distances. Both a *gbest* PSO and GCPSO algorithms have been evaluated. The *gbest* PSO and GCPSO clustering algorithms were further compared against K-means, FCM, KHM, H2 and a GA. In general, the PSO algorithms produced better results with reference to inter- and intra-cluster distances, while having quantization errors comparable to the other algorithms. The performance of different versions of PSO was investigated and the results suggested that algorithms that start with high diversity and then gradually reduces diversity perform better than other algorithms. A non-parametric version of the proposed fitness function was tested with encouraging results. Finally, the proposed approach was applied to multispectral imagery data.

In the next chapter, a new automatic image generation tool is proposed which is tailored specifically for verification and comparison of different unsupervised image classification algorithms. This tool is used to conduct a more elaborate comparison between the PSO and K-means clustering algorithms.

Chapter 5

SIGT: Synthetic Image Generation Tool for Clustering Algorithms

A new automatic image generation tool is proposed in this chapter tailored specifically for the verification and comparison of different unsupervised image classification algorithms. The tool can be used to produce different images (in raw format) with different criteria based on user specification. The user specifies the number of clusters to be included in the image along with the probability distribution that governs a set of points that belong to different clusters. On the other hand, the tool can be used to verify the degree of approximation a new algorithm has been able to achieve compared to the original image. This allows for a scientific confident comparison between any new algorithm and existing algorithms. The usefulness of the tool is demonstrated in this chapter with reference to the well-known K-means clustering algorithm and the PSO-based clustering algorithm proposed in the previous chapter.

5.1 Need for Benchmarks

Researchers usually use their own data sets to test the performance of their clustering algorithms [Puzicha *et al*. 2000; Rosenberger and Chehdi 2000; Lorette *et al*. 2000; Boujemaa 2000; Huang 2002]. In addition, many researchers create their own synthetic data to test their algorithms. This approach makes the comparison between different clustering algorithms difficult. To address this problem, it is necessary to create a simple tool which will help researchers to create synthetic images. Researchers can then apply their clustering algorithm on these images and evaluate the performance of these algorithms. Furthermore, researchers can agree to use some of these synthetic images as benchmarks making comparison between different clustering algorithms easier.

In this chapter, a new tool is proposed to generate benchmark images tailored specifically for clustering problems that have the capability to verify the clustering quality of any unsupervised image classification algorithm. This tool has the following benefits:

1. The tool can create synthetic images customized toward user-specific objectives. The user can create the images he/she wants with the desired complexity that suits his/her interests. The user specifies the number of clusters in advance to test the ability of the algorithm to find that number of clusters (especially in the case of unsupervised classification). Furthermore, the user can specify the degree of *closeness* (inter-cluster distances) between different clusters to control the complexity of different algorithms to be able to differentiate between close clusters. Finally, users are able, using the tool, to specify the distribution probability that govern the relationship between different points belonging to different clusters.
2. The tool can measure the quality of any clustering algorithm provided that it uses the tool's generated images and generate a thematic map image in a raw format. Hence, the user can measure the quality of a user-defined clustering algorithm to examine how efficient the algorithm is on different data sets.
3. According to the above benefits, the tool can be used to create a carefully crafted set of benchmark images. Hence, using SIGT, researchers can build common benchmark images to be used for comparison among different

clustering algorithms. The ability of the tool to easily create images with pre-defined clusters pushes towards this direction.

4. SIGT can be used to quantify the average performance of a user-specified clustering algorithm. This can be done by running the algorithm for a number of simulations on a library of benchmark images to statistically compare it to other algorithms.
5. This tool could be upgraded to generate a synthetic image from an existing image by relaxing some constraints. One way to do this is by calculating the histogram of the existing image and then composing a user defined separation period along the histogram, thus generating a modified cloned image.

Therefore, SIGT is best used as a preliminary test for any clustering algorithm (especially in the area of unsupervised image classification or segmentation). One of the advantages of the proposed tool is the ease with which one can measure the performance of a clustering algorithm, and compared its performance with other algorithms. The rest of the chapter is organized as follows: The proposed tool is described in detail in section 5.2. Section 5.3 provides experimental results verifying the applicability of the tool. Finally section 5.4 concludes the chapter.

5.2 SIGT: Synthetic Image Generation Tool

A synthetic image generation tool for clustering algorithms, SIGT, is proposed in this chapter to assist the unsupervised image classification research community's ability to compare and quantify the performance of different algorithms. The proposed tool consists of two units: a synthetic image generator unit, and a clustering verification unit. This section provides a detailed description of each unit.

5.2.1 Synthetic Image Generator

With the synthetic image generation algorithm, the user can generate a synthetic image in raw format (converting an image from/to raw format can be achieved easily by using different graphic editing tools such as Adobe Photoshop) suitable for his/her objectives by specifying the following characteristics of the desired image:

- Size in pixels (i.e. the number of image pixels), N_p
- Dynamic range in bits (e.g. 8-bit image)
- Number of clusters, K
- Histogram characteristics:
    - Percentage of points in the image that belongs to each cluster, C'_k, $\forall$ k = 1,…, K.
    - Each cluster width in pixels (e.g. 10-pixels wide), w'_k, $\forall$ k = 1,…, K.
    - The probability distribution that govern points in each cluster, p_k, $\forall$ k = 1,…, K. The tool allows the user to select any of the following discrete random distributions [Leon-Garcia 1994; Devore 1995]: Uniform, Binomial, Geometric, and/or Poisson. These distributions represent the most common distributions. Therefore, the user can create an image with the histogram of his/her choice.

- The separation (in number of pixels) between adjacent clusters $S_{k,kk} \geq 0$ where k, kk = 1,…, K, i.e. the inter-cluster distance between two adjacent clusters $\boldsymbol{C}_k$ and $\boldsymbol{C}_{kk}$.

After specifying the above information, the synthetic image can be constructed according to the algorithm summarized in Figure 5.1.

```
For each cluster $k = 1,\ldots, K$ do
    Distribute $C'_k.N_p/100$ pixels of cluster $\boldsymbol{C}_k$ according to $p_k$
Endloop
For each pixel $z_p$ do
    Assign $z_p$ randomly to any cluster $\boldsymbol{C}_k$, where $k = 1,\ldots, K$, based on the image
    histogram
Endloop
Create a thematic map image using

$z_p = \frac{255k}{K} \quad \forall\, p = 1,\ldots, N_p,\ k = 1,\ldots, K$        (5.1)

where $k$ is the index of cluster $\boldsymbol{C}_k$ to which $z_p$ belongs
```

Figure 5.1: The synthetic image generator algorithm

The synthetic image generation algorithm works as follows: The first step constructs a synthetic image histogram by distributing N_p image pixels into the K clusters according to the cluster distribution specified by the user. The number of pixels in cluster $\boldsymbol{C}_k$ can be determined by multiplying the percentage of points in the image belonging to cluster $\boldsymbol{C}_k$ by the number of image pixels N_p divided by 100 (i.e. $C'_k.N_p/100$). The second step assigns the synthetic image pixels randomly to the clusters according to the histogram created in the first step. Now, the synthetic image has been generated with the specified histogram. Finally, a thematic map image is generated according to equation (5.1). This map will help the user to verify the accuracy of any clustering algorithm when used in the clustering verification unit.

The synthetic image generator unit therefore generates two images: The first is the synthetic image representing the data set specified by the user, and the second is a thematic map of the generated synthetic image.

5.2.2 Clustering Verification Unit

The clustering verification unit verifies the average classification accuracy of the clustering algorithm being evaluated. This is done by comparing the thematic map generated from the synthetic image generation unit (or if the user has a thematic map representing the correct clustering) and the thematic map resulting from the clustering algorithm (this thematic map should be generated using equation (5.1)). This unit consists of two components, namely the clustering validity checker and the average performance analyzer. These components are described next.

5.2.2.1 Clustering Validity Checker Component

The clustering validity checker component verifies the average performance of any clustering algorithm with reference to a single image. It calculates the average classification accuracy over a number of trials. The component requires the user to specify the following information:

- Dynamic range in bits.
- Original image name (e.g. synthetic image name generated from the first unit).
- Reference thematic map name representing the correct classification (e.g. thematic map generated from the first unit), TM_r.
- Executable program name of the clustering algorithm.
- Thematic map resulting from the clustering algorithm being examined (e.g. K-means, FCM, etc.), TM_c.
- Image size in pixels, N_p.
- Number of clusters, K.
- Number of runs of the clustering program, N_r.

The clustering verification algorithm is summarized in Figure 5.2.

For each pixel in the image, the cluster numbers as generated from TM_r and TM_c are compared. If the cluster numbers match, a counter, *mcount*, is incremented. Finally, the classification accuracy is calculated using equation (5.2). This algorithm is repeated N_r times followed by calculating the average classification accuracy.

The clustering verification unit produces a binary image showing the difference between the two thematic maps. This difference is represented as a white colored pixel for each incorrectly classified pixel. Furthermore, the unit calculates an accuracy percentage according to these differences.

```
$mcount = 0$     /* number of correctly classified pixels */
For each pixel $z_p$ do
    $k$ = cluster number as in $TM_r$
    $kk$ = cluster number as in $TM_C$
    If $k \neq kk$ then
        Mark $z_p$ as a white dot in the difference image
    else
        $mcount = mcount + 1$
    Endif
Endloop
    $Classification\ accuracy = (mcount/N_p).100$                    (5.2)
```

Figure 5.2: The clustering verification algorithm

5.2.2.2 Average Performance Analyzer Component

The average performance analyzer component verifies the average performance of any clustering algorithm with reference to a set of images specified by the user (i.e. a benchmark library). It calculates the average classification accuracy by running the program a user-specified number of trials on each image in the library. In this regards, the user should specify the following information:

- Executable program name of the clustering algorithm.
- Thematic map resulting from the clustering algorithm being examined (e.g. K-means, FCM, etc.), TM_c.
- Number of trials to run the clustering program, N_r.

Other information (e.g. the number of benchmarks, name of benchmark images, reference thematic maps, etc.) should be specified in an input file called *SIGT_classifier_input.txt*.

The clustering verification algorithm in section 5.2.2.1 is also used in this component. The only difference is that the clustering program is applied to each image in the benchmark library. Note that the tool deals only with raw format images. The next section presents results obtained from SIGT to illustrate its features.

5.3 Experimental Results

This section shows how SIGT can be used to standardize the unsupervised image classification verification process. For the sake of showing its applicability, two clustering algorithms were used, namely K-means and PSO-based clustering algorithms. The tool can generate 8-bit, flexible size images. A preliminary core of a benchmark library consisting of ten synthetic images with different levels of complexity and pixel intensity distribution were created. Table 5.1 shows user-specified parameters used in generating the images. The first and second images were generated such that they resemble very clear separated clusters. Therefore, it must be very easy for most clustering techniques to determine these clusters with great accuracy. The first image resembles a two-cluster image, while the second resembles a three-cluster image. The third synthetic image was generated to be slightly more difficult than the previous two in such a way that two of its three clusters are close to each other. The fourth image has three clusters adjacent to each other. On the other hand, the fifth image has one more cluster than the fourth. In fact, the fourth and fifth images better approximate real-life images than the first three, which can only be considered for functionality verification rather than competitance of different clustering algorithms in early phases of algorithm creation. The remaining images are the most complex among all, where different difficulty levels are introduced in such a way that only competitive clustering algorithms will be able to efficiently cluster the regions in the images. A large number of adjacent clusters of different probability distributions were used in constructing these images.

For all the experiments, both K-means and *gbest* PSO-based clustering algorithms (using equation (4.6)) were averaged over 30 trials for each image in the benchmark library. The average classification accuracy and confidence interval (CI) are calculated (see Table 5.2). For the PSO-based clustering algorithm, 50 particles are used for 50 generations, $w_1 = w_2 = 0.3$ and $w_3 = 0.4$. The inertia weight, w, is set to 0.72, and $c_1 = c_2 = 1.49$. The velocities are clamped using $V_{max} = 255$.

Using the clustering verification unit of SIGT, the thematic maps obtained from both K-means and PSO clustering algorithms were compared with the thematic maps generated by the synthetic image generation unit. The images representing the difference in thematic maps are included in Table 5.2. The average classification accuracy, calculated using equation (5.2), and the confidence interval of both algorithms are included in Table 5.2 for each image.

It is observed that as the separation between adjacent clusters decreases, the classification accuracy becomes lower. Note that the PSO-based clustering algorithm performed better than K-means in all the cases except two (Image 8 and 10). The rationale for the poor performance of the PSO-based clustering algorithm when applied to Image 10 is the choice of w_1, w_2 and w_3. When the PSO-based clustering algorithm was applied to Image 10 using $w_1 = w_2 = 0$ and $w_3 = 1$ (i.e. making the PSO fitness function similar to the objective function of K-means) the average classification accuracy significantly improved to reach 80.44% ± 7.411 with CI = [74.214, 86.674].

From the overall average performance, it can be concluded that the PSO-based clustering algorithm is in general better than the K-means clustering algorithm which verifies the results of chapter 4. To be able to make such a conclusion is one of the main benefits of SIGT.

Although the synthetic images represent no visually appealing shape, their histograms represent exactly what the user intends to test. The rationale behind this is that a clustering algorithm generally clusters pixels in a spectral domain (as represented in the histogram) rather than a spatial domain (as represented by the image shape). Therefore, the shape is generally not used, but rather the image histogram. However, in image segmentation the spatial information is important as already discussed in Section 3.2.

5.4 Conclusions

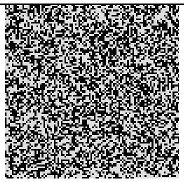
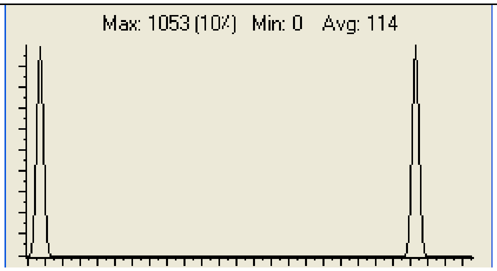
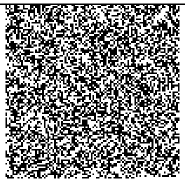
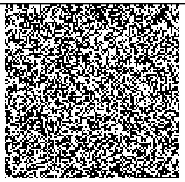
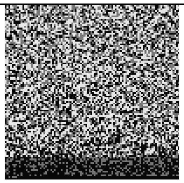
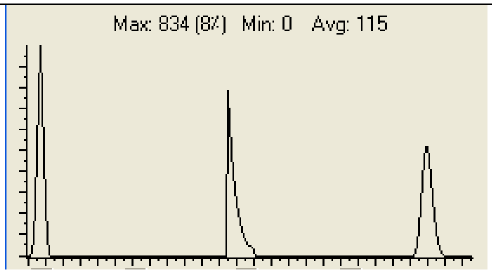
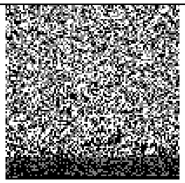
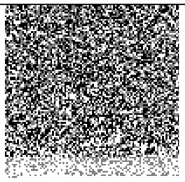
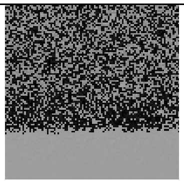
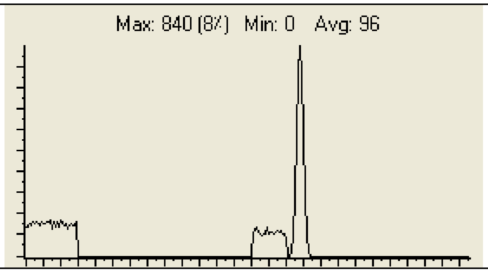
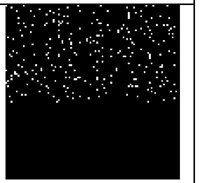
A new tool for synthetic image generation (SIGT) was proposed and implemented. The tool consists of two units: a synthetic image generator and a clustering verification unit. The first unit allows the user to create a synthetic image based on a user-specified histogram suitable for the required application. The second unit allows the user to measure the efficiency of a clustering algorithm. The main purpose of SIGT was to provide a simple and easy tool to generate synthetic images that can be used as benchmarks and to conduct a preliminary test on a clustering algorithm in order to measure its performance and compare it with other clustering algorithms. Different features of SIGT were demonstrated by a set of experiments aided by a well-known clustering algorithm (K-means) and the recent PSO-based clustering algorithm. These experiments have shown that the tool can be used to generate a synthetic image based on a user-specified histogram, measure the quality of any clustering algorithm, and create benchmarks.

After showing the potential of PSO as a clustering algorithm and proposing a tool that can aid in evaluating the efficiency of a clustering algorithm, the next chapter will address the difficult problem of determining the "optimal" number of clusters in a data set.

Table 5.1: Synthetic image details and classification accuracy

Benchmark No.	*K*	***% of each cluster*** (C'_k)	***Cluster distribution*** (p_k)
1	2	1 (50%) 2 (50%)	1 (Binomial) 2 (Binomial)
2	3	1 (34%) 2 (33%) 3 (33%)	1 (Binomial) 2 (Geometric) 3 (Poisson)
3	3	1 (40%) 2 (20%) 3 (40%)	1 (Uniform) 2 (Uniform) 3 (Binomial)
4	3	1 (40%) 2 (20%) 3 (40%)	1 (Uniform) 2 (Uniform) 3 (Uniform)
5	4	1 (30%) 2 (20%) 3 (30%) 4 (20%)	1 (Uniform) 2 (Uniform) 3 (Uniform) 4 (Poisson)
6	10	1 (10%) 2 (5%) 3 (10%) 4 (10%) 5 (5%) 6 (10%) 7 (15%) 8 (10%) 9 (10 %) 10 (15%)	1 (Uniform) 2 (Uniform) 3 (Uniform) 4 (Poisson) 5 (Uniform) 6 (Binomial) 7 (Geometric) 8 (Uniform) 9 (Poisson) 10 (Binomial)
7	6	1 (20%) 2 (20%) 3 (15%) 4 (30%) 5 (5%) 6 (10%)	1 (Poisson) 2 (Binomial) 3 (Uniform) 4 (Uniform) 5 (Uniform) 6 (Uniform)
8	4	1 (25%) 2 (25%) 3 (25%) 4 (25%)	1 (Geometric) 2 (Binomial) 3 (Binomial) 4 (Uniform)
9	7	1 (20%) 2 (10%) 3 (35%) 4 (5%) 5 (15%) 6 (15%) 7 (10%)	1 (Uniform) 2 (Uniform) 3 (Uniform) 4 (Uniform) 5 (Uniform) 6 (Uniform) 7 (Binomial)
10	4	1 (25%) 2 (25%) 3 (25%) 4 (25%)	1 (Poisson) 2 (Binomial) 3 (Uniform) 4 (Uniform)

Table 5.2: Synthetic images, Histograms and Thematic Maps

Bench Mark No.	Synthetic Image	Histogram	K-Means Sample TM_c (Avg. Classification Accuracy±SD[a]) [CI]	PSO Sample TM_c (Avg. Classification Accuracy±SD) [CI]	Best K-Means TM Difference (Best Classification Accuracy)	Best PSO TM Difference (Best Classification Accuracy)
1			 100%±0 [100, 100]	 100%±0 [100, 100]	(100%)	(100%)
2			 100%±0 [100, 100]	 100%±0 [100, 100]	(100%)	(100%)
3			79.58%±18.672 [72.901, 86.265]	96.49%±0.491 [96.310, 96.662]	(98.25%)	 (97.30%)

[a]SD stands for standard deviation

Table 5.2 (*continued*).

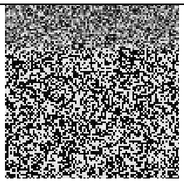
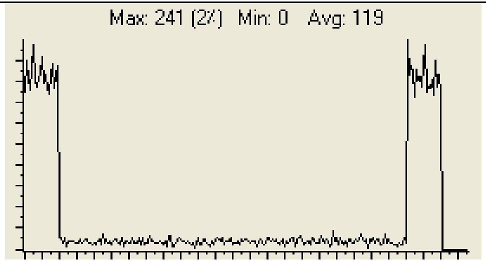
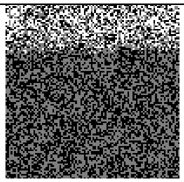
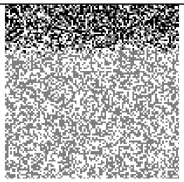
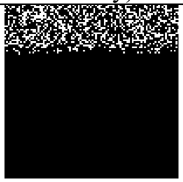
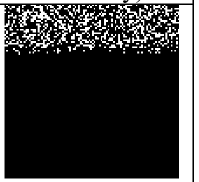
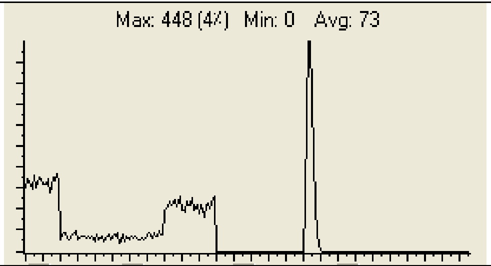
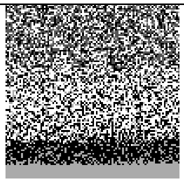
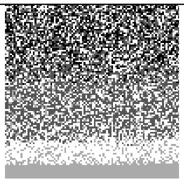
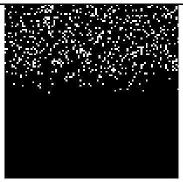
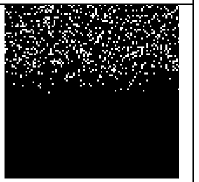
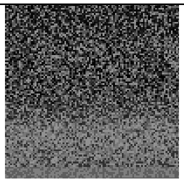
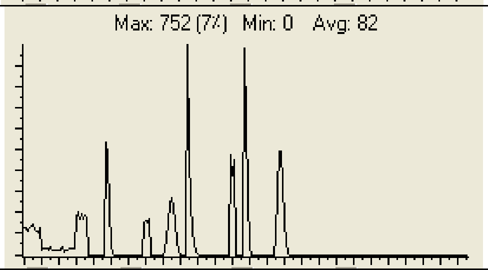
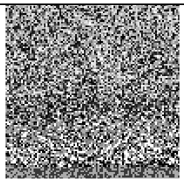
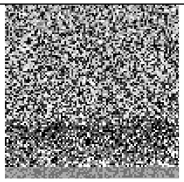
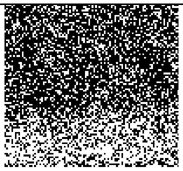
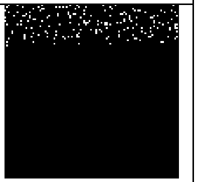
Bench Mark No.	Synthetic Image	Histogram	K-Means Sample *TM*$_c$ (Avg. Classification Accuracy±SD) [CI]	PSO Sample *TM*$_c$ (Avg. Classification Accuracy±SD) [CI]	Best K-Means TM Difference (Best Classification Accuracy)	Best PSO TM Difference (Best Classification Accuracy)
4			 90.56%±0 [90.560, 90.560]	 90.69%±0.060 [90.664, 90.707]	 (90.56%)	 (90.77%)
5			 91.75%±7.647 [89.018, 94.490]	 92.18%±0.318 [92.070, 92.298]	 (93.21%)	 (92.48%)
6			 50.91%±9.543 [47.494, 54.323]	 60.53%±26.043 [51.213, 69.852]	 (56.71%)	 (98.11%)

Table 5.2 (*continued*).

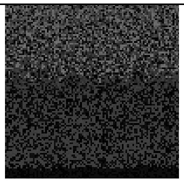
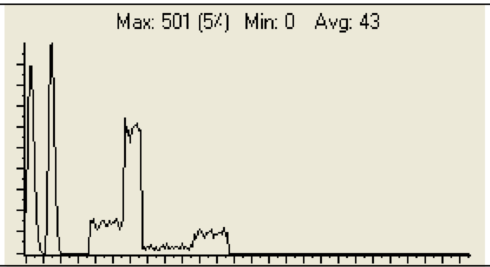
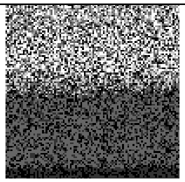
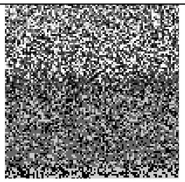
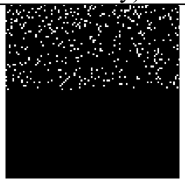
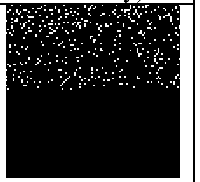
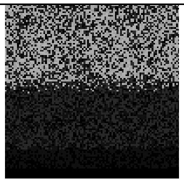
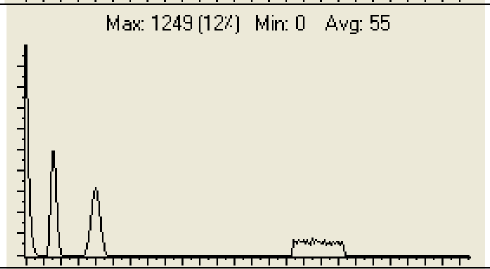
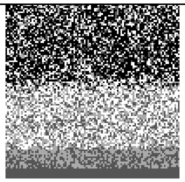
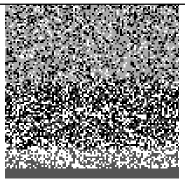
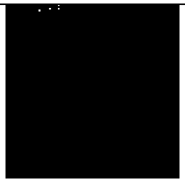
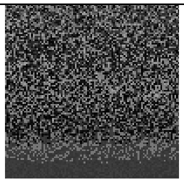
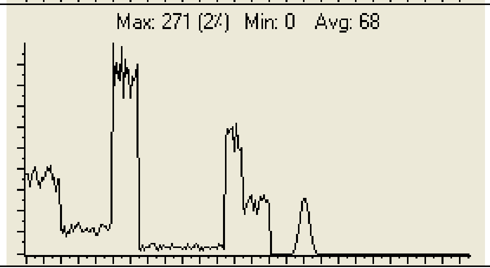
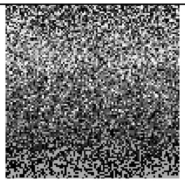
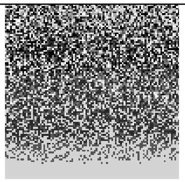
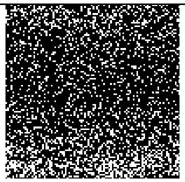
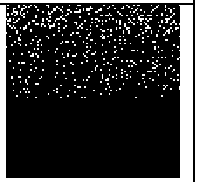
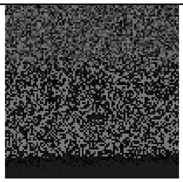
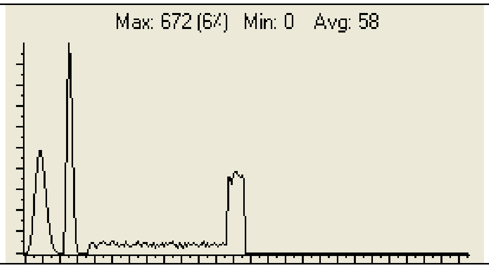
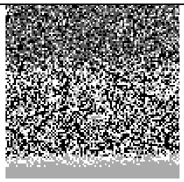
Bench Mark No.	Synthetic Image	Histogram	K-Means Sample TM_c (Avg. Classification Accuracy±SD) [CI]	PSO Sample TM_c (Avg. Classification Accuracy±SD) [CI]	Best K-Means TM Difference (Best Classification Accuracy)	Best PSO TM Difference (Best Classification Accuracy)
7			 60.62%±27.284 [50.856, 70.383]	 77.14%±14.021 [72.119, 82.154]	 (95.81%)	 (95.59%)
8			 99.96%±0 [99.96, 99.96]	 99.96%±0 [99.96, 99.96]	 (99.96%)	(99.96%)
9			 55.23%±20.975 [47.723, 62.734]	 60.19%±13.428 [55.384, 64.995]	 (75.99%)	 (94.08%)

Table 5.2 (*continued*).

Bench Mark No.	Synthetic Image	Histogram	K-Means Sample TM_c (Avg. Classification Accuracy±SD) [CI]	PSO Sample TM_c (Avg. Classification Accuracy±SD) [CI]	Best K-Means TM Difference (Best Classification Accuracy)	Best PSO TM Difference (Best Classification Accuracy)
10			 88.83%±0 [88.83, 88.83]	60.14%±0.211 [60.065, 60.216]	(88.83%)	(60.47%)
		Average Performance	81.74%±18.25123 [70.43,93.05]	83.73%±16.63667 [73.41,94.04]		

Chapter 6

Dynamic Clustering using Particle Swarm Optimization with Application to Unsupervised Image Classification

A new dynamic clustering approach (DCPSO), based on PSO, is proposed in this chapter. This approach is applied to unsupervised image classification. The proposed approach automatically determines the "optimum" number of clusters and simultaneously clusters the data set with minimal user interference. The algorithm starts by partitioning the data set into a relatively large number of clusters to reduce the effects of initial conditions. Using binary particle swarm optimization the "best" number of clusters is selected. The centroids of the chosen clusters are then refined via the K-means clustering algorithm. The proposed approach is then applied on synthetic, natural and multispectral images. The experiments conducted show that the proposed approach generally found the "optimum" number of clusters on the tested images. A genetic algorithm and a random search version of dynamic clustering are presented and compared to the particle swarm version.

6.1 The Dynamic Clustering using PSO (DCPSO) Algorithm

This section presents the DCPSO algorithm. For this purpose, define the following symbols:

N_c is the maximum number of clusters.

N_d is the dimension of the data set.

N_p is the number of patterns to be clustered.

$\boldsymbol{Z} = \{z_{j,p} \in \Re \mid j = 1,\ldots, N_d \text{ and } p = 1,\ldots, N_p\}$ is the set of patterns.

$\boldsymbol{M} = \{m_{j,k} \in \Re \mid j = 1,\ldots, N_d \text{ and } k = 1,\ldots, N_c\}$ is the set of N_c cluster centroids.

$\boldsymbol{S} = \{\boldsymbol{x}_1,\ldots, \boldsymbol{x}_i,\ldots, \boldsymbol{x}_s\}$ is the swarm of s particles such that $\boldsymbol{x}_i$ indicates particle i, with $x_{i,k} \in \{0,1\}$ for $k = 1,\ldots, N_c$ such that if $x_{i,k} = 1$ then the corresponding centroid $\boldsymbol{m}_k$ in $\boldsymbol{M}$ has been chosen to be part of the solution proposed by particle $\boldsymbol{x}_i$. Otherwise, if $x_{i,k} = 0$ then the corresponding $\boldsymbol{m}_k$ in $\boldsymbol{M}$ is not part of the solution proposed by $\boldsymbol{x}_i$.

n_i is the number of clusters used by the clustering solution represented by particle $\boldsymbol{x}_i$ such that

$$n_i = \sum_{k=1}^{N_c} x_{i,j} \text{ , with } n_i \le N_c.$$

$\boldsymbol{M}_i$ is the clustering solution represented by particle $\boldsymbol{x}_i$ such that $\boldsymbol{M}_i = (\boldsymbol{m}_k)\ \forall\ k\!: x_{i,k} = 1$ with $\boldsymbol{M}_i \subseteq \boldsymbol{M}$.

n_τ is the number of clusters used by the clustering solution represented by the global best particle $\hat{\boldsymbol{y}}$ (assuming that *gbest* PSO is used) such that

$$n_\tau = \sum_{k=1}^{N_c} \hat{y}_k \text{ , with } n_\tau \le N_c.$$

$\boldsymbol{M}_\tau$ is the clustering solution represented by $\hat{\boldsymbol{y}}$ such that $\boldsymbol{M}_\tau = (\boldsymbol{m}_k)\ \forall\ k\!:\ \hat{y}_k = 1$ with $\boldsymbol{M}_\tau \subseteq \boldsymbol{M}$.

$\boldsymbol{M}_r$ is the set of centroids in $\boldsymbol{M}$ which have not been chosen by $\hat{\boldsymbol{y}}$, i.e. $\boldsymbol{M}_r = (\boldsymbol{m}_k),\ \forall\ k\!:$ $\hat{y}_k = 0$ with $\boldsymbol{M}_r \subseteq \boldsymbol{M}$ (i.e. $\boldsymbol{M}_r \cap \boldsymbol{M}_\tau = \varnothing$ and $\boldsymbol{M}_r \cup \boldsymbol{M}_\tau = \boldsymbol{M}$).

p_{ini} is a user-specified probability defined in Kuncheva and Bezdek [1998], which is used to initialize a particle position, $\boldsymbol{x}_i$, as follows:

$$x_{i,k}(t) = \begin{cases} 0 & \text{if } r_k(t) \geq p_{\text{ini}} \\ 1 & \text{if } r_k(t) < p_{\text{ini}} \end{cases} \tag{6.1}$$

where $r_k(t) \sim U(0,1)$. Obviously a large value for p_{ini} results in selecting most of the centroids in ***M***.

The algorithm which uses some of the ideas presented by Kuncheva and Bezdek [1998]: A pool of cluster centroids, ***M***, is randomly chosen from ***Z***. The swarm of particles, ***S***, is then randomly initialized. Binary PSO is then applied to find the "best" set of cluster centroids, $\boldsymbol{M}_\tau$, from ***M***. K-means is applied to $\boldsymbol{M}_\tau$ in order to refine the chosen centroids. ***M*** is then set to $\boldsymbol{M}_\tau$ plus $\boldsymbol{M}_r$, which is a randomly chosen set of centroids from ***Z*** (this is done to add diversity to ***M*** and to reduce the effect of the initial conditions). The algorithm is then repeated using the new ***M***. When the termination criteria are met, $\boldsymbol{M}_\tau$ will be the resulting "optimum" set of cluster centroids and n_τ will be the "optimum" number of clusters in ***Z***. The DCPSO algorithm is summarized in Figure 6.1.

The termination criterion can be a user-defined maximum number of iterations or a lack of progress in improving the best solution found so far for a user-specified consecutive number of iterations, *TC*. In this chapter, the latter approach is used with $TC_1 = 50$ for Step 6 and $TC_2 = 2$ for step 10. These values for *TC* were set empirically. N_c and s are user defined parameters.

```
1) Select $\boldsymbol{m}_k \in \boldsymbol{M}$, $\forall\ k = 1,\ldots, N_c$ where $1 < N_c < N_p$, randomly from $\boldsymbol{Z}$
2) Initialize the swarm $\boldsymbol{S}$, with $x_{i,k} \sim U\{0,1\}$, $\forall\ i = 1,\ldots, s$ and $k = 1,\ldots, N_c$ using equation (6.1)
3) Randomly initialize the velocity, $\boldsymbol{v}_i$, of each particle $i$ in $\boldsymbol{S}$ such that $v_{i,k} \in [-5,5]$, $\forall\ i = 1,\ldots, s$ and $k = 1,\ldots, N_c$. The range of [-5,5] was set empirically
4) For each particle $\boldsymbol{x}_i$ in $\boldsymbol{S}$
    a. Partition $\boldsymbol{Z}$ according to the centroids in $\boldsymbol{M}_i$ by assigning each data point $\boldsymbol{z}_p$ to the closest (in terms of the Euclidean distance) cluster in $\boldsymbol{M}_i$
    b. Calculate the clustering validity index, $VI$, using one of the clustering validity indices as defined in section 3.1.4 to measure the quality of the resulting partitioning of $\boldsymbol{Z}$ (i.e. $VI = V$, $VI = S\_Dbw$ or $VI=1/D$ since $D$ should be maximized)
    c. $f(\boldsymbol{x}_i) = VI$
5) Apply the binary PSO velocity and position update equations (2.8) and (2.15) on all particles in $\boldsymbol{S}$
6) Repeat steps 4) and 5) until a termination criterion is met
7) Adjust $\boldsymbol{M}_\tau$ by applying the K-means clustering algorithm
8) Randomly re-initialize $\boldsymbol{M}_r$ from $\boldsymbol{Z}$
9) Set $\boldsymbol{M} = \boldsymbol{M}_r \cup \boldsymbol{M}_\tau$
10) Repeat steps 2) through 9) until a termination criterion is met
```

Figure 6.1: The DCPSO algorithm

A GA version of DCPSO can easily be implemented by replacing step 5 in the above algorithm with GA evolutionary operators for selection, crossover and mutation. On the other hand, a random search (RS) version of DCPSO, as described by Kunchevea and Bezdek [1998], can be implemented by removing step 5 and keeping only the best solution encountered so far.

As an illustration of the DCPSO algorithm, consider the following example.

Example 6.1

Let $N_p = 100$, $N_d = 1$, and $N_c = 5$.

Let $\boldsymbol{M}$ be

3	29	78	150	200

An example of a particle position, $\boldsymbol{x}_i$, is

0	1	1	0	1

which means that cluster centers 29, 78 and 200 are chosen for this particle such that $\boldsymbol{M}_i$ is

29	78	200

In other words, all data in $\boldsymbol{Z}$ are grouped in only these three clusters.

After step 6, assume the global best particle, $\hat{\boldsymbol{y}}$, is

0	1	0	1	1

Then, $\boldsymbol{M}_\tau$ is

29	150	200

Assume that after K-means is applied on $\boldsymbol{Z}$ using the centroids given by $\boldsymbol{M}_\tau$, the new $\boldsymbol{M}_\tau$ is given by

30.5	129.9	201

Then, randomly initialize the remaining N_c - n_τ (i.e. 5 - 3 = 2) clusters, representing $\boldsymbol{M}_r$, from $\boldsymbol{Z}$ (shown below in bold). The resultant $\boldsymbol{M}$ may look as follows:

110	30.5	**8**	129.9	201

The DCPSO algorithm is then repeated using the new $\boldsymbol{M}$.

6.1.1 Validity Index

One of the advantages of DCPSO is that it can use any validity index. Therefore, the user can choose the validity index suitable for his/her data set. In addition, any new index can easily be integrated with DCPSO. The validity indices used in this chapter are D, V and S_Dbw (as defined in section 3.1.4).

6.1.2 Time Complexity

The time complexity of DCPSO is based on the complexity of four processes, namely, the partitioning of $\boldsymbol{Z}$, calculating the quality of the partition, applying binary PSO and applying K-means. Assume that T_1 is the number of iterations taken by the PSO to converge (step 6 of the algorithm), and that T_2 is the number of iterations taken by DCPSO to converge (step 10 of the algorithm). Then the complexity of partitioning $\boldsymbol{Z}$ is $O(sT_1T_2N_cN_pN_d)$, while the complexity of calculating the quality of a partition will depend on the time complexity of the validity index which is, in general, some constant, ξ, multiplied by N_p for the indices used in this chapter. The complexity of this step is therefore $O(\xi T_1T_2N_p)$. Finally, the complexity of K-means is $O(N_p)$. The parameters T_1, T_2, N_c, s and ξ can be fixed in advance. Typically, T_1, T_2, N_c, s, ξ, N_d << N_p. Let ς be the multiplication of s, T_1, T_2, N_c and N_d (i.e. $\varsigma = sT_1T_2N_cN_d$). If $\varsigma << N_p$ then the time complexity of DCPSO will be $O(N_p)$. However, if $\varsigma \approx N_p$ then the time complexity of DCPSO will be $O(N_p^2)$.

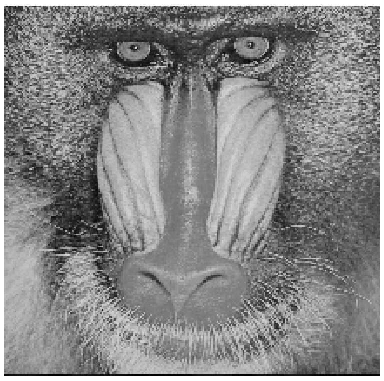
6.2 Experimental results

Experiments were conducted using both synthetic images and natural images. The synthetic images were generated by SIGT as given in Table 5.1. Furthermore, SIGT was used to generate another five different synthetic images for which the actual number of clusters was known in advance. These images have different numbers of clusters with varying complexities; they consist of well separated clusters, overlapping clusters or a combination of both. The new five synthetic images are given in Table 6.1 along with their histograms.

The following well known natural images were used: *Lenna*, *mandrill*, *jet* and *peppers*. These images are shown in Figure 6.2. Furthermore, one MRI and one satellite image of Lake Tahoe (as given in Figure 4.2) have been used to show the wide applicability of the proposed approach.

The remainder of this section is organized as follows: Section 6.2.1 applies DCPSO to the synthetic images using the three validity indices described in section 6.1.1. These results are compared with the unsupervised fuzzy approach (UFA) proposed by Lorette *et al.* [2000] (discussed in section 3.1.5) and the SOM approach (refer to section 3.1.6). In section 6.2.2, the same experiments are conducted on the natural images. Section 6.2.3 compares DCPSO with GA and RS versions on the natural images. Sections 6.2.4, 6.2.5 and 6.2.6 investigate the influence of the different DCPSO control parameters. Different PSO models (namely, *lbest*, *gbest* and *lbest-to-gbest*) are investigated in section 6.2.7. Finally, section 6.2.8 applies DCPSO to multispectral imagery data.

(a) Lenna

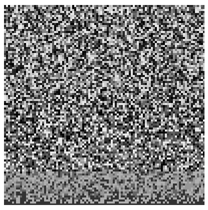
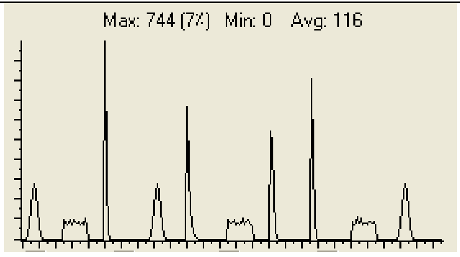
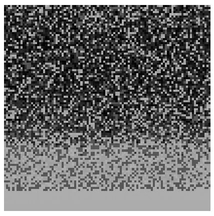
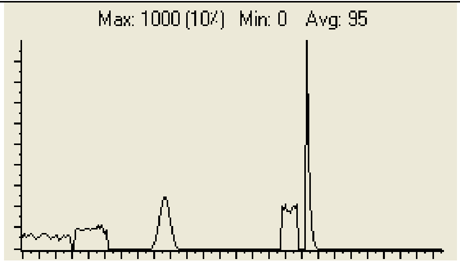
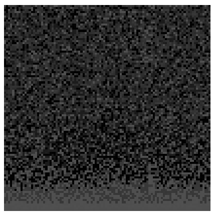
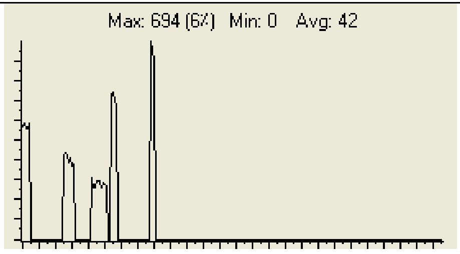
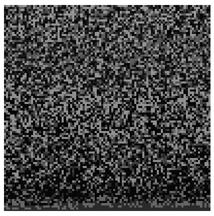
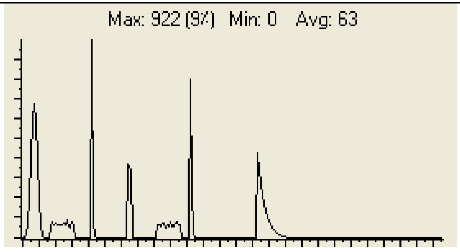
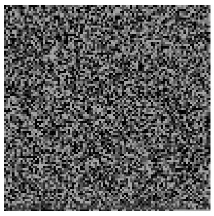
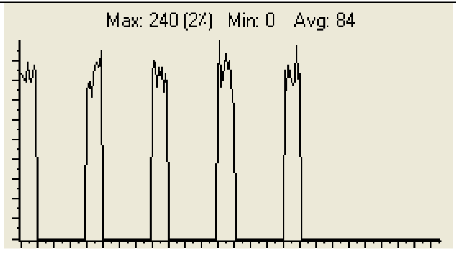

(b) Mandrill

(c) Jet

(d) Peppers

Figure 6.2: Natural Images

Table 6.1: Additional synthetic images used along with the corresponding histograms

Synthetic image no.	Image	Histogram
11		
12		
13		
14		
15		

The results reported in this section are averages and standard deviations over 20 simulations. Since *lbest-to-gbest* PSO was generally the best performer in chapter 4, *lbest-to-gbest* PSO is used in this section unless otherwise specified. Furthermore, if the best solution has not been improved after a user-specified number of iterations (50 iterations was used for all the experiments conducted) then the algorithm was terminated (Step 6 of the algorithm, Section 6.1). For the index proposed by Turi [2001], parameter c was set to 25 in all experiments as recommended by by Turi [2001]. The DCPSO parameters were empirically set as follows: $N_c = 20$, $p_{\text{ini}} = 0.75$ and $s = 100$ for all experiments conducted unless otherwise specified. In addition, the PSO parameters were set as follows: $w = 0.72$, $c_1 = c_2 = 1.49$ and $V_{\max} = 255$. For UFA, the user-defined parameter, ε, was set equal to $1/N_p$ as suggested by Lorette *et al.* [2000]. For the SOM, a Kohonen network of 5×4 nodes was used (to give a minimum of 20 codebook vectors). All implementation issues were set as in Pandya and Macy [1996]: the learning rate $\eta(t)$ was initially set to 0.9 then decreased by 0.005 until it reached 0.005; the neighborhood function $\Delta_w(t)$ was initially set to (5+4)/4 then decreased by 1 until it reached zero.

6.2.1 Synthetic images

Table 6.2 summarizes the results of DCPSO using the three validity indices described in section 6.1.1, along with the UFA and SOM results. It appears that UFA tends to overfit the data since it selected the maximum number of clusters as the correct one for all experiments. The rationale behind this failure is the choice of ε which has a significant effect on the resulting number of clusters. DCPSO using *S_Dbw* also generally overfits the data. On the other hand, DCPSO using *D*, DCPSO using *V* and

SOM have generally performed very well (especially DCPSO using *V*). Hence, it can be concluded that DCPSO using *V* is efficient with respect to the synthetic images.

Table 6.2: Experiments on synthetic images

Image	Actual no. of clusters	DCPSO using *D*	DCPSO using *V*	DCPSO using *S_Dbw*	SOM	UFA
1	2	2 ± 0	2 ± 0	5.55 ± 5.22	2	20
2	3	3 ± 0	3 ± 0	3 ± 0	3	20
3	3	2 ± 0	2 ± 0	4.4 ± 4.852	6	20
4	3	2.7 ± 1.345	5.15 ± 0.357	10.9 ± 5.458	10	20
5	4	10.85 ± 1.878	5 ± 0	15.5 ± 1.323	7	20
6	10	9.55 ± 2.246	7.2 ± 0.872	9.3 ± 0.458	9	20
7	6	3.35 ± 1.526	7.9 ± 0.995	10.8 ± 2.925	9	20
8	4	8.8 ± 2.379075	5 ± 0	4 ± 0	4	20
9	7	4.25 ± 0.433	5 ± 0	14.1 ± 3.52	13	20
10	4	7.9 ± 1.729	7 ± 0	13.95 ± 1.77	9	20
11	10	10.0 ± 0.950	10 ± 0	10 ± 0	10	20
12	5	9.0 ± 2.168	7.2 ± 0.4	11.65 ± 1.06	6	20
13	5	12.1 ± 2.119	5 ± 0	9.6 ± 2.107	5	20
14	7	7.5 ± 1.204	5 ± 0	7.8 ± 2.088	7	20
15	5	5 ± 0	5 ± 0	5 ± 0	5	20
Avg.	**5.2**	**6.53**	**5.43**	**9.04**	**7**	**20**

6.2.2 Natural images

Table 6.3 shows the results of DCPSO using the three validity indices described in section 6.1.1. These results are compared with the results of UFA and SOM. In addition, the results of snob for the Lenna, mandrill, jet and peppers images are copied from Turi [2001]. The optimal range for the number of clusters for the images of Lenna, mandrill, jet and peppers are also taken from Turi [2001] which was based on a visual analysis survey conducted by a group of ten people. Similarly, the optimal range for the MRI and Lake Tahoe images were estimated using a group of three people. It appears from the table that results of DCPSO using *S_Dbw*, UFA, SOM and snob were poor. DCPSO using *V* always found a solution within the optimal range. Therefore, the remaining experiments will use *V* as the validity index. These results clearly show the efficiency of DCPSO. Table 6.4 provides samples of the resultant segmented images generated by DCPSO using *V*.

Table 6.3: Experiments on natural images

Image	Optimal range	DCPSO using *D*	DCPSO using *V*	DCPSO using *S_Dbw*	SOM	UFA	Snob
Lenna	5 to 10	10.35 ± 1.652	6.85 ± 0.477	19.3 ± 0.843	20	20	31
Mandrill	5 to 10	6.05 ± 1.658	6.25 ± 0.433	19.25 ± 0.766	20	20	42
Jet	5 to 7	3.35 ± 2.151	5.3 ± 0.459	18.05 ± 1.465	14	20	22
peppers	6 to 10	10.55 ± 1.465	6 ± 0	18.8 ± 0.872	20	20	39
MRI	4 to 8	3 ± 0	5 ± 0	17.2 ± 1.4	19	20	-
Tahoe	3 to 7	3 ± 0	6.1 ± 0.539	14.3 ± 3.018	4	20	-
Avg.		**6.05**	**5.92**	**17.82**	**16.17**	**20**	**-**

Table 6.4: Samples of segmented images resulting from DCPSO using V

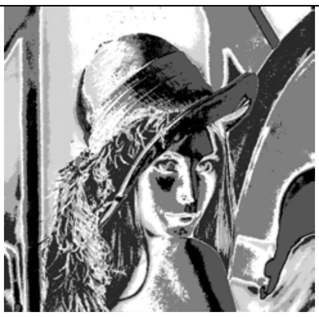
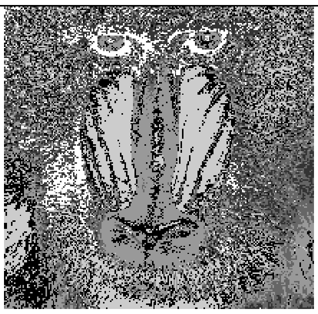
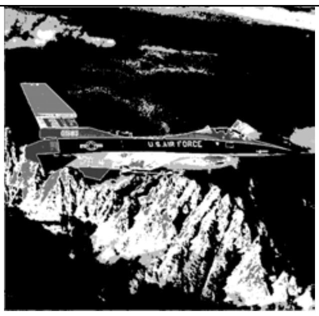
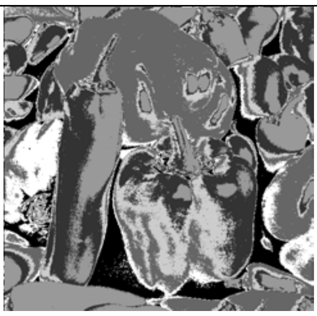
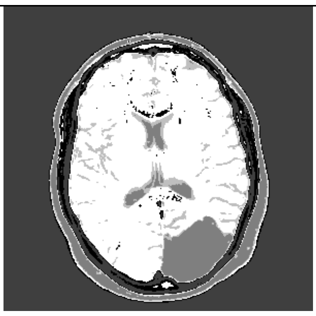
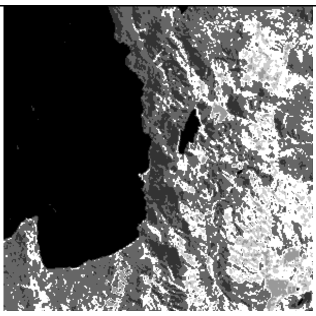
Image	Segmented image	No. of clusters
Lenna		7
Mandrill		6
Jet		5

Table 6.4: Samples of segmented images resulting from DCPSO using *V* (*continued*)

Image	Image	Image
peppers		6
MRI		5
Tahoe		6

6.2.3 Comparison with GA and RS

The previous experiments were conducted using the dynamic cluster PSO. In this section, a GA and RS version of the algorithm in Figure 6.1 (called DCGA and DCRS, respectively) are examined and compared with DCPSO. Both DCGA and DCRS used 100 individuals. For DCGA, elitism was used, keeping the fittest chromosome for the next generation. In addition, random selection has been used along with uniform crossover. The crossover probability was set to 0.8 with a mixing ratio of 0.5; a mutation probability of $1/N_c$ was used. Table 6.5 presents the results of applying DCPSO, DCGA and DCRS on the natural images. As expected, DCRS performed poorly due to its pure random search. DCGA performed comparably to DCPSO.

Table 6.5: Comparison of PSO-, GA- and RS- versions of the proposed approach

Image	Optimal range	DCPSO using V	DCGA using V	DCRS using V
Lenna	5 to 10	6.85 ± 0.477	6.45 ± 0.74	9.8 ± 1.661
Mandrill	5 to 10	6.25 ± 0.433	6.05 ± 0.589	8.75 ± 2.095
Jet	5 to 7	5.3 ± 0.459	5.3 ± 0.557	11.05 ± 1.627
peppers	6 to 10	6 ± 0	6.05 ± 0.218	10.55 ± 1.532
MRI	4 to 8	5 ± 0	5.5 ± 0.742	8.1 ± 1.179
Tahoe	3 to 7	6.1 ± 0.539	6.1 ± 0.831	9.25 ± 1.479
Avg.		**5.92**	**5.91**	**9.58**

6.2.4 Swarm Size

Reducing the swarm size (or population size in case of GA) from 100 to 20 particles (or GA chromosomes) did not generally affect the performance of either DCPSO or DCGA as illustrated in Table 6.6. However, comparing Table 6.5 and Table 6.6 it seems that on average less clusters are formed with less particles/chromosomes. In general, the computational requirements of DCPSO and DCGA can be reduced significantly without affecting the performance of DCPSO and DCGA.

Table 6.6: Comparison of PSO- and GA-versions of the proposed approach using a swarm size s = 20

Image	Optimal range	DCPSO using V	DCGA using V
Lenna	5 to 10	6.5 ± 0.806	6.4 ± 0.8
Mandrill	5 to 10	6.15 ± 0.357	5.85 ± 0.476
Jet	5 to 7	5.3 ± 0.458	5.35 ± 0.477
peppers	6 to 10	6.05 ± 0.218	6 ± 0
MRI	4 to 8	5.2 ± 0.4	5.15 ± 0.357
Tahoe	3 to 7	6.05 ± 0.384	6.2 ± 0.4
Avg.		**5.875**	**5.825**

6.2.5 The Termination Criteria

Given that all parameters are fixed at the values given in section 6.2.4, the influence of the termination criteria were evaluated for the natural images. The termination criterion for step 6 in the algorithm (section 6.1) is called TC_1 and for step 10 is called TC_2.

Table 6.7 and 6.8 summarize the effect of TC_1 and TC_2, respectively. In Table 6.7, TC_2 was fixed at 2. Table 6.7 shows that for the Lenna, Mandrill and MRI images all the tested values for TC_1 produced comparable results within the optimal range. For the Jet image, all the tested values for TC_1 performed comparably with TC_1=5 and TC_1=25 slightly worse than the other values. For the Peppers image, TC_1=75 and TC_1=100 performed better than other values and the results suggest that the "optimal" number of clusters in the Pepper image is 6 which seems to be a valid number. For the Tahoe image, all the test values for TC_1 (except TC_1=75) produced comparable results within the optimal range. From the results shown in Table 6.7, it can be concluded that the performance of DCPSO is generally insensitive to TC_1's values.

In Table 6.8, TC_1 was fixed at 50. Table 6.8 shows that for the Lenna and Peppers images all the values of TC_2 (except for TC_2=2) produced the "optimal" number of clusters. For the Mandrill image, all the tested values for TC_2 produced comparable results within the optimal range. For the Jet image, all the values of TC_2 (expect TC_2=2) produced results within the optimal range. TC_2=25 and TC_2=50 suggest that the "optimal" number of clusters in the MRI image is 5. This seems to be a valid number since the Brain MRI images consist mainly of three major tissue classes: gray matter, white matter and cerebrospinal fluid [Zhang *et al*. 2001]. Furthermore, the images contain the skull and the background. For the Tahoe image, TC_2=25 and TC_2=50 produced results outside the optimal range. From the results shown in Table 6.8, it can be concluded that the performance of DCPSO is relatively insensitive to TC_2's values.

Table 6.7: Effect of termination criterion TC_1 on the DCPSO using a swarm size $s = 20$ and $TC_2 = 2$

Image	TC_1	Optimal range	DCPSO using V
Lenna	5	5 to 10	6.05 ± 0.921
	25	5 to 10	6.55 ± 0.740
	50	5 to 10	6.5 ± 0.806
	75	5 to 10	6.55 ± 0.740
	100	5 to 10	6.55 ± 0.805
Mandrill	5	5 to 10	6.05 ± 1.023
	25	5 to 10	6.1 ± 0.539
	50	5 to 10	6.15 ± 0.357
	75	5 to 10	5.95 ± 0.384
	100	5 to 10	6.05 ± 0.218
Jet	5	5 to 7	5.35 ± 0.726
	25	5 to 7	5.2 ± 0.4
	50	5 to 7	5.3 ± 0.458
	75	5 to 7	5.35 ± 0.477
	100	5 to 7	5.35 ± 0.477
Peppers	5	6 to 10	6.45 ± 1.023
	25	6 to 10	6.2 ± 0.678
	50	6 to 10	6.05 ± 0.218
	75	6 to 10	6.0 ± 0.0
	100	6 to 10	6.0 ± 0.0
MRI	5	4 to 8	5.7 ± 0.843
	25	4 to 8	5.25 ± 0.698
	50	4 to 8	5.2 ± 0.4
	75	4 to 8	5.1 ± 0.3
	100	4 to 8	5.15 ± 0.477
Tahoe	5	3 to 7	5.85 ± 0.477
	25	3 to 7	6.15 ± 0.572
	50	3 to 7	6.05 ± 0.384
	75	3 to 7	6.45 ± 0.669
	100	3 to 7	6.2 ± 0.4

Table 6.8: Effect of termination criterion TC_2 on the DCPSO using a swarm size s = 20 and TC_1= 50

Image	TC_2	Optimal range	DCPSO using V
Lenna	2	5 to 10	6.55 ± 0.740
	10	5 to 10	7.0 ± 0.0
	25	5 to 10	7.0 ± 0.0
	50	5 to 10	7.0 ± 0.0
Mandrill	2	5 to 10	6.1 ± 0.539
	10	5 to 10	6.25 ± 0.433
	25	5 to 10	6.15 ± 0.357
	50	5 to 10	6.15 ± 0.357
Jet	2	5 to 7	5.2 ± 0.4
	10	5 to 7	5.6 ± 0.49
	25	5 to 7	5.6 ± 0.49
	50	5 to 7	5.8 ± 0.4
Peppers	2	6 to 10	6.2 ± 0.678
	10	6 to 10	6.0 ± 0.0
	25	6 to 10	6.0 ± 0.0
	50	6 to 10	6.0 ± 0.0
MRI	2	4 to 8	5.25 ± 0.698
	10	4 to 8	5.05 ± 0.218
	25	4 to 8	5.0 ± 0.0
	50	4 to 8	5.0 ± 0.0
Tahoe	2	3 to 7	6.15 ± 0.572
	10	3 to 7	6.4 ± 0.49
	25	3 to 7	6.75 ± 0.829
	50	3 to 7	6.85 ± 0.792

6.2.6 p_{ini} and N_c

Given that all parameters are fixed at the values given in section 6.2.4, the influence of p_{ini} was evaluated for the natural images. The results are summarized in Table 6.9. Studying the results, it can be concluded that the performance of DCPSO is generally insensitive to the value of p_{ini}.

Table 6.9: Effect of p_{ini} on the DCPSO using a swarm size s = 20

Image	p_{ini}	Optimal range	DCPSO using V
Lenna	0.25	5 to 10	6.7 ± 0.64
	0.5	5 to 10	6.5 ± 0.742
	0.75	5 to 10	6.5 ± 0.806
	0.9	5 to 10	6.65 ± 0.726
Mandrill	0.25	5 to 10	6.05 ± 0.218
	0.5	5 to 10	6.05 ± 0.21
	0.75	5 to 10	6.15 ± 0.357
	0.9	5 to 10	6.1 ± 0.539
Jet	0.25	5 to 7	5.5 ± 0.592
	0.5	5 to 7	5.3 ± 0.458
	0.75	5 to 7	5.3 ± 0.458
	0.9	5 to 7	5.3 ± 0.458
Peppers	0.25	6 to 10	6.0 ± 0.0
	0.5	6 to 10	6.0 ± 0.0
	0.75	6 to 10	6.05 ± 0.218
	0.9	6 to 10	6.0 ± 0.0
MRI	0.25	4 to 8	5.3 ± 0.781
	0.5	4 to 8	5.35 ± 0.726
	0.75	4 to 8	5.2 ± 0.4
	0.9	4 to 8	5.3 ± 0.9
Tahoe	0.25	3 to 7	6.2 ± 0.51
	0.5	3 to 7	6.35 ± 0.477
	0.75	3 to 7	6.05 ± 0.384
	0.9	3 to 7	6.05 ± 0.497

Given that all parameters are fixed at the values given in section 6.2.4, the influence of N_c was evaluated for the natural images. The results are summarized in Table 6.10. Studying the results, it appears that using N_c = 10 generally results in choosing the lower bound of the optimal range. However, using N_c = 50 tends to overfit the data by producing results outside the optimal range. Table 6.10 shows that, N_c = 20 generates the best results for the natural images. Hence, it can be concluded that the performance of DCPSO is sensitive to the value of N_c.

Table 6.10: Effect of N_c on the DCPSO using a swarm size $s = 20$

Image	N_c	Optimal range	DCPSO using V
Lenna	10	5 to 10	5.4 ± 0.583
	20	5 to 10	6.5 ± 0.806
	50	5 to 10	16.8 ± 3.516
Mandrill	10	5 to 10	5.55 ± 0.497
	20	5 to 10	6.15 ± 0.357
	50	5 to 10	15.95 ± 3.57
Jet	10	5 to 7	5.05 ± 0.218
	20	5 to 7	5.3 ± 0.458
	50	5 to 7	15.35 ± 2.495
Peppers	10	6 to 10	5.9 ± 0.3
	20	6 to 10	6.05 ± 0.218
	50	6 to 10	16.7 ± 2.722
MRI	10	4 to 8	5.25 ± 0.433
	20	4 to 8	5.2 ± 0.4
	50	4 to 8	11.45 ± 3.892
Tahoe	10	3 to 7	5.35 ± 0.572
	20	3 to 7	6.05 ± 0.384
	50	3 to 7	12.9 ± 4.265

6.2.7 Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*-PSO

In this section, the effect of different models of PSO is investigated. A comparison is made between *gbest*-, *lbest*- and *lbest-to-gbest*-PSO (which has been used in the above experiments) using a swarm size of 20 particles. For *lbest*-PSO, a neighborhood size of $l = 2$ was used. Table 6.11 summarizes the result of the comparison. The results show no significant difference in performance.

Table 6.11: Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*-PSO versions of DCPSO using V (s = 20)

Image	Optimal range	*gbest*- PSO	*lbest* PSO (l=2)	*lbest-to-gbest*-PSO
Lenna	5 to 10	6.6 ± 0.735	6.55 ± 0.669	6.5 ± 0.806
Mandrill	5 to 10	6.1 ± 0.3	6.1 ± 0.539	6.15 ± 0.357
Jet	5 to 7	5.5 ± 0.5	5.25 ± 0.433	5.3 ± 0.458
peppers	6 to 10	6.15 ± 0.726	6.15 ± 0.654	6.05 ± 0.218
MRI	4 to 8	5.0 ± 0.0	5.3 ± 0.458	5.2 ± 0.4
Tahoe	3 to 7	6.25 ± 0.829	6.1 ± 0.3	6.05 ± 0.384
Avg.		**5.933**	**5.908**	**5.875**

6.2.8 Multispectral Imagery Data

To show the applicability of DCPSO to multidimensional feature spaces, DCPSO was applied to the four-channel multispectral image set of the Lake Tahoe region in the US. The four bands of the image set was already shown in Figure 4.9. Table 6.12 gives the results of applying *lbest-to-gbest* DCPSO using V on the image set. The results reported in Table 6.12 are averages and standard deviations over 10 simulations. All parameters are fixed at the values given in section 6.2.4. It appears from the table that DCPSO using V found a solution within the optimal range. The results show the efficiency of DCPSO when applied to multispectral imagery data. Figure 6.3 shows a sample of the resultant segmented image (or thematic map) generated by DCPSO using V.

Table 6.12: Applying *lbest-to-gbest* DCPSO using V ($s = 20$) on multispectral image set

Image	Optimal range	DCPSO using V
Four-bands Lake Tahoe	3 to 7	5.8 ± 0.6

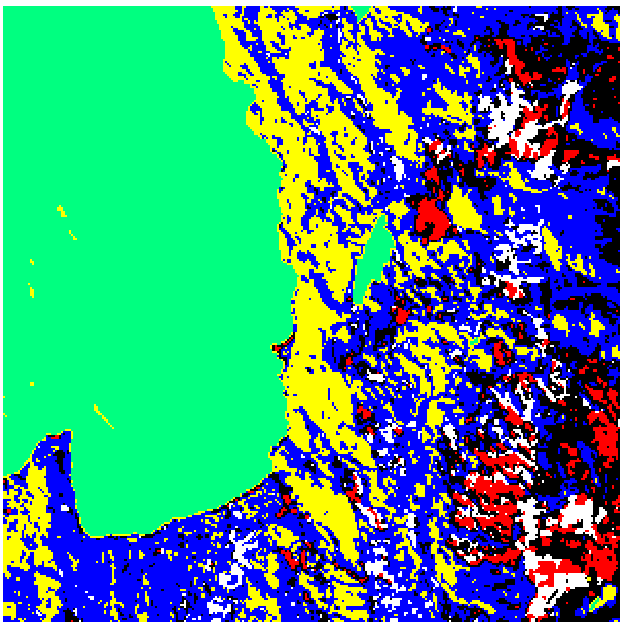


Figure 6.3: 6-Clusters thematic map obtained using DCPSO

6.3 Conclusions

This chapter presented DCPSO, a new dynamic clustering algorithm based on PSO with application to unsupervised image classification. DCPSO clusters a data set without requiring the user to specify the number of clusters in advance. This is an important feature since knowing the number of clusters in advance is often not easy. DCPSO uses a validity index to measure the quality of the resultant clustering. One of the advantages of this approach is that DCPSO can work with any validity index. In addition, the proposed approach can be used with a GA or RS. DCPSO was applied on synthetic (where the number of clusters was known in advance) as well as natural images (including MRI and satellite images), and was compared with other unsupervised clustering techniques. From these experiments it can be concluded that DCPSO, using the validity index proposed by Turi [2001], has outperformed other approaches. In general, DCPSO successfully found the optimum number of clusters on the tested images. DCPSO was then compared to both DCGA and DCRS, with DCPSO and DCGA outperforming DCRS. The influence of the different DCPSO control parameters was then investigated. The use of different PSO models (namely, *lbest*, *gbest* and *lbest-to-gbest*) was also studied. Finally, DCPSO was successfully applied to multispectral imagery data.

The next chapter applies the PSO clustering approach to two difficult problems in the fields of pattern recognition and image processing, namely, color image quantization and spectral unmixing.

Chapter 7

Applications

This chapter presents PSO-based approaches to tackle the color image quantization and spectral unmixing problems. The proposed approaches are then applied on different image sets to evaluate their performance, and they are compared with other *state-of-the-art* approaches.

7.1 A PSO-based Color Image Quantization Algorithm

A PSO-based color image quantization algorithm is developed in this section. The algorithm randomly initializes each particle in the swarm to contain K centroids (i.e. color triplets). The K-means clustering algorithm is then applied to each particle at a user-specified probability to refine the chosen centroids. Each pixel is then assigned to the cluster with the closest centroid. The PSO is then applied to refine the centroids obtained from the K-means algorithm. The proposed algorithm is then applied to commonly used images. It is shown from the conducted experiments that the proposed algorithm generally results in a significant improvement of image quality compared to other well-known approaches. The influence of different values of the algorithm control parameters is studied. Furthermore, the performance of different versions of PSO is also investigated.

7.1.1 The PSO-based Color Image Quantization (PSO-CIQ) Algorithm

This section defines the terminology used throughout section 7.1. A measure is given to quantify the quality of the resultant quantized image, after which the PSO-CIQ algorithm is introduced.

Define the following symbols:

N_p	denotes the number of image pixels
K	denotes the number of clusters (i.e. colors in the colormap)
$\mathbf{z}_p$	denotes the coordinates of pixel p
$\mathbf{m}_k$	denotes the centroid of cluster k (representing one color triple in the colormap)

In this section, the terms centroid and color triple are used interchangeably.

Measure of Quality

The most general measure of performance is the mean square error (MSE) of the quantized image using a specific colormap. The MSE was defined in equation (3.27), and is repeated here for convenience:

$$MSE = \frac{\sum_{k=1}^{K} \sum_{\forall \mathbf{z}_p \in \mathbf{C}_k} (\mathbf{z}_p - \mathbf{m}_k)^2}{N_p} \tag{7.1}$$

where $\mathbf{C}_k$ is the k^{th} cluster.

The PSO-CIQ Algorithm

In this subsection, a new post-clustering color image quantization approach is described. The proposed approach is of the class of quantization techniques that performs clustering of the color space.

In the context of color image quantization, a single particle represents a colormap (i.e. a particle consists of K cluster centroids representing RGB color triplets). The RGB coordinates in each color triple are floating-point numbers. Each particle $\boldsymbol{x}_i$ is constructed as $\boldsymbol{x}_i = (\boldsymbol{m}_{i,1},\ldots,\boldsymbol{m}_{i,k},\ldots, \boldsymbol{m}_{i,K})$ where $\boldsymbol{m}_{i,k}$ refers to the kth cluster centroid vector of the ith particle. Therefore, a swarm represents a number of candidate colormaps. The quality of each particle is measured using the MSE (defined in equation (7.1)) as follows:

$$f(\boldsymbol{x}_i) = MSE(\boldsymbol{x}_i) \tag{7.2}$$

The algorithm initializes each particle randomly from the color image to contain K centroids (i.e. color triplets). The set of K color triplets represents the colormap. The K-means clustering algorithm is then applied to each particle at a user-specified probability, p_{kmeans}. The K-means algorithm is used in order to refine the chosen colors and to reduce the search space. Each pixel is then assigned to the cluster with the closest centroid. The fitness function of each particle is calculated using equation (7.2). The PSO velocity and update equations (2.8) and (2.10) are then applied. The procedure is repeated until a stopping criterion is satisfied. The colormap of the global best particle after $t_{\max}$ iterations is chosen as the optimal result.

The PSO-CIQ algorithm is summarized in Figure 7.1.

```
1. Initialize each particle by randomly choosing $K$ color triplets from the image
2. For $t$ = 1 to $t_{\max}$
    (a) For each particle $i$
        i. Apply K-means for a few iterations with a probability $p_{\text{kmeans}}$.
        ii. For each pixel $\boldsymbol{z}_p$
            • Calculate $d^2(\boldsymbol{z}_p - \boldsymbol{m}_{i,k})$ for all clusters $\boldsymbol{C}_{i,k}$
            • Assign $\boldsymbol{z}_p$ to $\boldsymbol{C}_{i,kk}$ where
              $d^2(\boldsymbol{z}_p - \boldsymbol{m}_{i,kk}) = \min_{\forall k=1,\ldots,K} \{d^2(\boldsymbol{z}_p - \boldsymbol{m}_{i,k})\}$
        iii. Calculate the fitness, $f(\boldsymbol{x}_i)$
    (b) Find the personal best position for each particle and the global best
        solution, $\hat{\boldsymbol{y}}(t)$
    (c) Update the centroids using equations (2.8) and (2.10)
```

Figure 7.1: The PSO-CIQ algorithm

In general, the complexity of the PSO-CIQ algorithm is O($sKt_{\max}N_p$). The parameters s, K and $t_{\max}$ can be fixed in advance. Typically s, K and $t_{\max} << N_p$. Therefore, the time complexity of PSO-CIQ is O(N_p). Hence, in general the algorithm has linear time complexity in the size of a data set.

7.1.2 Experimental Results

The PSO-CIQ algorithm was applied to a set of four commonly used color images namely: *Lenna*, *mandrill*, *jet* and *peppers* (shown in Figures 7.1(a), 7.2(a), 7.3(a) and 7.4(a), respectively). The size of each image is 512 × 512 pixels. All images are quantized to 16, 32 and 64 colors.

The rest of this section is organized as follows: Section 7.1.2.1 illustrates that the PSO-CIQ can be used successfully as a color image quantization algorithm by comparing it to other well-known color image quantization approaches. Section 7.1.2.2 investigates the influence of the different PSO-CIQ control parameters. Finally, the use of different PSO models (namely, *gbest*, *lbest* and *lbest-to-gbest*) are investigated in section 7.1.2.3.

The results reported in this section are averages and standard deviations over 10 simulations. Since *lbest-to-gbest* PSO was generally the best performer in chapter 4, *lbest-to-gbest* PSO is used in this section unless otherwise specified. The PSO-CIQ parameters were initially set as follows: $p_{\text{kmeans}} = 0.1$, $s = 20$, $t_{\max} = 50$, number of K-means iterations is 10 (the effect of these values are then investigated), $w = 0.72$, $c_1 = c_2 = 1.49$ and $V_{\max} = 255$ for all the test images. These parameters were used in this section unless otherwise specified. For the GCMA [Scheunders, A Genetic 1997] a population of 20 chromosomes was used, and evolution continued for 50 generations. For the SOM, a Kohonen network of 4×4 nodes was used when quantizing an image to 16 colors, a Kohonen network of 8×4 nodes was used when quantizing an image to 32 colors, and a Kohonen network of 8×8 nodes was used when quantizing an image to 64 colors. All SOM parameters were set as in Pandya and Macy [1996]: the learning rate $\eta(t)$ was initially set to 0.9 then decreased by 0.005 until it reached

0.005, the neighborhood function $\Delta_w(t)$ was initially set to (4+4)/4 for 16 colors, (8+4)/4 for 32 colors, and (8+8)/4 for 64 colors. The neighborhood function is then decreased by 1 until it reached zero.

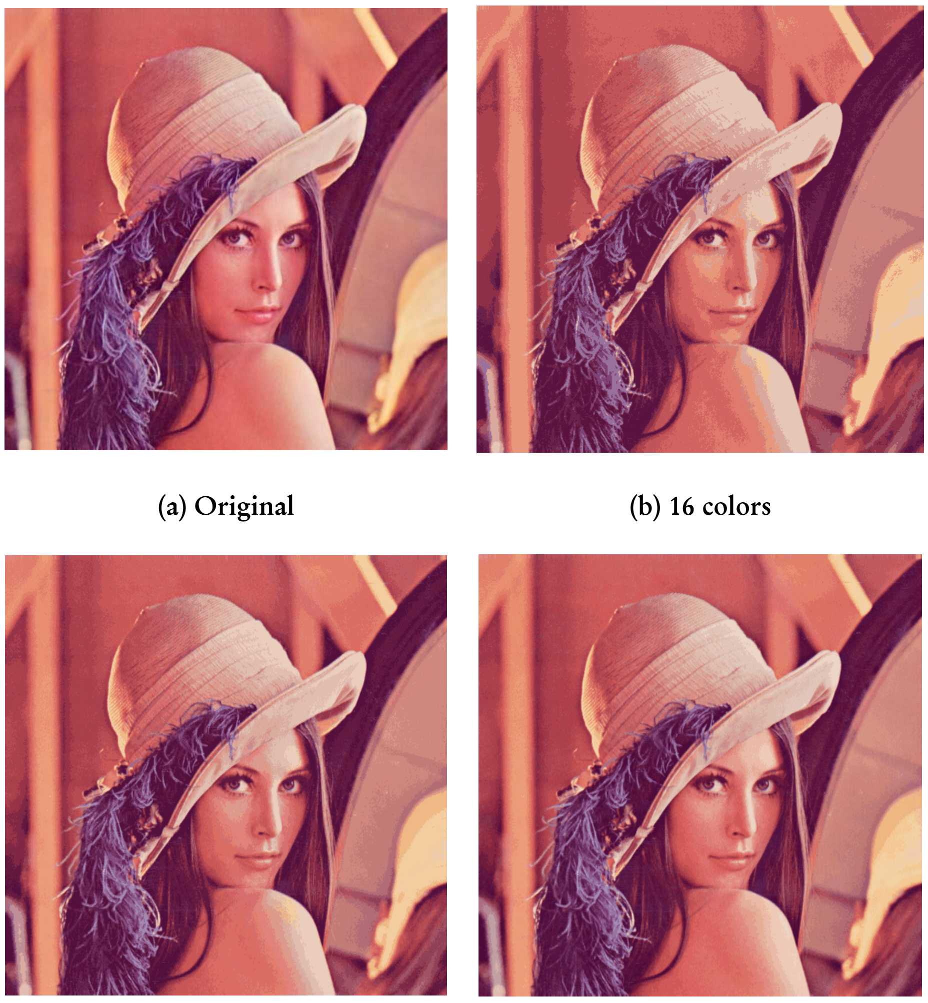
7.1.2.1 PSO-CIQ vs. Well-Known Color Image Quantization Algorithms

This section presents results to compare the performance of the PSO-CIQ algorithm with that of SOM and GCMA (both discussed in section 3.3.2) for each of the test images.

Table 7.1 summarizes the results for the four images. The results of the GCMA represent the best case over several runs and are copied from Scheunders [A Genetic 1997]. The results are compared based on the MSE measure (defined in equation 7.1). The results showed that, in general, PSO-CIQ outperformed GCMA in all the test images except for the mandrill image and the case of quantizing the Jet image to 64 colors. Furthermore, PSO-CIQ generally performed better than SOM for both Lenna and peppers images. SOM and PSO-CIQ performed comparably well when applied to the mandrill image. SOM generally performed better than PSO-CIQ when applied to the Jet image. Figures 7.2, 7.3, 7.4 and 7.5 show the visual quality of the quantized images generated by PSO-CIQ when applied to Lenna, peppers, jet and mandrill, respectively.

Table 7.1: Comparison between SOM, GCMA and PSO-CIQ

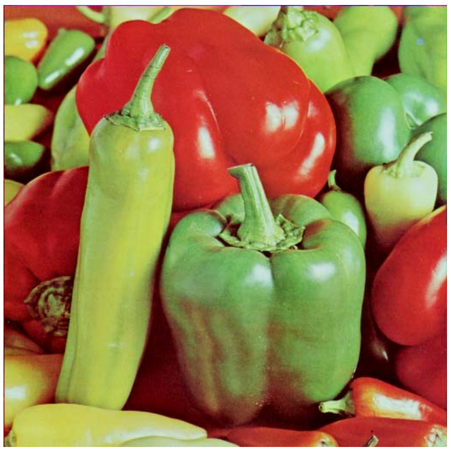
Image	*K*	**SOM**	**GCMA**	**PSO-CIQ**
Lenna	16	235.6 ± 0.490	332	210.203 ± 1.487
	32	126.400 ± 1.200	179	119.167 ± 0.449
	64	74.700 ± 0.458	113	77.846 ± 16.132
Peppers	16	425.600 ± 13.162	471	399.63 ± 2.636
	32	244.500 ± 3.854	263	232.046 ± 2.295
	64	141.600 ± 0.917	148	137.322 ± 3.376
Jet	16	121.700 ± 0.458	199	122.867 ± 2.0837
	32	65.000 ± 0.000	96	71.564 ± 6.089
	64	38.100 ± 0.539	54	56.339 ± 11.15
Mandrill	16	629.000 ± 0.775	606	630.975 ± 2.059
	32	373.600 ± 0.490	348	375.933 ± 3.42
	64	234.000 ± 0.000	213	237.331 ± 2.015

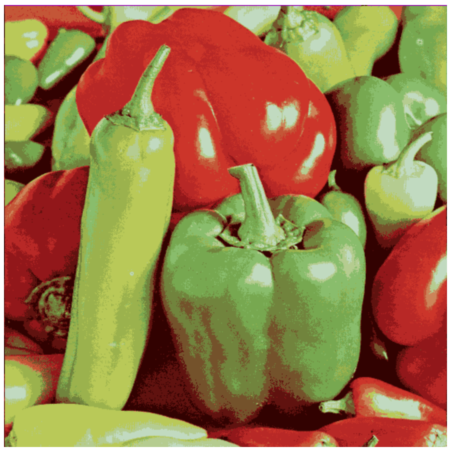


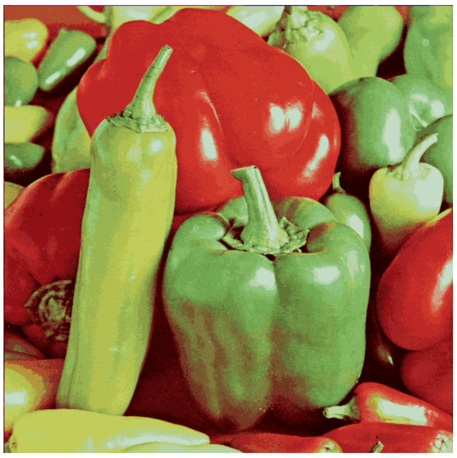
(a) Original (b) 16 colors

(c) 32 colors (d) 64 colors

Figure 7.2: Quantization results for the Lenna image using PSO-CIQ


(a) Original


(b) 16 colors


(c) 32 colors

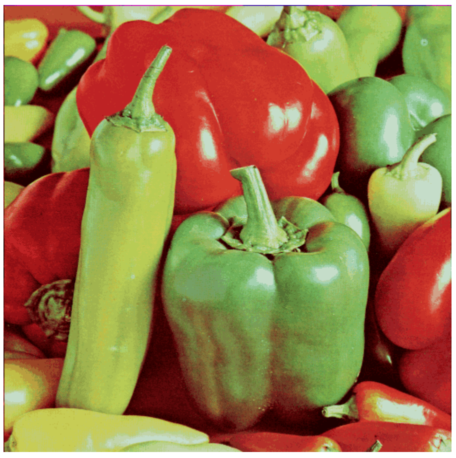
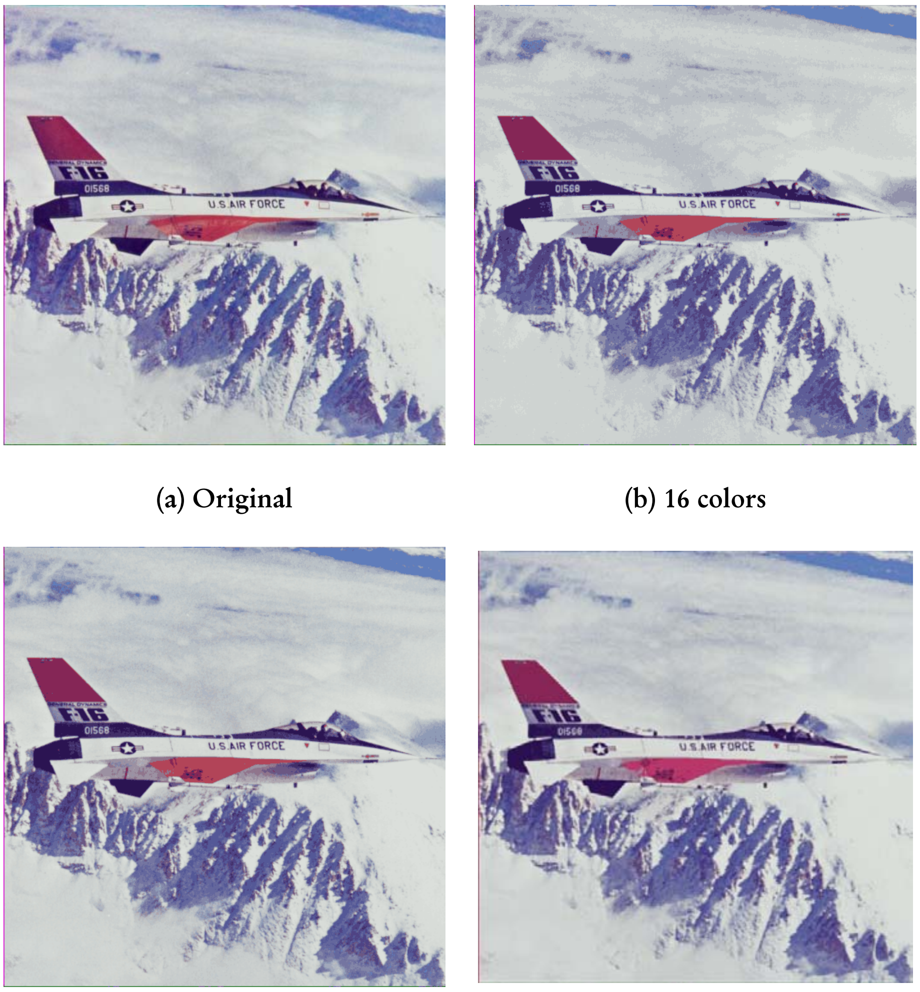

(d) 64 colors

Figure 7.3: Quantization results for the peppers image using PSO-CIQ



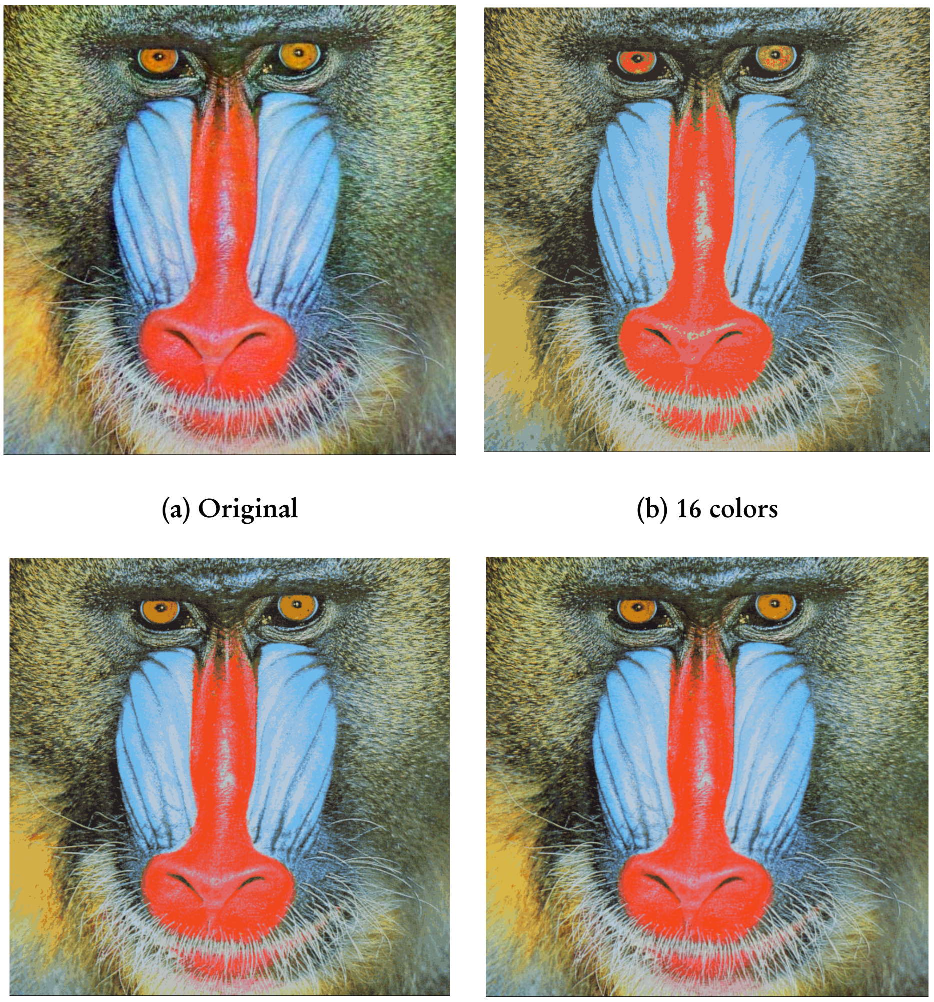
(a) Original

(b) 16 colors

(c) 32 colors

(d) 64 colors

Figure 7.4: Quantization results for the jet image using PSO-CIQ



(a) Original

(b) 16 colors

(c) 32 colors

(d) 64 colors

Figure 7.5: Quantization results for the mandrill image using PSO-CIQ

7.1.2.2 Influence of PSO-CIQ Parameters

The PSO-CIQ algorithm has a number of parameters that have an influence on the performance of the algorithm. These parameters include V_{max}, the swarm size, the number of PSO iterations, p_{kmeans} and the number of K-means iterations. This section investigates the influence of different values of these parameters using the Lenna image when quantized to 16 colors.

Velocity Clamping

Table 7.2 shows that using $V_{max} = 5$ or $V_{max} = 255$ generally produces comparable results.

Table 7.2: Effect of V_{max} on the performance of PSO-CIQ using Lenna image (16 colors)

	MSE
V_{max}=5	209.338 ± 0.402
V_{max}=255	210.203 ± 1.487

Swarm Size

Increasing the swarm size from 20 to 50 particles slightly improves the performance of the PSO-CIQ algorithm as shown in Table 7.3. Similarly, increasing the swarm size from 50 to 100 particles slightly improves the performance of the PSO-CIQ algorithm. On the other hand, reducing the swarm size from 20 to 10 particles significantly reduces the efficiency of the PSO-CIQ algorithm. The rationale behind these results is that increasing the number of particles increases diversity, thereby limiting the effects of initial conditions and reducing the possibility of being trapped in local minima.

Table 7.3: Effect of the swarm size on the performance of PSO-CIQ using Lenna image (16 colors)

	MSE
s = 10	212.196 ± 2.458
s = 20	210.203 ± 1.487
s = 50	210.06 ± 1.11
s = 100	209.468 ± 0.703

Number of PSO iterations

Increasing the number of PSO iterations, $t_{\max}$, from 50 to 100 slightly improves the performance of the PSO-CIQ algorithm as shown in Table 7.4. Similarly, increasing $t_{\max}$ from 100 to 150 slightly improves the performance of the PSO-CIQ algorithm. Therefore, it can be concluded that increasing $t_{\max}$ generally improves the performance of the PSO-CIQ algorithm.

Table 7.4: Effect of the number of PSO iterations on the performance of PSO-CIQ using Lenna image (16 colors)

	MSE
$t_{\max}$ = 50	210.203 ± 1.487
$t_{\max}$ = 100	209.412 ± 0.531
$t_{\max}$ = 150	208.866 ± 0.22

p_{kmeans}

Applying the K-means clustering algorithm to a larger set of particles is expected to improve the performance of the PSO-CIQ algorithm. The rationale behind this expectation is the fact that the K-means algorithm generally reduces the search space and refines the chosen colors. This expectation is verified by the results shown in Table 7.5 which shows that increasing the value of p_{kmeans} generally improves the performance of the PSO-CIQ algorithm. However, as a trade-off, increasing the value of p_{kmeans} will increase the computational requirements of the PSO-CIQ algorithm.

Table 7.5: Effect of p_{kmeans} on the performance of PSO-CIQ using Lenna image (16 colors)

	MSE
$p_{\text{kmeans}} = 0.1$	210.203 ± 1.487
$p_{\text{kmeans}} = 0.25$	209.238 ± 0.74
$p_{\text{kmeans}} = 0.5$	209.045 ± 0.594
$p_{\text{kmeans}} = 0.9$	208.886 ± 0.207

Number of K-means iterations

Reducing the number of K-means iterations from 10 to 5 degrades the performance of the PSO-CIQ as shown in Table 7.6. On the other hand, increasing the number of K-means iterations from 10 to 50 significantly improves the performance of the PSO-CIQ as shown in Table 7.6. These results suggest that increasing the number of K-means iterations improves the performance of the PSO-CIQ. However, when the number of K-means iterations was reduced to 5 iterations but at the same time p_{kmeans} was increased from 0.1 to 0.5 the generated MSE was 210.315 ± 1.563 which is significantly better than the corresponding result in Table 7.6. This result suggests that the number of K-means iterations can be reduced without affecting the performance of PSO-CIQ given that the p_{kmeans} is increased.

Table 7.6: Effect of the number of K-means iterations on the performance of PSO-CIQ using Lenna image (16 colors)

No. of K-means iterations	**MSE**
5	212.627 ± 3.7
10	210.203 ± 1.487
50	208.791 ± 0.111

7.1.2.3 Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*-PSO

In this section, the effect of different models of PSO is investigated using the Lenna image when quantized to 16 colors. A comparison is made between *gbest*-, *lbest*- and *lbest-to-gbest*-PSO (which has been used in the above experiments) using a swarm size of 20 particles. For *lbest*-PSO, a neighborhood size of $l = 2$ was used. Table 7.7 shows the result of the comparison. The results show no significant difference in performance.

Table 7.7: Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*- PSO versions of PSO-CIQ using Lenna image (16 colors)

	MSE
gbest PSO	209.841 ± 0.951
lbest PSO	210.366 ± 1.846
lbest-to-gbest PSO	210.203 ± 1.487

7.2 A PSO-based End-Member Selection Method for Spectral Unmixing of Multispectral Satellite Images

An end-member selection method for spectral unmixing that is based on PSO is developed in this section. The algorithm uses the K-means clustering algorithm and a method of dynamic selection of end-members subsets to find the appropriate set of end-members for a given set of multispectral images. The proposed algorithm has been successfully applied to test image sets from various platforms such as LANDSAT 5 MSS and NOAA's AVHRR. The experimental results of the proposed algorithm are encouraging. The influence of different values of the algorithm control parameters on performance is studied. Furthermore, the performance of different versions of PSO is also investigated.

7.2.1 The PSO-based End-Member Selection (PSO-EMS) Algorithm

This section introduces the PSO-EMS algorithm by first presenting a measure to quantify the quality of a spectral unmixing algorithm, after which the PSO-EMS algorithm is shown.

Measure of Quality

To measure the quality of a spectral unmixing algorithm, the root mean square (RMS) residual error can be used, defined as follows:

$$E = \sum_{j=1}^{N_b} \sqrt{MS_j} \tag{7.3}$$

where

$$\boldsymbol{MS} = \frac{\sum_{p=1}^{N_p} (\boldsymbol{z}_p - \boldsymbol{EM}.\boldsymbol{f})^2}{N_p}$$

where N_b is the number of spectral bands, N_p is the number of pixels in the image and $\boldsymbol{EM}.\boldsymbol{f}$ is defined in equation (3.28).

The PSO-EMS Algorithm

In the context of spectral unmixing, a single particle represents N_m end-members. That is, each particle $\boldsymbol{x}_i$ is constructed as $\boldsymbol{x}_i = (\boldsymbol{em}_{i,1},\ldots,\boldsymbol{em}_{i,k},\ldots, \boldsymbol{em}_{i,N_m})$ where $\boldsymbol{em}_{i,k}$ refers to the kth end-member vector of the ith particle. Therefore, a swarm represents a number of candidate end-members. The quality of each particle is measured using the RMS residual error (defined in equation 7.3) as follows:

$$f(\boldsymbol{x}_i) = E(\boldsymbol{x}_i) \tag{7.4}$$

The algorithm randomly initializes each particle from the multispectral image set to contain N_m end-members. The K-means clustering algorithm is then applied to each particle at a user-specified probability, p_{kmeans}. The K-means algorithm is used in order to refine the chosen end-members and to reduce the search space. Then for each particle i, the N_m end-members of the particle form the pool of available candidate end-members for the subsequent spectral unmixing procedure. Maselli's approach (refer to section 3.4.2) is used to dynamically select the N_e optimum end-member subsets from the pool of N_m end-members. Each pixel vector is then spectrally decomposed as a linear combination of its optimum subset of end-members. The RMS residual error for particle *i* is then calculated. The PSO velocity and update equations (2.8) and (2.10) are then applied. The procedure is repeated until a stopping criterion is satisfied. The N_m end-members of the best particle are used to generate the abundance images.

The Generation of Abundance Images

For each species represented by an end-member, the ensemble of all fractional components forms a concentration map (i.e. an abundance map). The fractional concentration maps are then optimally mapped to an 8-bit integer format for display and storage purposes. This is done using the following non-linear mapping function [Saghri *et al*. 2000]:

$$\Omega = \frac{255(f^{\exp} - (f_{\min})^{\exp})}{(f_{\max})^{\exp} - (f_{\min})^{\exp}} + 0.5 \qquad (7.5)$$

where:

Ω	is the mapped integer fractional component in the range of $0 \le \Omega \le 255$
f	is the fractional component
$f_{\min}$	is the minimum fractional component
$f_{\max}$	is the maximum fractional component
exp	is the floating-point exponent parameter in the range of $0 \le$ exp $\le$ 1.0. In this chapter, *exp* is set to 0.6 for the abundance images as suggested by Saghri *et al*. [2000].

The PSO-EMS algorithm is summarized Figure 7.6.

In general, the complexity of the PSO-EMS algorithm is $O(st_{\max}N_p)$. The parameters s and $t_{\max}$ can be fixed in advance. Typically s and $t_{\max} << N_p$. Therefore, the time complexity of PSO-EMS is $O(N_p)$. Hence, in general the algorithm has linear time complexity in the size of a data set.

```
1. Initialize each particle to contain $N_m$ randomly selected end-members
2. For $t = 1$ to $t_{\max}$
    (a) For each particle $i$
        i. Apply K-means for a few iterations with a probability
           $p_{\text{kmeans}}$.
        ii. For each pixel $\boldsymbol{z}_p$
            • Find the $N_e$ optimum end-member subset
            • Apply linear spectral unmixing using equation (3.28)
        iii. Calculate the fitness, $f(\boldsymbol{x}_i)$
    (b) Find the personal best position for each particle and the global best
        solution, $\hat{\boldsymbol{y}}(t)$
    (c) Update the end-members using equations (2.8) and (2.10)
3. Generate the abundance images using the $N_m$ end-members of particle $\hat{\boldsymbol{y}}(t)$
```

Figure 7.6: The PSO-EMS algorithm

7.2.2 Experimental Results

The PSO-EMS algorithm has been applied to two types of imagery data, namely LANDSAT 5 MSS (79 m GSD) and NOAA's AVHRR (1.1 km GSD) images. These image sets have been selected to test the algorithms on a variety of platforms with a relatively large GSD which represent good candidates for spectral unmixing in order to get sub-pixel resolution. The two image sets are described below:

LANDSAT 5 MSS: Figure 4.9 shows the four-channel multispectral test image set of the Lake Tahoe region in the US. Each channel is comprised of a 300 × 300, 8-bit per pixel (remapped from the original 6 bit) image and corresponds to a GSD of 79 m. The test image set is one of the North American Landscape Characterization (NALC) Landsat multispectral scanner data sets obtained from the U.S. Geological Survey (USGS). The result of a preliminary principal component study of this data set indicates that its intrinsic true spectral dimension N_e is 3. As in Saghri *et al.* [2002], a total of six end-members were obtained from the data set (i.e. $N_m = 6$).

NOAA's AVHRR: Figure 7.7 shows the five-channel multispectral test image set of an almost cloud-free territory of the entire United Kingdom (UK). This image set was obtained from the University of Dundee Satellite Receiving Station. Each channel (one visible, one near-infra red and three in the thermal range) is comprised of a 847 × 1009, 10-bit per pixel (1024 gray levels) image and corresponds to a GSD of 1.1 km. The result of a preliminary principal component study of this data set indicates that its intrinsic true spectral dimension N_e is 3. As in Saghri *et al.* [2000], a total of eight end-members were obtained from the data set (i.e. $N_m = 8$).

The rest of this subsection is organized as follows: Section 7.2.2.1 illustrates that the PSO-EMS can be used successfully as an end-member selection method by comparing it to the end-member selection method proposed by Saghri *et al.* [2000] (discussed in section 3.4.2), which is referred to in this chapter as ISO-UNMIX. The time complexity of ISO-UNMIX is $O(N_p)$. Saghri *et al.* [2000] showed that ISO-

UNMIX performed very well compared to other popular spectral unmixing methods. Section 7.2.2.2 investigates the influence of the different PSO-EMS control parameters. Finally, the use of different PSO models (namely, *gbest*, *lbest* and *lbest-to-gbest*) was investigated in section 7.2.2.3.

The results reported in this section are averages and standard deviations over 10 simulations. Since *lbest-to-gbest* PSO was generally the best performer in chapter 4, *lbest-to-gbest* PSO is used in this section unless otherwise specified. The PSO-EMS parameters were initially set as follows: $p_{\text{kmeans}} = 0.1$, $s = 20$, $t_{\max} = 100$, the number of K-means iterations is 10 (the effect of these values are then investigated), w =0.72, $c_1 = c_2 = 1.49$ and $V_{\max} = 255$ for the Lake Tahoe image set, while $V_{\max} = 1023$ for the UK image set. These parameters are used in this section unless otherwise specified.

7.2.2.1 PSO-EMS vs. ISO_UNMIX

This section presents results to compare the performance of the PSO-EMS algorithm with that of the ISO-UNMIX algorithm for each of the test image sets.

Table 7.8 summarizes the results for the two image sets. The results are compared based on the RMS residual error defined in equation (7.3). The results showed that, for both image sets, PSO-EMS performed significantly better than the ISO-UNMIX in terms of the RMS residual error. Figures 7.8 and 7.9 show the abundance images generated from ISO-UNMIX and PSO-EMS, respectively, when applied to the Lake Tahoe image set. In addition, Figures 7.10 and 7.11 show the abundance images generated from ISO-UNMIX and PSO-EMS, respectively, when applied to the UK image set. For display purposes the fractional species concentrations were mapped to 8-bits per pixels abundance images.

Table 7.8: Comparison between ISO-UNMIX and PSO-EMS

Image		**RMS**
LANDSAT 5 MSS	**ISO_UNMIX**	0.491837
	PSO-EMS	0.462197 ± 0.012074
NOAA's AVHRR	**ISO_UNMIX**	3.725979
	PSO-EMS	3.510287 ± 0.045442

7.2.2.2 Influence of PSO-EMS Parameters

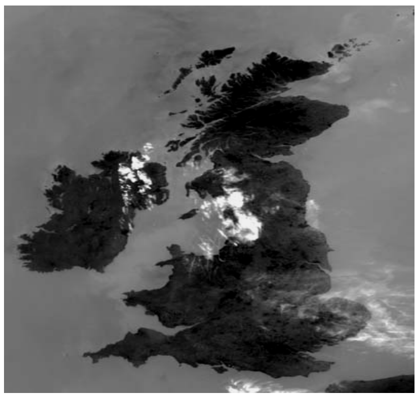
The PSO-EMS algorithm has a number of parameters that have an influence on the performance of the algorithm. These parameters include V_{max}, the swarm size, the number of PSO iterations, p_{kmeans} and the number of K-means iterations. This section investigates the influence of different values of these parameters using the Lake Tahoe image set.

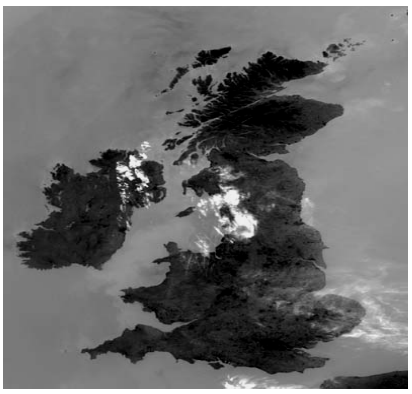
Velocity Clamping

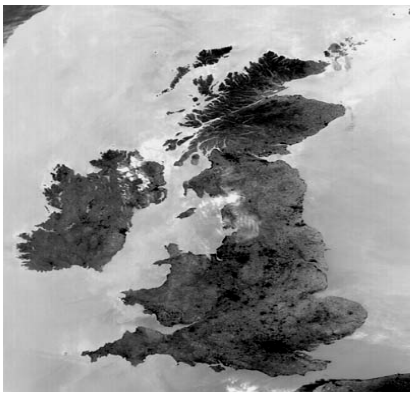
Table 7.9 shows that using $V_{max} = 5$ or $V_{max} = 255$ generally produces comparable results. However, the standard deviation in the case of $V_{max} = 5$ is smaller than the standard deviation in the case of $V_{max} = 255$. Hence, using $V_{max} = 5$ generates more stable results than using $V_{max} = 255$.

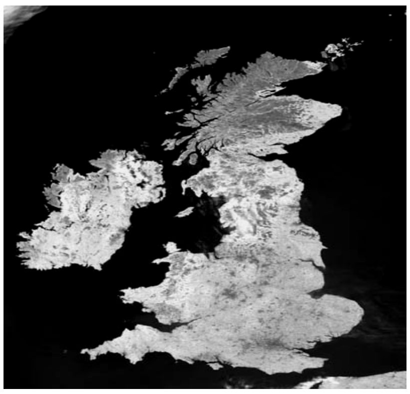
Table 7.9: Effect of V_{max} on the performance of PSO-EMS using Lake Tahoe image set

	RMS
V_{max}=5	0.469706 ± 0.000456
V_{max}=255	0.462197 ± 0.012074


(a) Band 1


(b) Band 2


(c) Band 3


(d) Band 4

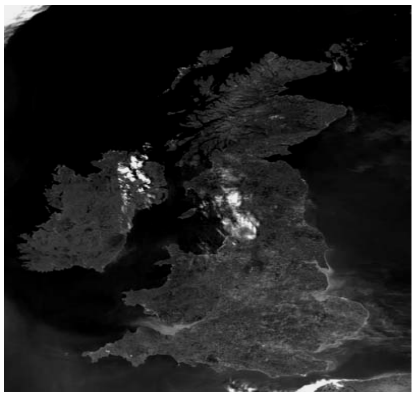

(e) Band 5

Figure 7.7: AVHRR Image of UK, Size: 847x1009 , 5 bands, 10-bits per pixel

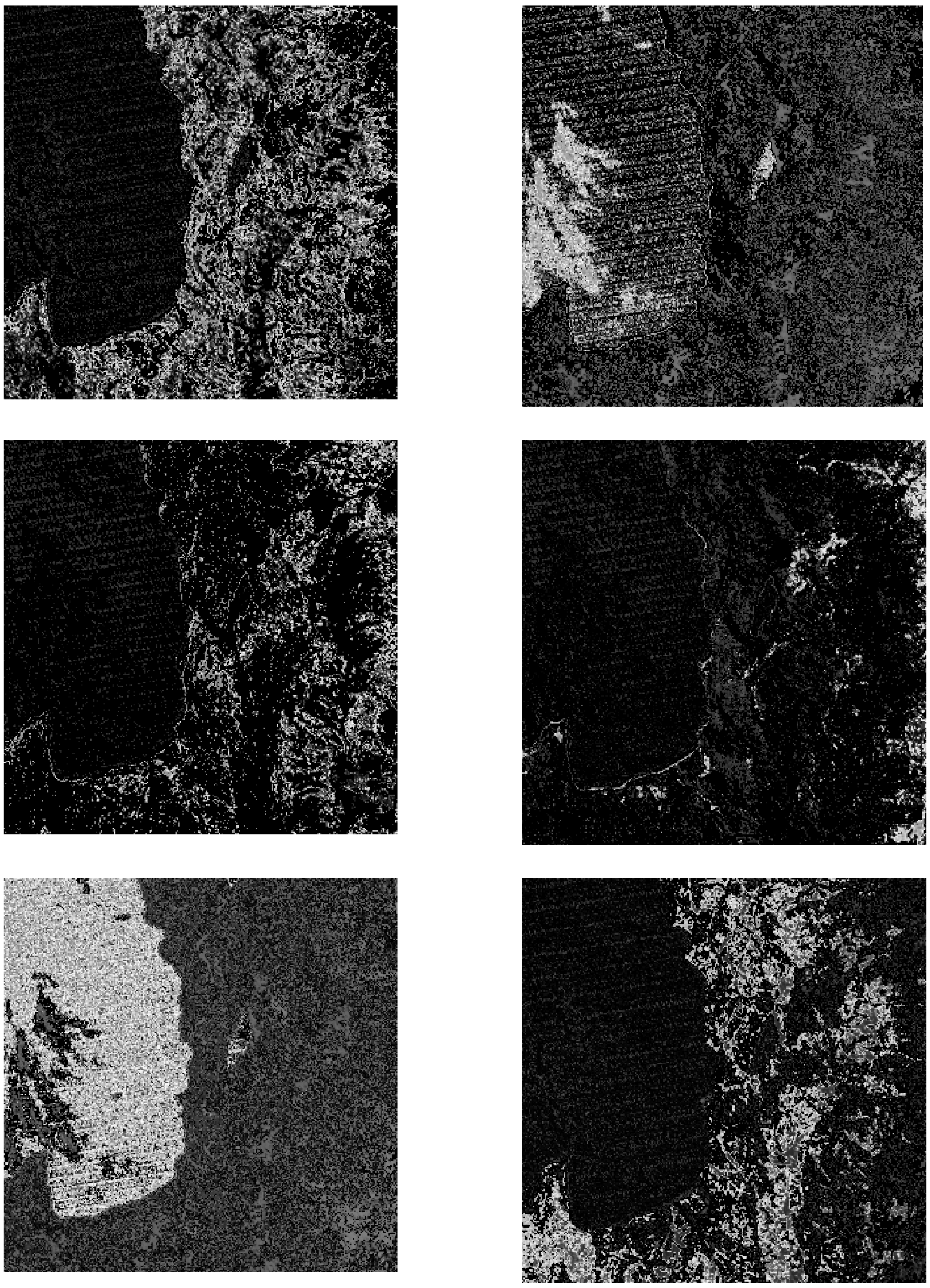


Figure 7.8: Species concentration maps resulting from the application of ISO-UNMIX to unmix the Lake Tahoe test image set

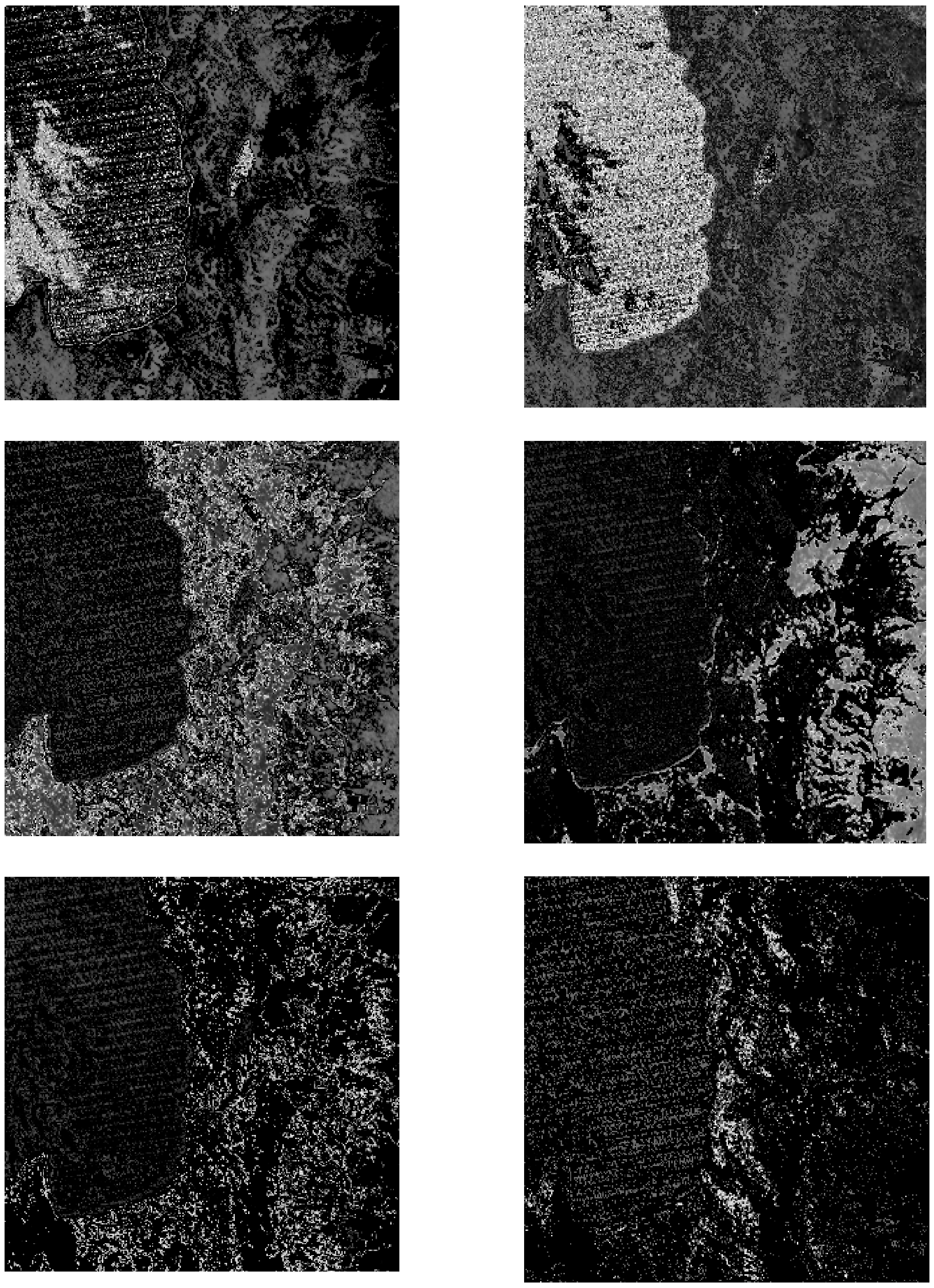


Figure 7.9: Species concentration maps resulting from the application of PSO-EMS to unmix the Lake Tahoe test image set

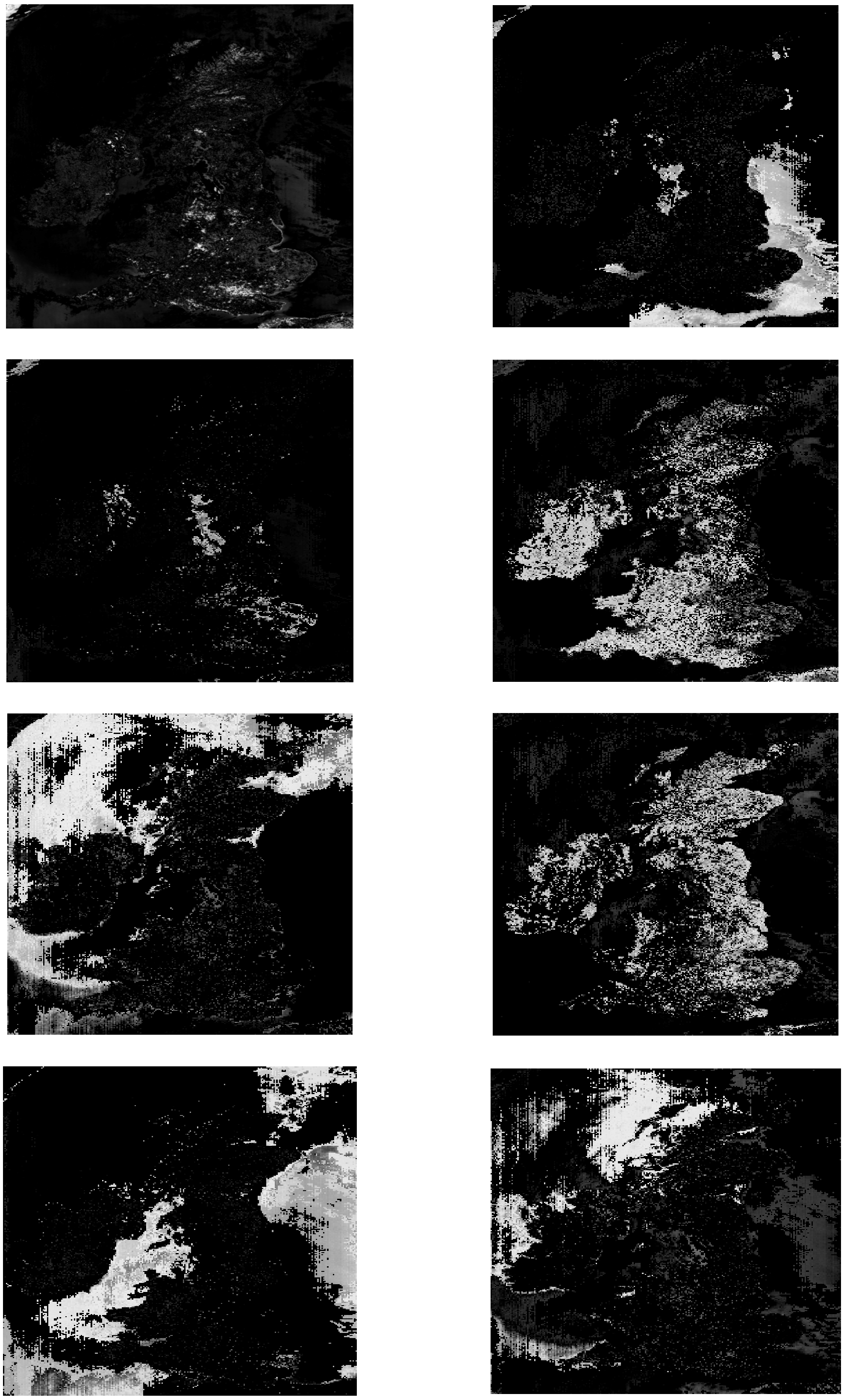


Figure 7.10: Species concentration maps resulting from the application of ISO-UNMIX to unmix the UK test image set

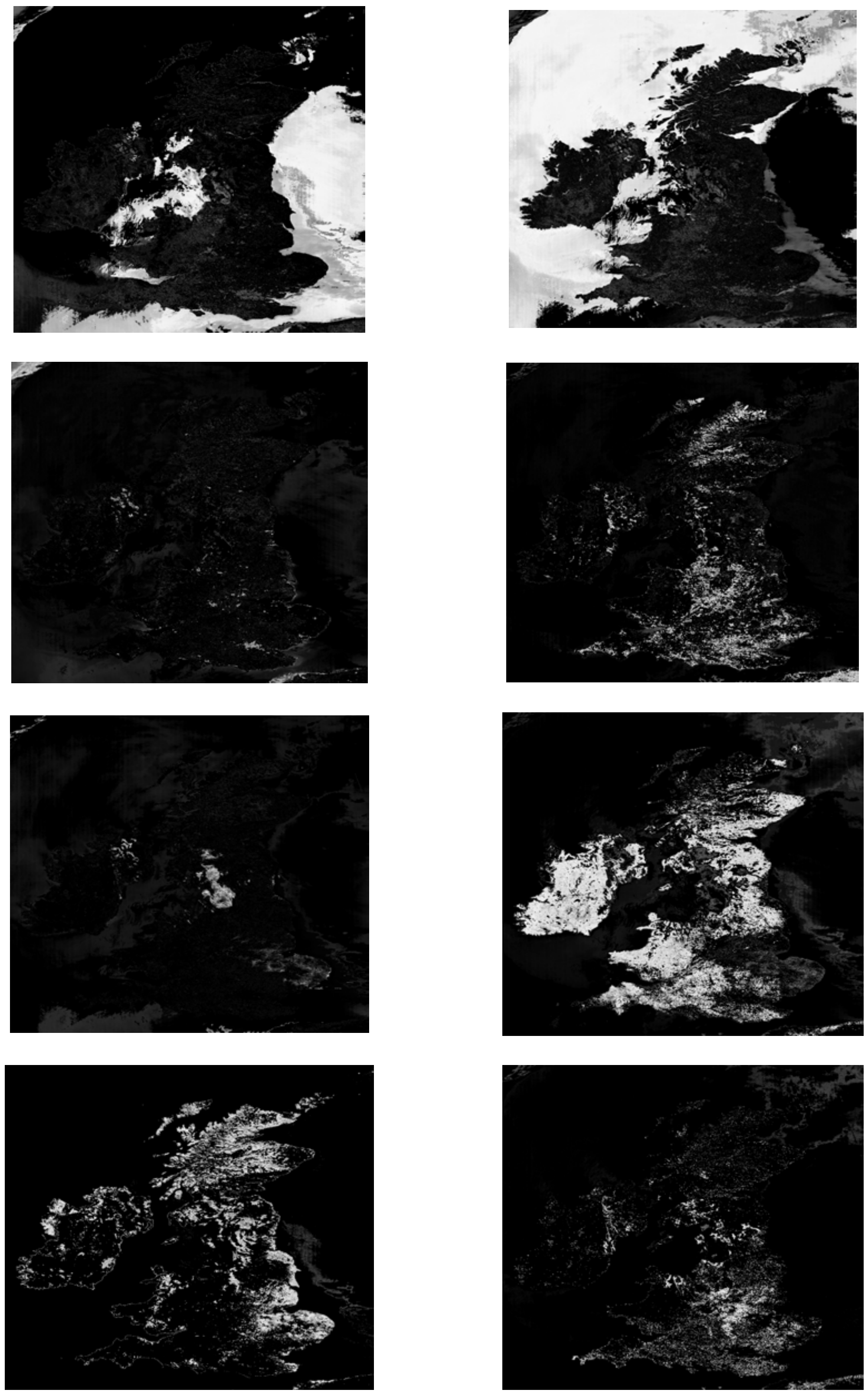


Figure 7.11: Species concentration maps resulting from the application of PSO-EMS to unmix the UK test image set

Swarm Size

Increasing the swarm size from 20 to 50 particles improves the performance of the PSO-EMS algorithm as shown in Table 7.10. On the other hand, reducing the swarm size from 20 to 10 particles significantly reduces the efficiency of the PSO-EMS algorithm. The rationale behind these results is that increasing the number of particles increases diversity, thereby limiting the effects of initial conditions and reducing the possibility of being trapped in local minima.

Table 7.10: Effect of the swarm size on the performance of PSO-EMS using Lake Tahoe image set

	RMS
$s = 10$	0.468706 ± 0.004753
$s = 20$	0.462197 ± 0.012074
$s = 50$	0.459195 ± 0.009389

Number of PSO iterations

Reducing the number of PSO iterations, $t_{\max}$, from 100 to 50 did not reduce the performance of the PSO-EMS algorithm as shown in Table 7.11. Similarly, increasing $t_{\max}$ from 100 to 150, did not significantly improve the performance of the PSO-EMS.

Table 7.11: Effect of the number of PSO iterations on the performance of PSO-EMS using Lake Tahoe image set

	RMS
$t_{\max} = 50$	0.468041 ± 0.004735
$t_{\max} = 100$	0.462197 ± 0.012074
$t_{\max} = 150$	0.465614 ± 0.00739

p_{kmeans}

Applying the K-means clustering algorithm to a larger set of particles is expected to improve the performance of the PSO-EMS algorithm. The rationale behind this expectation is the fact that the K-means algorithm generally reduces the search space and refines the end-members. This expectation is verified by the results shown in Table 7.12 which shows that increasing the value of p_{kmeans} significantly improves the performance of the PSO-EMS algorithm. However, as a trade-off, increasing the value of p_{kmeans} will increase the computational requirements of the PSO-EMS algorithm.

Table 7.12: Effect of p_{kmeans} on the performance of PSO-EMS using Lake Tahoe image set

	RMS
$p_{\text{kmeans}} = 0.1$	0.462197 ± 0.012074
$p_{\text{kmeans}} = 0.25$	0.460776 ± 0.009120
$p_{\text{kmeans}} = 0.5$	0.454029 ± 0.007051
$p_{\text{kmeans}} = 0.9$	0.445367 ± 0.012339

Number of K-means iterations

Reducing number of K-means iterations from 10 to 5 degrades the performance of the PSO-EMS as shown in Table 7.13. On the other hand, increasing the number of K-means iterations from 10 to 50 did not improve the performance of the PSO-EMS as shown in Table 7.13. These results suggest that using 10 iterations of K-means is a good choice for the Lake Tahoe image set. However, when the number of K-means iterations was reduced to 5 iterations but at the same time p_{kmeans} was increased from 0.1 to 0.5 the generated RMS was equal to 0.458149 ± 0.004554 which is significantly better than the results in Table 7.13. This result suggests that the number of K-means iterations can be reduced without affecting the performance of PSO-EMS given that the p_{kmeans} is increased.

Table 7.13: Effect of the number of K-means iterations on the performance of PSO-EMS using Lake Tahoe image set

No. of K-means iterations	**RMS**
5	0.468407 ± 0.004212
10	0.462197 ± 0.012074
50	0.466708 ± 0.004524

7.2.2.3 Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*-PSO

In this section, the effect of different models of PSO is investigated using the Lake Tahoe image set. A comparison is made between *gbest*-, *lbest*- and *lbest-to-gbest*-PSO (which has been used in the above experiments) using a swarm size of 20 particles. For *lbest* PSO, a neighborhood size of $l = 2$ was used. Table 7.14 summarizes the result of the comparison. The results show no significant difference in performance. However, the standard deviation in the case of *lbest-to-gbest* PSO is the largest. Hence, using *lbest-to-gbest* PSO generates the least stable result.

Table 7.14: Comparison of *gbest*-, *lbest*- and *lbest-to-gbest*- PSO versions of PSO-EMS using Lake Tahoe image set

	RMS
gbest PSO	0.465809 ± 0.006562
lbest PSO	0.465020 ± 0.004942
lbest-to-gbest PSO	0.462197 ± 0.012074

7.3 Conclusions

This chapter addressed two difficult problems in the field of pattern recognition and image processing. The two problems are color image quantization and spectral unmixing. First, the chapter presented a PSO-based color image quantization algorithm (PSO-CIQ). The PSO-CIQ algorithm was compared against other well-known color image quantization techniques. In general, the PSO-CIQ performed better than the other techniques when applied to a set of commonly used images. The effects of different PSO-CIQ control parameters were studied. The performance of different versions of PSO was then investigated.

The chapter then presented a new spectral unmixing approach using PSO (PSO-EMS). The PSO-EMS algorithm has as objective to determine the appropriate set of end-members for a given multispectral image set. The PSO-EMS algorithm was compared against a relatively recent end-member selection method which was proposed by Saghri *et al.* [2000]. The PSO-EMS algorithm produced better results when applied to test image sets from various platforms such as LANDSAT 5 MSS and NOAA's AVHRR. The effects of different PSO-EMS control parameters were then studied. Finally, the performance of different versions of PSO was investigated.

From the results presented, it can be concluded that the PSO is an efficient optimization algorithm for difficult pattern recognition and image processing problems.

Chapter 8

Conclusion

This chapter briefly highlights the findings and contributions of this thesis and discusses directions for future research.

8.1 Summary

This thesis investigated the application of an efficient optimization method known as Particle Swarm Optimizer to the field of pattern recognition and image processing.

Chapter 4 presented a clustering approach using PSO. The objective of the proposed algorithm is to simultaneously minimize the quantization error and intra-cluster distances, and to maximize the inter-cluster distances. The application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images was investigated. The proposed algorithm was compared against *state-of-the-art* clustering algorithms. In general, the PSO algorithms produced better results with reference to inter- and intra-cluster distances, while having quantization errors comparable to the other algorithms. The performance of different versions of PSO was investigated and the results suggest that algorithms that start with high diversity and then gradually go to low diversity perform better than other algorithms. To test its performance on multidimensional feature spaces, the proposed approach was applied to multispectral imagery data.

Chapter 5 presented a tool for synthetic image generation (SIGT). The tool consists of two units: a synthetic image generator and a clustering verification unit.

The first unit allows the user to create a synthetic image based on a user-specified histogram suitable for the required application. The second unit allows the user to measure the efficiency of a clustering algorithm. Different features of SIGT were demonstrated by a set of experiments aided by the K-means clustering algorithm and the PSO-based clustering algorithm proposed in chapter 4. These experiments have demonstrated that the tool can help researchers in the field of unsupervised image classification to generate synthetic images, measure the quality of a clustering algorithm, compare different clustering algorithms and to create benchmarks.

Chapter 6 presented a new dynamic clustering algorithm based on PSO, called DCPSO, with application to unsupervised image classification. DCPSO clusters a data set without requiring the user to specify the number of clusters *a priori*. DCPSO uses a validity index to measure the quality of the resultant clustering. DCPSO has been applied to synthetic images (where the number of clusters was known *a priori*) as well as natural images (including MRI and satellite images), and was compared with other dynamic clustering techniques. In general, DCPSO successfully found the "optimum" number of clusters on the tested images. Genetic algorithm and random search versions of the proposed approach were presented and compared to the particle swarm version with both the genetic and PSO versions outperforming the random search version. The influence of the different DCPSO control parameters was then investigated. The use of different PSO versions was also studied. Finally, to test its performance in multidimensional feature space, the DCPSO was applied to multispectral imagery data.

Chapter 7 addressed two difficult problems in the field of pattern recognition and image processing. The two problems are: color image quantization and spectral unmixing. First, the chapter presented a PSO-based color image quantization algorithm (PSO-CIQ). The PSO-CIQ algorithm was compared against other well-known color image quantization techniques. In general, the PSO-CIQ performed better than other techniques when applied to a set of commonly used images. The effects of different PSO-CIQ control parameters were studied. The performance of different versions of PSO was then investigated. Chapter 7 then presented a new spectral unmixing approach using PSO (PSO-EMS). The objective of the PSO-EMS algorithm is to determine the appropriate set of end-members for a given multispectral image set. The PSO-EMS algorithm performed well when applied to test image sets from various platforms such as LANDSAT 5 MSS and NOAA's AVHRR. The effects of different PSO-EMS control parameters were then studied. Finally, the performance of different versions of PSO was investigated.

From the results presented in this thesis, it can be concluded that the PSO is an efficient optimization algorithm for difficult pattern recognition and image processing problems. These problems are considered difficult because they are NP-hard and combinatorial problems.

8.2 Future Research

Directions for future research are briefly summarized below.

PSO-based Clustering Algorithm

Although the parametric fitness function used by the PSO-approach contains multiple objectives, no special multi-objective optimization techniques have been used. Future research can investigate the use of a PSO multi-objective approach, which may produce better results. In addition, incorporating spatial information into the PSO-based clustering algorithm (when used in image segmentation applications) needs to be investigated. One way to incorporate spatial information is to consider the eight neighboring pixels of each pixel as proposed by Liew *et al*. [2000].

SIGT

Future additions to the tool may include a unit to generate a synthetic image from an existing real (synthetic) image by relaxing some constrains. In addition, further studies may use SIGT to do an elaborate analysis and comparison of clustering algorithms**.**

DCPSO

The application of the DCPSO algorithm (described in Section 6.1) to general data needs to be investigated. Furthermore, the effect of high dimensionality on the performance of the DCPSO should be investigated. Experiments for validating the efficiency of randomly re-initializing $\boldsymbol{M}_r$ (i.e. step 8 in Figure 6.1) need to be conducted. The DCPSO uses the K-means clustering algorithm to refine the cluster centroids. Future research can investigate the use of other more efficient clustering algorithms such as FCM and KHM. In addition, incorporating spatial information into the DCPSO algorithm (when used in image segmentation applications) needs to be investigated.

PSO-CIQ

The PSO-CIQ (described in section 7.1.1) uses the K-means clustering algorithm to refine the color triplets. Future research should investigate the use of other more efficient clustering algorithms such as FCM and KHM. Experiments need to be conducted to compare the PSO-CIQ with the multi-start K-means (with the best result generated from applying K-means $st_{\max}p_{\text{kmeans}}$ times, where each K-means starts from random cluster centroids). Finally, the PSO-CIQ uses the RGB color space. Although the RGB model is the most widely used model, it has some weaknesses. One of these weaknesses is that equal distances in the RGB color space may not correspond to equal distance in color perception. Hence, future research may try to apply the PSO-CIQ to other color spaces (e.g. the L*u*v* color space [Watt 1989]).

PSO-EMS

Experiments need to be conducted to compare the PSO-EMS with the multi-start K-means (with the best result generated from applying K-means $st_{\max}p_{\text{kmeans}}$ times, where each K-means starts from random cluster centroids). The performance of the PSO-EMS (described in section 7.2.1) when applied to hyperspectral Satellite imagery is a potential topic for future research.

Bibliography

E. Aarts and J. Lenstra. *Local Search in Combinatorial Optimization*. Princeton Universality Press, 2003.

H. Abbas and M. Fahmy. Neural Networks for Maximum Likelihood Clustering. *Signal Processing*, vol. 36, no.1, pp. 111-126, 1994.

B. Al-kazemi and C. Mohan. Multi-phase Discrete Particle Swarm Optimization. In *the Third International Workshop on Frontiers in Evolutionary Algorithms*, Atlantic City, New Jersey, USA, 2000.

N. Alldrin, A. Smith and D. Turnbull. Clustering with EM and K-means, unpublished Manuscript, 2003, http://louis.ucsd.edu/~nalldrin/research/cse253_wi03.pdf (visited 15 Nov 2003).

K. Al-Sultan. A Tabu Search Approach to Clustering Problems. *Pattern Recognition*, vol. 28, pp. 1443-1451, 1995.

M. Amadasun and R. King. Low-level Segmentation of Multispectral Images via Agglomerative Clustering of Uniform Neighborhoods. *Pattern Recognition*, vol. 21, no. 3, pp. 261-268, 1988.

M. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, USA, 1973.

P. Angeline. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference. In *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 601-610, 1998.

P. Angeline. Using Selection to Improve Particle Swarm Optimization. In *International Conference on Evolutionary Computation*, Piscataway, New Jersey, USA, pp. 84-89, IEEE Service Center, 1998.

J. Antoniades, D. Haas, P. Palmadesso, M. Baumback and L. J. Rickard. Use of Filter Vectors in Hyperspectral Data Analysis. In *Proceedings of SPIE*, vol. 2553, pp 128-139, 1995.

G. Babu and M. Murty. A Near-Optimal Initial Seed Value Selection in K-means Algorithm Using a Genetic Algorithm. *Pattern Recognition Letters*, vol. 14, no. 10, pp. 763-769, 1993.

F. Bach and M. Jordan. Learning Spectral Clustering. *Neural Information Processing Systems 16 (NIPS 2003)*, 2003.

T. Bäck. Self-Adaptation in Genetic Algorithms. In *Proceedings of the First European Conference on Artificial Life*, pp. 227-235, MIT Press, 1992.

T. Bäck, F. Hoffmeister and H. Schwefel. A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, pp. 2-9, 1991.

S. Baek, B. Jeon, D. Lee and K. Sung. Fast Clustering Algorithm for Vector Quantization. *Electronics Letters*, vol. 34, no. 2, pp. 151-152, 1998.

R. Balasubramanian and J. Allebach. A New Approach to Platte Selection for Color Images. *Image Technology*, vol. 17, pp. 284-290, 1990.

G. Ball and D. Hall. A Clustering Technique for Summarizing Multivariate Data. *Behavioral Science*, vol. 12, pp. 153-155, 1967.

A. Bateson and B. Curtiss. A Method for Manual Endmember Selection and Spectral Unmixing. *Remote Sensing of Enviornment*, vol. 55, pp 229-243, 1996.

J. Bezdek. A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 1-8, 1980.

J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

H. Bischof, A. Leonardis and A. Selb. MDL Principle for Robust Vector Quantization. *Pattern Analysis and Applications*, vol. 2, pp. 59-72, 1999.

C. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

N. Boujemaa. On Competitive Unsupervised Clustering. In *the International Conference on Pattern Recognition (ICPR'00)*, vol. 1, pp. 1631-1634, 2000.

E. Bonabeau, M. Dorigo and T. Theraulaz. *From Natural to Artificial Swarm Intelligence*. Oxford University Press, New York, USA, 1999.

M. Bramlette. Initialisation, Mutation and Selection Method in Genetic Algorithms for Function Optimization. In *Proceedings of the Fourth International Conference in Genetic Algorithms*, pp. 100-107, Morgan Kaufmann, 1991.

J. Braquelaire and L. Brun. Comparison and Optimization of Methods of Color Image Quantization. *IEEE Transactions on Image Processing*, vol. 6 no. 7, pp. 1048-1052, 1997.

C. Carpineto and G. Romano. A Lattice Conceptual Clustering System and Its Application to Browsing Retrieval. *Machine Learning*, vol. 24, no. 2, pp. 95-122, 1996.

M. Celenk. A Color Clustering Technique for Image Segmentation. *Computer Vision, Graphics and Image Processing*, vol. 52, pp. 145-170, 1990.

M. Chang, I. Sezan and M. Tekalp. Adaptive Bayesian Segmentation of Color Images. *Journal of Electronic Imaging*, vol. 3, no. 4, pp. 404-414, 1994.

C. Chen, J. Luo and K. Parker. Image Segmentation via Adaptive K-means Clustering and Knowledge-Based Morphological Operations with Biomedical Applications. *IEEE Transactions on Image Processing*, vol. 7, no. 12, pp. 1673-1683, 1998.

H. Cheng, X. Jaing, Y. Sun and J. Wang. Color Image Segmentation: Advances & Prospects. *Pattern Recognition*, vol.34, pp. 2259-2281, 2001.

S. Cheng and C. Yang. A Fast and Novel Technique for Color Quantization using Reduction of Color Space Dimensionality. *Pattern Recognition Letters*, vol. 22, pp. 845-856, 2001.

J. Chinneck. Practical Optimization: a Gentle Introduction, 2000. http://www.sce.carleton.ca/faculty/chinneck/po.html (visited 1 July 2004).

S. Chu and J. Roddick. A Clustering Algorithm Using Tabu Search Approach with Simulated Annealing for Vector Quantization. *Chinese Journal of Electronics*, vol. 12, no. 3, pp. 349-353, 2003.

F. Chung. *Spectral Graph Theory*. Society Press, 1997.

M. Clerc. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. In *Proceedings of the Congress on Evolutionary Computation*, Washington DC, USA, vol. 3, pp. 1951-1957, IEEE Press, 1999.

M. Clerc and J. Kennedy. The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73, 2001.

G. Coath and S. Halgamuge. A Comparison of Constraint-handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization

Problems. In *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, Canbella, Australia. pp. 2419-2425, 2003.

C.A. Coello Coello. *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design, PhD Thesis*. Tulane University, 1996.

C. Coello Coello and M. Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation*, Piscataway, New Jersey, USA, vol. 2, pp. 1051-1056, IEEE Service Center, 2002.

G. Coleman and H. Andrews. Image Segmentation by Clustering. In *Proceedings of IEEE*, vol. 67, pp. 773-785, 1979.

D. Comaniciu and P. Meer. Robust Analysis of Feature Spaces: Color Image Segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 750-755, 1997.

D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, 2002.

E. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, 2nd Edition, 1997.

D. Davies and D. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, 1979.

A. Dekker. Kohonen Neural Networks for Optimal Colour Quantization. *Network: Computation in Neural Systems*, vol. 5, pp. 351-367, 1994.

M. Delgado, A. Skarmeta and H. Barberá. A Tabu Search Approach to the Fuzzy Clustering Problem. In *the Sixth IEEE International Conference on Fuzzy Systems*, Barcelona, 1997.

J. Devore. *Probability and Statistics for Engineering and the Sciences, fourth edition*. Duxbury Press, 1995.

L. Diaz and T. Milligan. *Antenna Engineering Using Physical Optics: Practical CAD Techniques and Software (Artech House Antenna and Propagation Library*), Artech House Publishers, 1996.

M. Dorigo. *Optimization, Learning and Natural Algorithms (in Italian), PhD thesis*. Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

M. Dorigo and G. Di Caro. *The Ant Colony Optimization Meta-Heuristic*. *New Methods in Optimization*, D. Corne, M. Dorigo and F. Glover, Eds., McGraw-Hill, 1999.

M. Dorigo, V. Maniezzo and A. Colorni. Positive Feedback as a Search Strategy. Technical Report, Report no. 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

J. C. Dunn. Well Separated Clusters and Optimal Fuzzy Partitions. *Journal of Cybernetics*, vol. 4, pp. 95-104, 1974.

R. Eberhart, P. Simpson and R. Dobbins. *Computational Intelligence PC Tools*. Morgan Kaufmann, 1996.

R. Eberhart and Y. Shi. Comparison between Genetic Algorithms and Particle Swarm Optimization. In *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 611-619. Springer-Verlag, 1998.

R. Eberhart and Y. Shi. Evolving Artificial Neural Networks. In *Proceedings of the International Conference on Neural Networks and Brain*, Beijing, China, PL5-PL13, 1998.

R. Eberhart and Y. Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the Congress on Evolutionary Computing*, San Diego, USA, pp. 84-89, 2000.

A. El-Gallad, M. El-Hawary and A. Sallam. Swarming of Intelligent Particles for Solving the Nonlinear Constrained Optimization Problem. *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 9, no. 3, pp. 155-163, 2001.

A. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley and Sons, 2002.

S. Esquivel and C. Coello Coello. On the use of Particle Swarm Optimization with Multimodal Functions. In *Proceedings of IEEE Congress on Evolutionary Computation*, pp 1130-1136, 2003.

B. Everitt. *Cluster Analysis*. Heinemann Books, London, 1974.

J. Fieldsend and S. Singh. A Multi-objective Algorithm based upon Particle Swarm Optimization, an Efficient Data Structure and Turbulence. In *The 2002 UK Workshop on Computational Intelligence*, UK, pp. 34-44, 2002.

E. Fiume and M. Quellette. On Distributed, Probabilistic Algorithms for Computer Graphics. *Graphics Interface '89*, pp. 211-218, 1989.

R. Fletcher. *Practical Methods of Optimization, second edition*. John Wiely & Sons, 2000.

C. Floudas and P. Pardalos. *Recent Advances in Global Optimization*. Princeton Universality Press, 1992.

L. Fogel. *Evolutionary Programming in Perspective: The Top-down View. Computational Intelligence: Imitating Life*, J.M. Zurada, R. Marks II and C. Robinson, Eds., Piscataway, New Jersey, USA, IEEE Press, 1994.

E. Forgy. Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of Classification. *Biometrics*, vol. 21, pp. 768-769, 1965.

B. Freisleben and A. Schrader. An Evolutionary Approach to Color Image Quantization. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 459-464, 1997.

H. Frigui and R. Krishnapuram. Clustering by Competitive Agglomeration. *Pattern Recognition Letters*, vol. 30, no. 7, pp. 1109-1119, 1997.

H. Frigui and R. Krishnapuram. A Robust Competitive Clustering Algorithm with Applications in Computer Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no.5, pp. 450-465, 1999.

P. Frnti, J. Kivijrvi and O. Nevalainen. Tabu Search Algorithm for Codebook Generation in Vector Quantization. *Pattern Recognition*, vol. 31, no. 8, pp. 1139-1148, 1998.

C. Fuh, S. Cho and K. Essig. Hierarchical Color Image Region Segmentation for Content-Based Image Retreival Systems. *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 156-162, 2000.

Y. Fukuyama and H. Yoshida. A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Seoul, S. Korea, pp. 87-93, 2001.

K. Gabarro. Tabu Search Algorithm, http://www.lsi.upc.es/~mallba/public/library/firstProposal-BA/node11.html (visited 18 August 2004).

Z. Gaing. Particle Swarm Optimization to Solving the Economic Dispatch Considering the Generator Constraints. *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1187-1195, 2003.

I. Gath and A. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 773-781, 1989.

M. Gervautz and W. Purgathofer. *A Simple Method for Color Quantization: Octree Quantization*. Graphics Gems, Academic Press, N.Y., 1990.

F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190-206, 1989.

F. Glover. Tabu Search – Part II. *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4-32, 1990.

D. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

P. Gray, W. Hart, L. Painton, C. Phillips, M. Trahan and John Wagner. A Survey of Global Optimization Methods, Sandia National Laboratories, 1997, http:// www.cs.sandia.gov/opt/survey (visited 2 July 2004).

M. Halkidi, Y. Batistakis and M. Vazirgiannis. On Clustering Validation Techniques. *Intelligent Information Systems Journal*, Kluwer Pulishers, vol. 17, no. 2-3, pp.107-145, 2001.

M. Halkidi and M. Vazirgiannis. Clustering Validity Assessment: Finding the Optimal Partitioning of a data set. In *Proceedings of ICDM Conference*, CA, USA, 2001.

M. Halkidi and M. Vazirgiannis. Clustering Validity Assessment using Multi representative. In *Proceedings of the Hellenic Conference on Artificial Intelligence, SETN*, Thessaloniki, Greece, 2002.

G. Hamerly. Learning Structure and Concepts in Data using Data Clustering, *PhD Thesis*. University of California, San Diego, 2003.

G. Hamerly and C. Elkan. Alternatives to the K-means Algorithm that Find Better Clusterings. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM-2002)*, pp. 600-607, 2002.

G. Hamerly and C. Elkan. Learning the K in K-means. In *The Seventh Annual Conference on Neural Information Processing Systems*, 2003.

P. Heckbert. Color Image Quantization for Frame Buffer Display. *ACM Computer Graphics*, vol. 16, no. 3, pp. 297-307, 1982.

N. Higashi and H. Iba. Particle Swarm Optimization with Gaussian Mutation. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, Indianapolis, Indiana, USA. pp. 72-79, 2003.

A. Hlavka and M. A. Spanner. Unmixing AVHRR Imagery to Access Clearcuts and Forest Regrowth on Oregon. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, pp 788-795, 1995.

J. Holland. Outline for a Logical Theory of Adaptive Systems. *Journal of the ACM*, vol. 3, pp. 297-314, 1962.

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Michigan, USA, 1975.

F. Hoppner, F. Klawonn, R. Kruse and T. Runkler. *Fuzzy Cluster Analysis, Methods for Classification, Data Analysis and Image Recognition*. John Wiley & Sons Ltd, 1999.

R. Horst, P. Pardalos and N. Thoai. *Introduction to Global Optimization*, second edition. Kluwer Academic Publishers, 2000.

X. Hu. Particle Swarm Optimization: Bibliography, 2004. http://www.swarmintelligence.org/bibliography.php (visited 8 February 2005).

X. Hu and R. Eberhart. Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. In *Proceedings of congress on Evolutionary Computation*, Hawaii, USA, pp. 1666-1670, 2002.

X. Hu and R. Eberhart. Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization. In *Proceedings of congress on Evolutionary Computation*, Hawaii, USA, pp. 1677-1681, 2002.

X. Hu and R. Eberhart. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *the Sixth World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, USA, 2002.

K. Huang. A Synergistic Automatic Clustering Technique (Syneract) for Multispectral Image Analysis. *Photogrammetric Engineering and Remote Sensing*, vol. 1, no.1, pp. 33-40, 2002.

A. Ismail and A. Engelbrecht. Global Optimization Algorithms for Training Product Unit Neural Networks. In *IEEE International Conference on Neural Networks,* Como, Italy,2000.

A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, USA, 1988.

A. Jain, R. Duin and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no.1, pp. 4-37, 2000.

R. Jain, R. Kasturi and B. Schunck. *Machine Vision*. McGraw-Hill, Inc., New York, USA, 1995.

A. Jain, M. Murty and P. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323,1999.

C. Janikow and Z. Michalewicz. An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithm. In *Proceedings of the Fourth International Conference in Genetic Algorithms*, pp. 31-36, Morgan Kaufmann, 1991.

A. Jensen and S. Kristensen. Basic PSO versus Multi Swarm PSO. Topics of Evolutionary Computation, EVALife, Department of Computer Science, University of Aarhus, Denmark, 2002.

D. Judd, P. Mckinley and A. Jain. Large-scale Parallel Data Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 871-876, 1998.

R. Kass and L. Wasserman. A Reference Bayesian Test for Nested Hypotheses and its Relationship to the Schwarz Criterion. *Journal of the American Statistical Association*, vol. 90, no. 431, pp. 928-934, 1995.

T. Kaukoranta, P. Fränti and O. Nevalainen. A New Iterative Algorithm for VQ Codebook Generation. *International Conference on Image Processing*, pp. 589-593, 1998.

J. Kennedy. Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. In *Proceedings of the Congress on Evolutionary Computation*, pp. 1931-1938, 1999.

J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, vol. 4, pp. 1942-1948, 1995.

J. Kennedy and R. Eberhart. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the Conference on Systems, Man, and Cybernetics*, pp. 4104-4109, 1997.

J. Kennedy and R. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

J. Kennedy and R. Medes. Population Structures and Particle Swarm Performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Hawaii, USA, 2002.

J. Kennedy and W. Spears. Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator. In *IEEE International Conference on Evolutionary Computation*, Achorage, Alaska, USA, 1998.

R. Klein and R. Dubes. Experiments in Projection and Clustering by Simulated Annealing. *Pattern Recognition*, vol. 22, pp. 213-220, 1989.

T. Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences, 30, Springer-Verlag, NewYork, USA, 1995.

B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, *second edition*. Springer-Verlag, Berlin, 2002.

C. Kotropoulos, E. Augé and I. Pitas. *Two-layer Learning Vector Quantizer for Color Image Quantization*. *Signal Processing IV: Theories and Applications*, J. Vandewalle, R. Boite, M. Moonen, A. Oosterlinck, Eds., pp. 1177-1180, 1992.

J. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

T. Krink, and M. Løvbjerg. The LifeCycle model: Combining Particle Swarm Optimisation, Genetic Algorithms and HillClimbers. In *Proceedings of Parallel Problem Solving from Nature VII*, pp. 621-630, 2002.

T. Krink, J. Vesterstrøm, J. Riget. Particle Swarm Optimization with Partial Particle Extension. In *Proceedings of the Fourth Congress on Evolutionary Computation*, 2002.

Krishnapuram and Keller. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 98-110, 1993.

Krishnapuram and Keller. The Possibilistic C-Means algorithm: Insights and Recommendations. *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 385-393, 1996.

L. Kuncheva and J. Bezdek. Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search?. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 28, no. 1, pp. 160-164, 1998.

S. Kwok and A. Constantinides. A Fast Recursive Shortest Spanning Tree for Image Segmentation and Edge Detection. *IEEE Transactions on Image Processing*, vol. 6, no. 2, pp. 328-332, 1997.

C. Lee and E. Antonsson. Dynamic Partitional Clustering Using Evolution Strategies. In *The Third Asia-Pacific Conference on Simulated Evolution and Learning*, 2000.

A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering, second edition*. Addison Wesley, 1994.

Y. Leung, J. Zhang and Z. Xu. Clustering by Space-Space Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no.12, pp. 1396-1410, 2000.

A. Liew, S. Leung and W. Lau. Fuzzy Image Clustering Incorporating Spatial Continuity. In *IEE Proceedings Vision, Image and Signal Processing*, vol. 147, no. 2, 2000.

T. Lillesand and R. Kiefer. *Remote Sensing and Image Interpretation*, John Wiley & Sons Publishing, New York, USA, 1994.

A. Lorette, X. Descombes and J. Zerubia. Fully Unsupervised Fuzzy Clustering with Entropy Criterion. In *International Conference on Pattern Recognition (ICPR'00)*, vol. 3, pp. 3998-4001, 2000.

M. Løvberg. *Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self Organized Critically*, *Master's Thesis*. Department of Computer Science, University of Aarhus, Denmark, 2002.

M. Løvberg, T. Rasmussen and T. Krink. Hybrid Particle Swarm Optimiser with Breeding and Subpopulation. In *Proceedings of the Third Genetic and Evolutionary Computation Conference*, vol. 1, pp. 469-476, 2001.

M. Løvberg and T. Krink. Extending Particle Swarm Optimizers with Self-Organized Criticality. In *Proceedings of the Fourth Congress on Evolutionary Computation*, vol. 2, pp. 1588-1593, 2002.

S. Lu and K. Fu. A Sentence-to-Sentence Clustering Procedure for Pattern Analysis. *IEEE Transaction on Systems, Man and Cybernetics*, vol. 8, pp. 381-389, 1978.

L. Lucchese and S. Mitra. Color Image Segmentation: A State-of-the-Art Survey. In *Proceedings of the Indian National Science Academy (INSA-A)*, New Delhi, India, vol. 67, no. 2, pp. 207-221, 2001.

J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings Fifth Berkeley Symposium on Mathematics, Statistics and Probability*, vol. 1, pp. 281-297, 1967.

F. Maselli. Multiclass Spectral Decomposition of Remotely Sensed Scenes by Selective Pixel Unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, no. 5, pp. 1809-1819, 1998.

U. Maulik and S. Bandyopadhyay. Genetic Algorithm-Based Clustering Technique. *Pattern Recognition*, vol. 33, pp. 1455-1465, 2000.

G. McLachlan and T. Krishnan. *The EM algorithm and Extensions*. John Wiley & Sons, Inc., 1997.

K. Mehrotra, C. Mohan and Rakka. *Elements of Artificial Neural Networks*. MIT Press, 1997.

Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*, third edition. Springer-Verlag, Berlin, 1996.

Z. Michalewicz and D. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, Berlin, 2000.

C. Mohan and B. Al-Kazemi. Discrete Particle Swarm Optimization. In *Proceedings Workshop on Particle Swarm Optimization*, Purdue School of Engineering and Technology, USA, 2001.

F. Murtagh, A. Raftery and J. Starck. Bayesian Inference for Color Image Quantization via Model-Based Clustering Trees. Technical Report no. 402. Department of Statistics, University of Washington, USA, 2001.

A. Ng, M. Jordan and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. In *Proceedings of Neural Information Processing Systems (NIPS 2001)*, 2001.

J. Oliver and D. Hand. Introduction to Minimum Encoding Inference. Technical Report no. 94/205. Department of Computer Science, Monash University, Australia, 1994.

E. Ozcan and C. Mohan. Analysis of a Simple Particle Swarm Optimization System. *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 8, pp. 253-258, 1998.

N. Pal and J. Biswas. Cluster Validation using Graph Theoretic Concepts. *Pattern Recognition*, vol. 30, no. 6, 1997.

A. Pandya and R. Macy. *Pattern Recognition with Neural Networks in C++*. CRC Press, 1996.

T. Pappas. An Adaptive Clustering Algorithm for Image Segmentation. *IEEE Transactions on Signal Processing*, vol. 40, no, 4, pp. 901-914, 1992.

P. Pardalos, A. Migdalas and R. Burkard. *Combinatorial and Global Optimization*. World Scientific Publishing Company, 2002.

L. Parra, C. Spence, P. Sajda, A. Ziehe and K. Müller. Unmixing Hyperspectral Data. In *Advances in Neural Information Processing Systems 12*, MIT Press, pp. 942-948, 2000.

K. Parsopoulos and M. Vrahatis. *Particle Swarm Optimization Method for Constrained Optimization Problems*. *Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies*, P. Sincak, J. Vascak, V. Kvasnicka and J. Pospichal, Eds., IOS Press, 2002.

E. Peer, F. Van den Bergh and A. Engelbrecht. Using Neighborhoods with the Guaranteed Convergence PSO. In *Swarm Intelligence Symposium*, Piscataway, New Jersey, USA, pp. 235-242, IEEE Service Center, 2003.

D. Pelleg and A. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 727-734, Morgan Kaufmann, San Francisco, CA, 2000.

J. Puzicha, T. Hofmann and J. M. Buhmann. Histogram Clustering for Unsupervised Image Segmentation. In *IEEE Proceedings of the Computer Vision and Pattern Recognition*, vol. 2, pp. 602-608, 2000.

V. Raghavan and K. Birchand. A Clustering Strategy Based on a Formalism of the Reproductive Process in a Natural System. In *Proceedings of the Second International Conference on Information Storage and Retrieval*, pp. 10-22, 1979.

R. Rardin. *Optimization in Operations Research*. Prentice Hall, New Jersey, USA, 1998.

A. Ratnaweera, S. Halgamuge and H. Watson. Particle Swarm Optimization with Self-adaptive Acceleration Coefficients. In *Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery 2002 (FSKD 2002)*, pp. 264-268, 2003.

R. Rendner and H. Walker. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, vol. 26, no. 2, 1984.

R. Reynolds, B. Peng and J. Brewster. Cultural swarms II: Virtual algorithm emergence. In *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, Canbella, Australia, pp. 1972-1979, 2003.

J. Riget and J. Vesterstrøm. A Diversity-Guided Particle Swarm Optimizer – The ARPSO. EVALife Technical Report no. 2002-2, 2002.

Rissanen. Modeling by Shortest Data Description. *Automatica,* vol. 14, pp. 465-471, 1978.

J. Robinson, S. Sinton and Y. Rahmat-Samii. Particle Swarm, Genetic Algorithm, and their Hybrids: Optimization of a Profiled Corrugated Horn Antenna. In *IEEE International Symposium on Antennas & Propagation*. San Antonio, Texas, USA, 2002.

C. Rosenberger and K. Chehdi. Unsupervised Clustering Method with Optimal Estimation of the Number of Clusters: Application to Image Segmentation. In *The International Conference on Pattern Recognition (ICPR'00)*, vol. 1, pp. 1656-1659, 2000.

X. Rui, C. Chang and T. Srikanthan. On the initialization and Training Methods for Kohonen Self-Organizing Feature Maps in Color Image Quantization. In *Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications*, 2002.

E. Saber, A. Tekalp and G. Bozdagi. Fusion of Color and Edge Information for Improved Segmentation and Edge Linking. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2176-2179, 1996.

J. Saghri, A. Tescher, F. Jaradi and M. Omran. A Viable End-Member Selection Scheme for Spectral Unmixing of Multispectral Satellite Imagery Data. *Journal of Imaging Science and Technology*, vol. 44, no. 3, pp. 196-203, 2000.

J. Saghri, A. Tescher and M. Omran. Class-Prioritized Compression of Multispectral Imagery Data. *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 246-256, 2002.

A. Salman. Linkage *Crossover Operator for Genetic Algorithms, PhD Dissertation*. School of Syracuse University, USA, 1999.

P. Scheunders. A Comparison of Clustering Algorithms Applied to Color Image Quantization. *Pattern Recognition Letters*, vol. 18, no. 11-13, pp. 1379-1384, 1997.

P. Scheunders. A Genetic C-means Clustering Algorithm Applied to Image Quantization. *Pattern Recognition*, vol. 30, no. 6, 1997.

P. Scheunders and S. De Backer. Joint Quantization and Error Diffusion of Color Images using Competitive Learning. In *International Conference on Image Processing*, vol. 1, pp. 811, 1997.

L. Schoofs and B. Naudts. Swarm Intelligence on the Binary Constraint Satisfaction Problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii USA, 2002

J. J. Settle and N. A. Drake. Linear Mixing and Estimation of Ground Cover Proportions. *International Journal in Remote Sensing*, vol. 14, no. 6, pp 1159-1177, 1993.

S. Shafer and T. Kanade. Color Vision. *Encyclopedia of Artificial Intelligence*, pp. 124-131, Wiley, 1987.

Y. Shi and R. Eberhart. A Modified Particle Swarm Optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, Piscataway, New Jersey, pp. 69-73, 1998.

Y. Shi and R. Eberhart. Parameter Selection in Particle Swarm Optimization. *Evolutionary Programming VII: Proceedings of EP 98*, pp. 591-600, 1998.

Y. Shi and R. Eberhart. Fuzzy Adaptive Particle Swarm Optimization. In *Proceedings Congress on Evolutionary Computation*, Seoul, S. Korea, 2001.

J. Shi and J. Malik. Normalized Cuts and Image Segmentation. In *Proceedings of IEEE International Conference on computer Vision and Pattern Recognition*, pp. 731-737, 1997.

P. Sneath and R. Sokal. *Numerical Taxonomy*. Freeman, London, UK, 1973.

J. Spall. *Introduction to Stochastic Search and Optimization*, first edition. Wiley-Interscience, 2003.

R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Global Optimization*, vol. 11, pp. 341-359, 1997.

M. Su. Cluster Analysis: Chapter two Lecture notes, 2002, http://selab.csie.ncu.edu.tw/~muchun/course/cluster/CHAPTER%202.pdf (visited 15 August 2004).

P. Suganthan. Particle Swarm Optimizer with Neighborhood Optimizer. In *Proceedings of the Congress on Evolutionary Computation*, pp. 1958-1962, 1999.

S. Theodoridis and K. Koutroubas. *Pattern Recognition*. Academic Press, 1999.

J. Tou. DYNOC – A Dynamic Optimal Cluster-seeking Technique. *International Journal of Computer and Information Sciences*, vol. 8, no. 6, pp. 541-547, 1979.

J. Tou and R. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, Massachusetts, USA, 1974.

I. Trelea. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. *Information Processing Letters*, vol. 85, no. 6, pp. 317-325, 2003.

M. Trivedi and J. Bezdek. Low-level Segmentation of Aerial Images with Fuzzy Clustering. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 4, pp. 589-598, 1986.

D. Tsou and C. MacNish. Adaptive Particle Swarm Optimisation for High-dimensional Highly Convex Search Spaces. In *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, Canbella, Australia. pp. 783-789, 2003.

R.H. Turi. *Clustering-Based Colour Image Segmentation, PhD Thesis*. Monash University, Australia, 2001.

T. Uchiyama and M. Arbib. Color Image Segmentation using Competitive Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1197-1206, 1994.

F. Van den Bergh. *An Analysis of Particle Swarm Optimizers, PhD Thesis*. Department of Computer Science, University of Pretoria, South Africa, 2002.

F. Van den Bergh and A. Engelbrecht. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal,* vol. 26, pp. 84-90, 2000.

F. Van den Bergh and A.P. Engelbrecht. Effects of Swarm Size on Cooperative Particle Swarm Optimizers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, USA, pp. 892-899, 2001.

F. Van den Bergh and A.P. Engelbrecht. A New Locally Convergent Particle Swarm Optimizer. *In Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, 2002.

D. Van der Merwe and A. Engelbrecht. Data Clustering using Particle Swarm Optimization. In *IEEE Congress on Evolutionary Computation*. Canberra, Australia, pp. 215-220, 2003

P. Van Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1987.

C. Veenman, M. Reinders and E. Backer. A Maximum Variance Cluster Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273-1280, 2002.

C. Veenman, M. Reinders and E. Backer. A Cellular Coevolutionary Algorithm for Image Segmentation. *IEEE Transactions on Image Processing*, vol. 12, no. 3, pp. 304-316, 2003.

K. Veeramachaneni, T. Peram, C. Mohan and L. Osadciw. Optimization Using Particle Swarm with Near Neighbor Interactions. *Lecture Notes Computer Science,* vol. 2723, Springer Verlag, 2003.

L. Velho, J. Gomes and M. Sobreiro. Color Image Quantization by Pairwise Clustering. In *Proceedings of the Tenth Brazilian Symposium on Computer Graphics and Image Processing*, pp. 203-207, 1997.

G. Venter and J. Sobieszczanski-Sobieski. Particle Swarm Optimization. In *the 43rd AIAA/ASME/ASCE/AHA/ASC Structures, Structural Dynamics and Materials Conference*, Denver, Colorado, USA, 2002.

C. Wallace. An Improved Program for Classification. Technical Report no. 47. Department of Computer Science, Monash University, Australia, 1984.

C. Wallace and D. Boulton. An Information Measure for Classification. *The Computer Journal*, vol. 11, pp. 185-194, 1968.

C. Wallace and D. Dowe. Intrinsic Classification by MML – the snob program. In *Proceedings Seventh Australian Joint Conference on Artificial Intelligence, UNE*, Armidale, NSW, Australia, pp. 37-44, 1994.

S. Wan, P. Prusinkiewicz and S. Wong. Variance-based Color Image Quantization for Frame Buffer Display. Color Research and Application, vol. 15, no. 1, pp. 52-58, 1990.

A. Watt. *Three-Dimensional Computer Graphics*. Addison-Wesley, 1989.

H. Weiss. Genetic Algorithms and Optimum Robot Design, Institute of Robotics and Mechatronics, 2003, http://www.robotic.dlr.de/Holger.Weiss/garep/node3.html (visited 6 July 2004).

D. Whitley and S. Rana. Search, Binary Representations, and Counting Optima. In *Proceeding of a Workshop on Evolutionary Algorithms*, Sponsored by the Institute for Mathematics and its Applications, 1998.

J. Wu, H. Yan and A. Chalmers. Color Image Segmentation Using Fuzzy Clustering and Supervised Learning. *Journal of Electronic Imaging*, vol. 3, no. 4, pp. 397-403, 1994.

X. Wu and K. Zhang. A Better Tree-Structured Vector Quantizer. In *Proceedings IEEE Data Compression Conference*, pp. 392-401, 1991.

Z. Xiang. Color Image Quantization by Minimizing the Maximum Inter-cluster Distance. *ACM Transactions on Graphics*, vol. 16, no. 3, pp. 260-276, 1997.

Z. Xiang and G. Joy. Color Image Quantization by Agglomerative Clustering. *IEEE Computer Graphics and Applications*, vol. 14, no. 3, pp. 44-48, 1994.

X. Xie, W. Zhang and Z. Yang. A Dissipative Particle Swarm Optimization. In *IEEE Congress on Evolutionary Computation*, Honolulu, Hawaii, USA, 2002.

K. Yasuda, A. Ide and N. Iwasaki. Adaptive Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 1554-1559, 2003.

H. Yoshida, K. Kawata, Y. Fukuyama and Y. Nakanishi. A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Stability. In *Proceedings of the International Conference on Intelligent System Application to Power Systems*, Rio de Janeiro, Brazil, pp. 117–121, 1999.

B. Zhang. Generalized K-Harmonic Means - Boosting in Unsupervised Learning. Technical Report HPL-2000-137. Hewlett-Packard Labs, 2000.

B. Zhang, M. Hsu and U. Dayal. K-Harmonic Means - A Data Clustering Algorithm. Technical Report HPL-1999-124. Hewlett-Packard Labs, 1999.

W. Zhang, Y. Liu and M. Clerc. An Adaptive PSO Algorithm for Reactive Power Optimization. In *Advances in Power System Control Operation and Management*, Hongkong. 2003.

W. Zhang and X. Xie. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. In *IEEE International Conference on Systems, Man and Cybernetics*, Washington DC, USA, pp. 3816-3821, 2003.

Y. Zhang, M. Brady and S. Smith. Segmentation of Brain MR Images Through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm. *IEEE Transactions on Medical Imaging*, vol. 20, no. 1, pp. 45-57, 2001.

Y. Zheng, L. Ma, L. Zhang and J. Qian. Robust PID Controller Design using Particle Swarm Optimizer. In *Proceedings of IEEE International Symposium on Intelligence Control*, pp. 974-979, 2003.

Appendix A

Definition of Terms and Symbols

This appendix lists the terms and symbols frequently used in this thesis.

pattern is a single object or data point used by the clustering algorithm.

cluster is a set of similar patterns, and patterns from different clusters are not similar.

N_c	is the maximum number of clusters.
N_d	is the dimension of the data set.
N_p	is the number of patterns to be clustered. If the data set is an image (or a set of images) N_p denotes the number of image pixels.
$\Re$	is the set of all real numbers
$\boldsymbol{Z}$	denotes the dataset being clustered (i.e. the set of patterns).
$\boldsymbol{z}_p$	denotes the coordinates of pattern (or pixel) p.
$\boldsymbol{C}_k$	denotes the k^{th} cluster.
$\boldsymbol{m}_k$	denotes the centroid of $\boldsymbol{C}_k$.
K	denotes the number of clusters.
$\boldsymbol{x}_i$	is the current position of particle i.
$\boldsymbol{v}_i$	is the current velocity of particle i.
$\boldsymbol{y}_i$	is the personal best position of particle i.
$\hat{\boldsymbol{y}}_i$	is the neighborhood best position of particle i.
$\hat{\boldsymbol{y}}$	is the position of the global best particle.
f	denotes the function being optimized.
t	denotes time or time steps.

Appendix B

Derived Publications

This appendix provides a list of publications that have been published, or are currently being reviewed, that were derived from the work introduced in this thesis.

Journal Publications:

1. M. Omran, A. Engelbrecht and A. Salman. Particle Swarm Optimization Method for Image Clustering. To appear in the in the *International Journal of Pattern Recognition and Artificial Intelligence*.
2. A. Salman, M. Omran and A. Engelbrecht. SIGT: Synthetic Image Generation Tool for Clustering Algorithms. To appear in the *ICGST International Journal on Graphics, Vision and Image Processing (GVIP)*, vol. V2, pp. 33-44, January, 2005.
3. M. Omran, A. Salman and A. Engelbrecht. Dynamic Clustering using Particle Swarm Optimization with Application in Image Segmentation. *Pattern Recognition*, submitted 2004.
4. M. Omran, A. Engelbrecht and A. Salman. A PSO-based End-Member Selection Method for Spectral Unmixing of Multispectral Satellite Images, *Pattern Recognition*, submitted, 2004.
5. M. Omran, A. Engelbrecht amd A. Salman. A PSO-based Color Image Quantizer, *Special issue of Soft Computing Journal in image processing*, submitted, 2004.

Book Chapter:

6. M. Omran, A. Engelbrecht and A. Salman. Image Classification using Particle Swarm Optimization. *Recent Advances in Simulated Evolution and Learning*, K. Tan, M. Lim, X. Yao and L. Wang (Editors), World Scientific, Series on Advances in Natural Computation, 2004.

Conference Publications:

7. M. Omran, A. Salman and A. Engelbrecht. Image Classification using Particle Swarm Optimization. In *Conference on Simulated Evolution and Learning*, Singapore, pp. 370-374, November 2002.