Ground cover classification of satellite imagery on parallel architectures

Liu, Bo

2005

Liu, B. (2005). Ground cover classification of satellite imagery on parallel architectures.
Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2364>

<https://doi.org/10.32657/10356/2364>

Nanyang Technological University

Downloaded on 26 Apr 2023 17:02:23 SGT

GROUND COVER CLASSIFICATION OF SATELLITE IMAGERY ON PARALLEL ARCHITECTURES

LIU BO

SCHOOL OF COMPUTER ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

2005

Ground Cover Classification of Satellite Imagery on Parallel Architectures

Liu Bo

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2005

*To my father,
in memory of my beautiful childhood with you beside*

*To my family,
thank you for giving me the opportunity to receive good education
however hard life was*

Acknowledgments

I would like to express my deepest gratitude and respect to the following people, who have done so much in my research of the past two years.

First and foremost, I would specially thank my supervisor, Dr. Timo Rolf Bretschneider, for his remarkable guidance, advices and encouragement during my research work in NTU. In the research, he always encouraged me to explore new problems and face new challenges. His broad knowledge, insightful vision and amiable personality helped so much in my study and research.

I owe a lot to Mrs. Sharon Lim Siok Lin (Satellite Engineering Centre), who has contributed so many excellent suggestions with respect to the Parallel Processing Unit of the X-Sat project.

I also want to express my gratitude to Mr. Ng Hee Khiang, Mrs. Ng-Goh Siew Lai and Mr. Ek Ming Hong. They were so patient to help me with the technical problems and have done a lot of work in building the testbed.

I also want to say thanks to Mr. Li Yikun, Mr. Wang Lizhe, Mr. Liu Tong, Mrs. Song Ying, and all others who have contributed to my research work in one way or another.

Contents

Acknowledgments	ii
Summary	vi
List of Figures	vii
List of Algorithms	ix
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.2 Scope and Objectives	3
1.3 Organisation of this Thesis	4
2 Basic Concepts and Related Work	5
2.1 Digital Image Classification in Remote Sensing	5
2.1.1 Unsupervised Satellite Image Classification	8
2.1.2 Presentation of Classification Result	9
2.2 Unsupervised Machine Learning and Clustering	11
2.2.1 Mixture Densities	11
2.2.2 K-Means Clustering Algorithm	12
2.2.3 Fuzzy k-Means Clustering Algorithm	13
2.2.4 ISODATA Algorithm	15
2.2.5 ISMC Algorithm	17
2.2.6 Mean Shift Clustering Algorithm	20
2.3 Criterion Functions for Clustering Algorithm	24
2.3.1 Sum-of-Squared Error Criterion	25
2.3.2 Classification Accuracy	26

2.3.3	Scatter Criteria	26
2.4	Parallel Image Processing Platforms	28
2.4.1	Image Processing Platform on PC Cluster	28
2.4.2	Image Processing Platform on Computational Grid	31
2.4.3	Parallel Image Processing Platform on X-Sat	34
3	Enhanced ISMC Image Classification Algorithm	37
3.1	Enhanced ISMC Algorithm	37
3.2	Approximation of the Largest Distance	39
3.3	Design of the D-ISMC Algorithm	40
3.4	Implementation on Parallel Architectures	45
3.4.1	Implementation on a PC Cluster	45
3.4.2	Implementation on the PPU of X-Sat	46
3.5	Evaluation of the Classification Result	49
3.6	Summary	55
4	Mean-Shift Based Image Classification Algorithm	56
4.1	Overlapping Issue in Classification Algorithms	56
4.2	Implementation of Mean Shift Algorithm on Parallel Architectures	63
4.2.1	Design of the Distributed Mean Shift Classification Algorithm	63
4.2.2	Implementation on a PC Cluster	65
4.3	Evaluation of the Classification Result	67
4.4	Summary	71
5	Recommendations for Future Developments	74
5.1	Floating Point Computing	74
5.2	Histogram Building and Information Exchange	75
5.3	Accelerating Cleaning-up Stage	75
6	Conclusions	76
	Bibliography	78
	Appendices	84

A 6-band Landsat Test Image	84
B Other Test Images	85
C matlab source code for function mixgen	92
D matlab source code for function emgauss	96

Summary

Unsupervised image classification is a very powerful image processing tool used in the domain of remote sensing. In this thesis, a detailed review is performed on some existed unsupervised classification algorithms, such as k-means, fuzzy k-means, ISODATA, Improved Splitting and Merging Clustering (ISMC) algorithm, and Mean-Shift algorithm. In particular, the ISMC algorithm and the Mean-Shift algorithm are analysed carefully here.

To improve the classification accuracy of the ISMC algorithm, two k-means steps were added after the splitting step and the merging step, respectively. For the assessment, the classification results are compared with those of the k-Means as well as the ISMC algorithm. The findings were that the classification accuracy increased as well as the processing speed.

In the analysis of the Mean Shift Algorithm, to handle the overlapping issue in the feature space, the spatial information of every pixel was taken into account and a filter derived. With the so-called dominate filter, the overlapping of cluster clouds in the feature space is considered and a higher degree of classification accuracy was achieved.

To improve the processing speed, the two above algorithms were both implemented on a PC Cluster, and satisfying speedups were gained. Due to external time delays in completing the Parallel Processing Unit (PPU) of the X-Sat satellite only a rough design is given for this particular architecture without real implementations and experimental results.

An approximation method to find the largest pair-wise distance in a data set is also presented with computing complexity of $O(n)$ in contrast to the $O(n \times \log n)$ complexity of the ISMC algorithm. According to the experiments, this algorithm can find the largest pair-wise distance with a probability of 0.96.

List of Figures

1.1	Flow of digital image classification system.	2
2.1	Unsupervised classification on 6-band Landsat remotely sensed image . .	7
2.2	Spectral classes in two-band image data	9
2.3	Unsupervised classification on two-band image data	10
2.4	Flowchart of the ISMC algorithm.	21
2.5	Iterative modification of the histogram by the Mean Shift algorithm . . .	23
2.6	Shareing data between cluster nodes via NFS	30
2.7	Message passing within cluster	31
2.8	Architecture of the used hierarchical grid system.	33
2.9	Hierarchical parallel programming environment consisting of MPICH-G2 and MPICH programs [13].	34
2.10	Hierarchical parallel programming environment consisting of MPICH-G2 and SUN Grid Engine.	35
2.11	Architecture of Parallel Processing Unit [12]	36
3.1	Classification by the ISMC algorithm.	38
3.2	Classification by the Enhanced-ISM algorithm.	38
3.3	Flowchart of the Enhanced-ISM algorithm.	39
3.4	Approximative method for finding the largest pair-wise distance.	41
3.5	Schedule of the D-ISM algorithm on a PC-Cluster	46
3.6	Flow chart of the D-ISM implementation on the PPU of X-Sat	47
3.7	Comparison of classification results : (a) raw satellite data, (b) ISMC classification, (c) app-ISM classification, (d) EISM classification . . .	50
3.8	Trace of S_w versus image identifier.	51

3.9	Trace of S_b versus image identifier.	52
3.10	Processing time versus image identifier.	53
3.11	Processing time versus different image sizes for the EISMC algorithm. . .	53
3.12	Speed up graph for D-ISMC algorithm on a PC Cluster	54
4.1	Data set in 2D space	58
4.2	256x256 image created with Gaussian Mixture Generator	59
4.3	class description: (a) ground truth map, (b) and overlapping map. . . .	59
4.4	Classification map of the standard Mean Shift algorithm	61
4.5	2D feature space of the classification result the Mean Shift algorithm. . .	61
4.6	Two scenarios handled by the Dominate Filter	63
4.7	Classification map of the modified Mean Shift algorithm.	64
4.8	Classification result of the modified Mean Shift algorithm in 2D space .	64
4.9	Parallelisation steps of D-MeanShift	66
4.10	Plot of Classification Accuracy	68
4.11	Comparison of classification results: (a) raw image data, (b) EISMC clas- sification, (c) Mean Shift classification	70
4.12	Trace of S_w versus image identifier	71
4.13	Trace of S_b versus image identifier	72
4.14	Processing time for different image sizes.	72
4.15	D-MeanShift speedup graph	73
A.1	1024x768 6-band LANDSAT image	84
B.1	517x472 3-band IKONOS satellite image	85
B.2	934x500 3-band LANDSAT satellite image	86
B.3	512x512 3-band LANDSAT satellite image	87
B.4	704x503 3-band MOMS satellite image	88
B.5	271x329 3-band SPOT satellite image	89
B.6	429x436 3-band SPOT satellite image	90
B.7	801x867 3-band SPOT satellite image	91

List of Algorithms

1	k-Means Clustering	13
2	Fuzzy k-Means Clustering	14
3	ISODATA Clustering	17
4	Mean Shift Algorithm	22
5	Mean Shift Clustering Algorithm	24
6	Approximation of the Largest Distance	40
7	D-ISMC Algorithm	42
8	Master-Task of D-ISMC	44
9	Slave-Task of D-ISMC	44
10	Dominate Filter Algorithm	62
11	D-MeanShift Algorithm	66
12	Master-Task of D-MeanShift	67
13	Slave-Task of D-MeanShift	67

List of Tables

1.1	Satellite images involved in this study	4
-----	---	---

Chapter 1

Introduction

1.1 Motivation

Remotely sensed imagery represents a growing source of information in many applications, for example, urban planning, disaster monitoring, and all different kinds of mapping tasks such as updating spatial information of current maps, mapping of large unmapped areas in medium and small scales. One of the key issues for all these applications is the ground cover classification, i.e. the raw satellite data has to be analysed to determine regions with homogenous spectral characteristic and to thematic maps. Ground cover classification is defined as *the extraction of differentiated classes or themes, land use and land cover categories, from raw remotely sensed digital satellite data* [27].

Figure 1.1 shows the flow of a digital image classification system, which is utilised to create thematic maps from raw satellite images. As shown in the figure, after being processed by the feature extraction procedure, the original multi-spectral image is turned into the feature images. Then an unsupervised classification module is added in, and the feature image is “learned” and labelled to output the thematic map [48]. A thematic map is an informational representation of an image which shows the spatial distribution of a particular theme. Themes can be as diversified as their areas of interest. Examples of theme are soil, vegetation, water depth and atmosphere. Inside a theme, sub-themes can be defined, and thus the process of classification needs to be more refined [25].

CHAPTER 1. INTRODUCTION

Figure 1.1: Flow of digital image classification system.

Nowadays the spatial resolution and the spectral resolution have both improved greatly in comparison to then years ago. However, the technology to rapidly process images has not kept pace with the growing volume and the complexity of the provided data. As a result, two interesting issues with respect to image processing in the remote sensing area need to be considered:

- (i) It is too expensive and sometimes impossible to classify the huge amount of remote sensing images supervisely by experts in the corresponding area.
- (ii) Whether the supervised classification methods can meet the increasing amount and complexity of remotely sensed images. Hence, fast unsupervised image classification algorithms with high accuracy are needed badly.

To solve the first issue, highly accurate unsupervised image classification algorithms are badly in need. An unsupervised image classification algorithm is a kind of analytical procedure without too much a-priori information, which can find the corresponding

theme automatically according to the spectral characteristic. To solve the second issue, on one hand, the existing algorithms need to boost their speeds, on the other hand parallel architectures can be taken into consideration to accelerate the speeds of traditional sequential algorithms.

1.2 Scope and Objectives

The objective of this research is *to improve the accuracy of unsupervised satellite image classification algorithms, and to design the corresponding parallel algorithm to improve the processing speed on some parallel architectures.*

To build high speed and highly accurate satellite image classification systems, several parallel architectures were taken into consideration:

- PC Clusters
- Computational Grid
- Parallel Processing Units (PPU) of X-Sat¹

Here, PC Clusters are the widely used low-cost platform, while the computational Grid is the new generation of super computer in the 21st century [22]. The PPU is the high performance computational module on board of X-Sat [11]. In this thesis, a special emphasis is placed on comparing and analysing the classification performance on these three platforms.

According to the objective stated above, a review of different existed unsupervised classification algorithms such as k-means, fuzzy k-means and ISODATA algorithms is carried out. These algorithms are analysed and enhanced not only in processing speed but also in classification accuracy, and then the implementations of these algorithms on different parallel architectures are discussed.

¹X-Sat is the mini-satellite project of Nanyang Technological University

Table 1.1: Satellite images involved in this study

	Satellite Name	Spatial Resolution	Spectral Resolution	Size
A.1	LANDSAT	30x30	6	3000x3000
B.1	IKONOS	4x4	3	517x472
B.2	LANDSAT	30x30	3	934x500
B.3	LANDSAT	30x30	3	512x512
B.4	MOMS	4.5x4.5	3	704x503
B.5	SPOT	20x20	3	271x329
B.6	SPOT	20x20	3	429x436
B.7	SPOT	20x20	3	801x867

To evaluate the classification results of different algorithms, several satellite images from different satellites are involved in the experiments of this study. The information of these images are listed in Figure 1.1, and the images are shown roughly in Appendix A and Appendix B.

1.3 Organisation of this Thesis

The rest of the thesis is organised as follows. Chapter 2 summaries relevant research works on unsupervised image classification algorithms. The considered algorithms include the k-means algorithm, the fuzzy k-means algorithm and the ISODATA algorithm. Several criterion functions for clustering algorithm are also discussed. Chapter 3 introduces the original ISMC algorithm and proposes the Enhanced-ISMC algorithm. Finally a comparison of the performances is presented. Chapter 4 introduces the Mean Shift Algorithm and the corresponding application in unsupervised satellite image classification. The implementations of different algorithms on different parallel architectures are introduced in Chapter 3 and Chapter 4. Chapter 6 summarises the study about unsupervised satellite image classification and outlines future work.

Chapter 2

Basic Concepts and Related Work

In this chapter some basic concepts about digital image classification of remotely sensed imageries are introduced in Section 2.1, and some related unsupervised image classification algorithms such as K-Means and ISODATA algorithms are reviewed in Section 2.2. A special review about several parallel image processing architectures is presented in Section 2.4, and the corresponding parallel architectures are PC cluster, Computational Grid and Parallel Processing Unit of X-Sat.

2.1 Digital Image Classification in Remote Sensing

Satellite remote sensing has been employed as a tool in ground cover analysis for a long time [17]. Indeed, this form of remote sensing is used extensively today and can now benefit from *digital image processing techniques* and *the power of parallel computing architectures* [25].

Digital image classification is one of the most fundamental operations performed on multi-spectral remotely sensed images. It consists of assigning pixel samples and segmenting them to model classes whose spectral signatures are derived from known ground truth data [48]. The overall objective of digital image classification procedure is to *automatically categorise all pixels in an image into land cover classes or themes* [35]. Normally, multi-spectral data is used to perform the classification and the spectral

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

pattern which is presented within the Digital Numbers (DNs) of each pixel. That is, different feature types manifest different combinations of DN values based on their inherent spectral reflectance and emittance properties. In this light, a spectral “pattern” is not at all geometric in character.

Rather, the term “pattern” refers to the set of radiance measurements obtained in the various wavelength bands for each pixel. *Spectral pattern recognition* refers to the family of classification procedures that utilises this pixel-by-pixel spectral information as the basis for automatic land cover classification; *Spatial pattern recognition* involves the categorisation of image pixels on the basis of their spatial relationship with pixels surrounding them [25] [35]. Spatial classifiers might consider such aspects as image texture, pixel proximity, feature size, shape, directionality, repetition and context. These types of classifiers attempt to replicate the kind of spatial synthesis done by the human analyst during the visual interpretation process. Accordingly, they tend to be much more complex and more computationally intensive than spectral pattern recognition procedures [52].

Figure 2.1 shows an example of image classification, the original image is a 7-band Landsat image, but here only six bands are used, because band 6 has a different spatial resolution. The result comes from the Enhanced-ISM algorithm [36], which is discussed in Chapter 4. As can be seen from the classification result, there are four main classes, denoted as red, yellow, blue and green, standing for water area, soil area, forest area and mountain area respectively.

Image classification methods can be divided into *supervised classification methods* and *unsupervised classification methods*. In the former type of classification methods, the image analyst *supervises* the pixel categorisation process by specifying numerical descriptors of the various land cover types present in a scene. Representative sample sites of known cover type (training areas) need to be used to compile a numerical “interpretation key” that describes the spectral attributes for each feature type of interest, and each

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Figure 2.1: Unsupervised classification on 6-band Landsat remotely sensed image

pixel in the data set is then compared numerically to each category in the interpretation key and labelled with the name of the category it “looks most like” [35]. While in the unsupervised approach, the image data is first classified by gathering them into the natural spectral groups or clusters, which present in the scene and then the image analyst determines the land cover identity of these spectral groups by comparing the classified image data to ground reference data [25].

Unsupervised classification can be done automatically by computers. Fast and high accuracy unsupervised classification can meet the expanding volume of new complex satellite images. The X-Sat is a micro-satellite [11], and considering the down-linking burden, many image processing steps have to be done on board of the satellite. So only

unsupervised classification is discussed in this thesis.

2.1.1 Unsupervised Satellite Image Classification

For reasons of the high frequency of update of satellite remote sensing images, high accuracy and high speed unsupervised classification algorithms are necessary in remote sensing systems [31] [44]. There are at least three basic reasons for interest in unsupervised procedures.

- (i) Collecting and labelling a large set of hyper-spectral satellite images in a supervised method is costly.
- (ii) Unsupervised methods can be used in looking for features in satellite image that are useful for categorisation.
- (iii) In the early stages of an investigation it may be valuable to perform exploratory data analysis and thereby gain some insight into the nature or the structure of the remotely sensed data [18].

Unsupervised classification does not employ training data as the basis for classification. Rather, this kind of classification methods involves *some algorithms that examine the unknown pixels in an image and aggregate them into a number of classes based on the natural groupings or clusters present in the image values* [43]. The basic assumption is that the values within a considering cover type should be close to each other in the corresponding measurement space, whereas data in different classes should be well separated comparatively [35].

The classes that result from unsupervised classification are *spectral classes*. Because they are based solely on the natural groupings in the image values, the identities of the spectral classes are not initially known. The analyst must compare the classified data

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

with some forms of reference data (such as larger scale imagery or maps) to determine the identity and informational value of the spectral classes [35]. Thus, in the *supervised* approach, the useful information categories should be first defined and then their spectral separability is examined; while in the *unsupervised* approach the spectrally separable classes should be determined firstly and then their information utilities are defined [52].

Figure 2.2: Spectral classes in two-band image data

Here, the unsupervised approach is illustrated with a two-band data set. Natural spectral groupings in the data can be visually identified by plotting a scatter diagram, as shown in Figure 2.2. “W” indicates the water radiance characteristic, “F” indicates the forest radiance characteristic, and “G” stands for soil. The task of the unsupervised approach is to find these three types separately and correctly as shown in Figure 2.3. There are numerous *clustering* algorithms that can be used to determine the natural spectral groupings present in a data set, and they are discussed in the following sections.

2.1.2 Presentation of Classification Result

The utilities of any image classification systems and the methods to present the classification result are ultimately dependent on the production of the output products that

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Figure 2.3: Unsupervised classification on two-band image data

effectively convey the interpreted information to its end user. Here the boundaries between remote sensing, computer graphics, digital cartography and geographic information system management become blurred, and a virtually unlimited selection of output products may be generated [35]. There are three general forms that are commonly used including *hard copy graphic products*, *tables of area statistics* and *digital data files*.

Graphic Products Since the classification results are in the form of a two-dimensional data array, hard copy graphic output can be easily generated by displaying different colours, tones or characters for each cell in the array according to the land cover category assigned to it [25].

Tabular Data Tabular data is a table which lists summary statistics on the areal extent of the cover types present in a scene or in user defined sub-scene areas [25].

Digital Information Files Digital information files contain the classification results which are recorded on some type of computer storage medium [25].

2.2 Unsupervised Machine Learning and Clustering

Today several different unsupervised machine-learning algorithms are commonly used in remote sensing. The two most frequently used algorithms are *the K-mean algorithm* and *the ISODATA clustering algorithm*, both of which are iterative procedures. In general, they both assign an arbitrary set of initial cluster vectors in the first step, and in the second step each pixel is classified to its closest cluster, while in the third step each new cluster mean vectors are calculated based on all the pixels in this cluster. The second and third steps are repeated until the “change” between the iteration is small enough. The “change” can be defined in several different ways, either by measuring the distances between the mean cluster vectors have changed from one iteration to another or by the percentage of pixels whose cluster labels have been changed between iterations [36] [48].

2.2.1 Mixture Densities

The problem of clustering method can be formulated as to *look for the potential classes from a data set*. Thereby each class has to obey the rule of a specific kind of distribution. In the following parts of this report each class is assumed to follow the Gaussian distribution [18], and the mixture model is defined like this:

- (i) All the samples come from an unknown number k of classes.
- (ii) The prior probabilities $P(\omega_j)$ $j = 1, \dots, k$ for each class are known.
- (iii) The forms for the class-conditional probability densities $p(X|\omega_j, \theta_j)$ $j = 1, \dots, k$ are known.
- (iv) The value for the k parameter vectors $\theta_1, \dots, \theta_k$ are known.
- (v) The category labels are unknown in advance.

The samples used here are selected according to the state of nature ω_j with probability $P(\omega_j)$, and the X is obtained according to the probability law $p(X|\omega_j, \theta_j)$ [24]. And the probability density function for the samples is given as:

$$p(X|\theta) = \sum_{i=1}^n p(X|\omega_j, \theta_j) P(\omega_j) \quad (\text{Eq. 2.1})$$

here $\theta = (\theta_1, \dots, \theta_k)^t$. Obviously, the density function of this form is called as a *mixture density*, the conditional densities $p(x|\omega_j, \theta_j)$ are called the *component densities*, and the prior probabilities $P(\omega_j)$ are called the *mixing parameters* [42]. The basic idea of unsupervised learning here is to *look for these underlining separated distributions correctly one by one*.

2.2.2 K-Means Clustering Algorithm

The k-Means clustering algorithm is one of the simplest unsupervised classification algorithms [44] [7], where k is the expected number of the cluster centroids. It supposes that every data point belongs to one specific class, and the objective class should be the cluster whose centroid is the nearest one to the data point.

The algorithm uses only the squared Euclidean distance $\|x_k - \hat{\mu}_i\|^2$ and finds the nearest mean $\hat{\mu}_m$ to pixel x_k . The approximation $\hat{P}(\omega_i|x_k, \hat{\theta})$ is defined as

$$\hat{P}(\omega_i|x_k, \hat{\theta}) \simeq \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise} \end{cases} \quad (\text{Eq. 2.2})$$

as shown above, if the label of the nearest cluster mean to data point x_k is m , then this data point belongs to this class totally; else this pixel does not belong to this cluster. Then the algorithm repeats the processes of computing the squared Euclidean distance and Eq. 2.2 from time to time, during these steps the new cluster means $\hat{\mu}_1, \dots, \hat{\mu}_k$ are computed after each iteration. Without other information, the expected number of clusters c has to be specified in advance. Here the known number of patterns is denoted

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

as n and the desired number of clusters is c . During the initialisation period, it is common to select c samples randomly from the data set and these samples are regarded as the initial cluster centres. The k-Means algorithm is then shown in Algorithm 1:

Algorithm 1 k-Means Clustering

```

1: begin initialise  $n, c, \mu_1, \mu_2, \dots, \mu_i$ 
2: repeat
3:   classify  $n$  samples according to nearest  $\mu_i$ 
4:   recompute  $\mu_i$ 
5: until no change in or change is less than a threshold
6: end

```

The computational complexity of this algorithm is $O(n \cdot d \cdot c \cdot T)$, where d is the number of feature vectors, c is the number of the expected cluster number and T is the number of iterations. In practice, the number of iterations can be set by the user or decided dynamically, and it is generally much less than the number of samples.

2.2.3 Fuzzy k-Means Clustering Algorithm

In whole iterations of the original k-means procedure, each data point is assumed to belong to some specific cluster, as implied by Eq. 2.2 and by lines 3 and 4 of Algorithm 1 [18]. This condition can be changed by assuming that each sample x_j has some grade or fuzzy memberships to a specific cluster. At first, these “fuzzy memberships” are equivalent to the probabilities $\hat{P}(\omega_i|x_j, \hat{\theta})$, where $\hat{\theta}$ is the parameter vector for the membership functions [37].

The fuzzy k-means clustering algorithm is to seek a minimum of a heuristic global cost function [18]:

$$J_{fuz} = \sum_{i=1}^c \sum_{j=1}^n [\hat{P}(\omega_i|x_j, \hat{\theta})]^b \|x_j - \mu_i\|^2 \quad (\text{Eq. 2.3})$$

here b is a free parameter chosen to adjust the “fuzzy factor” between the different clusters. If b is set to 0, J_{fuz} is merely the sum-of-squared errors criterion with each

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

pattern assigned to a single cluster, and if $b > 1$, this criterion allows each pattern to belong to multiple clusters.

The probabilities for the cluster membership of each point can be normalised as

$$\sum_{i=1}^c \hat{P}(\omega_i|x_j) = 1, \quad j = 1, \dots, n \quad (\text{Eq. 2.4})$$

The dependence on $\hat{\theta}$ is not shown here for simplicity. $\hat{P}_j$ is used to denote the prior probability of $\hat{P}(\omega_j)$. To get the minimum for J_{fuz} , the following equations are derived:

$$\partial J_{fuz}/\partial \mu_i = 0 \quad \text{and} \quad \partial J_{fuz}/\partial \hat{P}_j = 0 \quad (\text{Eq. 2.5})$$

Then the component μ_j is shown in Eq. 2.6 [18]:

$$\mu_j = \frac{\sum_{j=1}^n [\hat{P}(\omega_i|x_j)]^b x_j}{\sum_{j=1}^n [\hat{P}(\omega_i|x_j)]^b} \quad (\text{Eq. 2.6})$$

and

$$\hat{P}(\omega_i|x_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{r=1}^c (1/d_{rj})^{1/(b-1)}} \quad \text{and} \quad d_{ij} = \|x_j - \mu_i\|^2 \quad (\text{Eq. 2.7})$$

Generally speaking, the J_{fuz} criterion is minimised when the cluster centroid μ_j are very near to those points that have high estimated probability of being in cluster j . Because Eq. 2.6 and Eq. 2.7 rarely have analytic solutions, the cluster means and point probabilities are estimated iteratively [18]. The fuzzy k-Means algorithm is shown in Algorithm 2:

Algorithm 2 Fuzzy k-Means Clustering

- 1: **begin** initialise $n, c, b, \mu_1, \dots, \mu_c, \hat{P}(\omega_i|x_j), i = 1, \dots, c; j = 1, \dots, n$
 - 2: normalise $\hat{P}(\omega_i|x_j)$
 - 3: **repeat** recompute μ_i and $\hat{P}(\omega_i|x_j)$
 - 4: **until** change in μ_i and $\hat{P}(\omega_i|x_j)$ is small enough
 - 5: **end**
-

2.2.4 ISODATA Algorithm

From the above two sub-sections, both the k-Means algorithm and the Fuzzy k-Means algorithm need a fixed cluster number to be set in advance, and this number is not changed during the whole processing. So if this number is not set properly, the classification result may be quite different. ISODATA is such an unsupervised classification algorithm that it can determine the cluster number dynamically, and ISODATA stands for “*Iterative Self-Organising Data Analysis Technique*” [33].

Iterative repeatedly performs an entire classification (outputting a thematic raster layer) and recalculates statistics.

Self-Organising refers to the way in which it locates the clusters that are inherent in the data. The ISODATA clustering method uses the minimum spectral distance formula to form clusters.

Compared with the K-means algorithm, the ISODATA algorithm has some further refinements by splitting and merging clusters. Clusters are merged if either the number of members (pixel) in a cluster is less than a certain threshold or if the centres of two clusters are closer than a certain threshold. Clusters are split into two different clusters if the standard deviation of that cluster exceeds a predefined value and the number of members (pixels) is twice the threshold for the minimum number of members.

The ISODATA algorithm consists of a k-means clustering algorithm embedded within a loop and heuristics for cluster splitting and merging [10]. The algorithm uses a distance measure, d_I , derived from an assumed multi-variate Gaussian distribution, for which the probability distribution function of class i , $1 \leq i \leq M$, is given by:

$$p\{x|i\} = \frac{1}{(2\pi)^{P/2}|C_i|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu_i)^t C_i^{-1} (x - \mu_i)\right] \quad (\text{Eq. 2.8})$$

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Here x is a P -Dimensional vector, M is number of classes, containing the values in the P input channels, μ_i is the mean vector for class i , C_i is the covariance matrix of class i and $|\cdot|$ denotes determinant. A pixel is to be assigned to the class whose cluster centre it is the closest to (or with the highest likelihood). Assume that the image contains N pixels, and the pixels are labelled by n , $1 \leq n \leq N$. At a given iteration, the pixels in class i are defined by their pixel labels $I_i = \{n; X_n \in \text{class } i\}$. Denoting the number of pixels in class i as $|I_i|$, the class means and covariances can be estimated as

$$\mu_i = \frac{1}{|I_i|} \sum_{m \in I_i} X_m \quad (\text{Eq. 2.9})$$

$$C_i = \frac{1}{|I_i|} \sum_{m \in I_i} (X_m - \mu_i)(X_m - \mu_i)^t = \frac{1}{|I_i|} \sum_{m \in I_i} X_m X_m^t - \mu_i \mu_i^t \quad (\text{Eq. 2.10})$$

At each iteration, the current labelling is used to calculate the means and covariances as in Eq. 2.9 and Eq. 2.10. Each pixel in the image is then reassigned to a class by minimising the distance measure.

If the distance between two cluster centres is too small (i.e., smaller than a pre-defined threshold), these two clusters are merged into one cluster; if some cluster is too large (i.e., the standard deviation of the pixel values in that class is larger than a pre-defined threshold), this cluster is splitted into two. Following convergence a splitting or merging of the clusters is performed and the above procedure is repeated with the new number of clusters. The process stops after a fixed number of iterations or when the proportion of pixels with changing class labels fall below a fixed threshold (or possibly when the proportion of pixels in each class which changes classes falls below a threshold).

Initialisation is very important to the ISODATA algorithm. One approach is to assign the class centres randomly. This is the most non-committal approach and assumes no advanced knowledge of possible clusters. Another approach is that the pixels are partitioned initially by splitting the range of coherence in the image into M equal intervals,

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

became the coherence is a major information source. In the Eq. 2.9 and Eq. 2.10 there are not enough pixels in this class to significantly perturb the statistics of the majority class [45]. Hence the minority class can never establish its separate identity. To prevent this happen, prior knowledge should be taken in during the seeding progress, i.e. set up the initial partition to reflect understanding of the portions of the Landsat volume which can be occupied by various physical classes [52]. The algorithm is outlined in Algorithm 3:

Algorithm 3 ISODATA Clustering

```

1: begin
2:   Select  $N_c$  ( $N_c < M$ ) initial clusters arbitrary
3:   Select centres of each cluster  $\mu_j$  ( $\forall j \in C$ )
4:   while (stopping criteria is not satisfied) begin
5:     a K-means step
6:     Choose the clusters eligible for split or merge
7:     Update the number of clusters  $N_c$ 
8:     Determine the current centres,  $X_j$  ( $\forall j \in C$ )
9:   end while
10: end

```

2.2.5 ISMC Algorithm

The previously published *Improved Splitting-and-Merging Cluster (ISMC)* algorithm is a hybrid classification method for automatic clustering of satellite-observed scenes, and provides a reliable and accurate solution to the problem of *unsupervised image classification* [45]. It uses a partial clustering algorithm augmented by a hierarchical split-and-merge step at each iteration and dynamically computes the image-specific split-and-merge thresholds, and thus a high classification accuracy is gained [36].

In the following steps (this part is to review the basic idea of the pre-published ISMC algorithm [45]), the input vectors y are assumed to be $n_p \times 1$ column vectors, whereby n_p is the dimension of the input image, i.e. the number of spectral bands.

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Step 1) Distribute the initial clusters:

The ISMC algorithm regards the whole data set as a single cluster so the number of clusters is 1, and then computing the mean value of this cluster m_1 .

$$m_1 = \frac{\sum_{n=1}^N y_n}{N} \quad (\text{Eq. 2.11})$$

Here N stands for the amount of the input vectors.

Step 2) Assign input vectors to the initial clusters:

Each input vector y in the scene is assigned to the nearest cluster centres by computing the Euclidean distance from the vector to each cluster centre.

$$d_k = (y - m_k)^T (y - m_k) \quad k = 1 \cdots K \quad (\text{Eq. 2.12})$$

where K is the total number of clusters, and m_k denotes the cluster centre of the k th cluster. Each vector is assigned to a specific cluster according to the minimum d_k .

Step 3) Compute the between-cluster scatter matrix S_b :

$$S_b = \sum_{k=1}^K n_k m_k - m(m_k - m)^T \quad (\text{Eq. 2.13})$$

where n_k is the number of vectors in the k th cluster, and m is the mean vector of the entire data set.

Step 4) Compute the mean vector of the entire data set:

$$m = \frac{1}{N} \sum_{k=1}^K n_k m_k \quad (\text{Eq. 2.14})$$

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Step 5) Check for convergence:

The trace of the between cluster scatter matrix $T_r(S_b)$ is used here to check whether the convergence is met, which is a reliable measure of between-cluster scatter [45]. Convergence occurs when there is negligible change in $T_r(S_b)$ between iterations, for example, if

$$\frac{T_r(S_b) - T_r(S'_b)}{T_r(S_b)} < tol \quad (\text{Eq. 2.15})$$

here S'_b is the between-cluster scatter matrix for the previous iteration, and tol is a predefined numerical tolerance. The value for tol used here is 0.05.

Step 6) Split the clusters that need to be split:

Splitting of a cluster occurs if d_m , the Euclidean distance between component-wise cluster maximum and minimum vectors

$$d_m = (y_{max} - y_{min})^T (y_{max} - y_{min}) \quad (\text{Eq. 2.16})$$

exceeded the pre-defined splitting threshold τ_s . Then each vector in the cluster is assigned to one of the two new clusters, depending on whether it is closer to y_{max} or y_{min} .

Step 7) Repeat Step 6 until no clusters need to be split.

Step 8) Remove all empty clusters.

Step 9) Merge the clusters that need to be merged.

For every pair of clusters, compute the Euclidean distance d_m between the cluster mean vector using

$$d_m = (m_i - m_j)^T (m_i - m_j) \quad (\text{Eq. 2.17})$$

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

where m_i and m_j are the cluster mean vectors for the i th and j th cluster, respectively. If d_m is less than the merging threshold τ_m , combine all vectors in the i th and j th cluster into a single cluster.

Step 10) Repeat Step 9 until no clusters need to be merged.

Step 11) Based on the current K cluster centres, repeat the whole process beginning with Step 2.

The thresholds in the ISMC algorithm (τ_s and τ_m) are computed by multiplying the user-defined values **SPLIT** and **MERGE** by a dynamic, image-specific scaling factor, s

$$s = \beta \sum_{i=1}^{n_p} (max_i - min_i)^2 \quad (\text{Eq. 2.18})$$

where $\beta = 1.0 \times 10^{-4}$, n_p is the number of components in the information vector of each pixel, and max_i and min_i are the respective maximum and minimum values in the i th element of the information vector. Computations for τ_s and τ_m follow

$$\tau_s = \mathbf{SPLIT} \times s \quad (\text{Eq. 2.19})$$

$$\tau_m = \mathbf{Merge} \times s \quad (\text{Eq. 2.20})$$

The flowchart of the ISMC algorithm is shown in Figure 2.4.

2.2.6 Mean Shift Clustering Algorithm

This sub-section is to explain the basic mean shift theory from [14] [15]. Assume that the probability density function $P(x)$ of the feature vector x ($x \in R^p$) is unimodal. Feature vectors y s are defined as all the feature vectors which are lying within r distance from x . In other words, y are lying in a sphere S_x of radius r , centred at x ($\|y - x\| < r$). Then, the expected value of the vector $z = y - x$ is

$$\mu = E[z|S_x] = \int_{S_x} (y - x)p(y|S_x)dy = \int_{S_x} (y - x) \frac{p(y)}{p(y \in S_x)} dy \quad (\text{Eq. 2.21})$$

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Figure 2.4: Flowchart of the ISMC algorithm.

if S_X is small enough, $p(y \in S_X)$ can be approximated according to

$$p(y \in S_X) = p(x)V_{S_X} \quad (\text{Eq. 2.22})$$

where $V_{S_X} = c.r^p$ is the volume of the sphere. The first order approximation of $p(y)$ is

$$p(y) = p(x) + (y - x)^T \nabla p(x) \quad (\text{Eq. 2.23})$$

where $\nabla p(x)$ is the gradient of the probability density function in x . Then

$$\mu = \int_{S_X} \frac{(y - x)(y - x)^T}{V_{S_X}} \frac{\nabla p(x)}{p(x)} dy \quad (\text{Eq. 2.24})$$

since the first term term vanishes. The value of the integral if

$$\mu = \frac{r^2}{p + 2} \frac{\nabla p(x)}{p(x)} \quad (\text{Eq. 2.25})$$

or

$$E[x|x \in S_x] - x = \frac{r^2}{p + 2} \frac{\nabla p(x)}{p(x)} \quad (\text{Eq. 2.26})$$

The left side of Eq. 2.26 is called the mean shift vector. The mean shift vector is the vector of the difference between *the local mean* and *the centre of the window*. The mean shift vector preserves three important properties:

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

- (i) The mean shift vector is small for regions which have large $P(x)$ and small $\nabla p(x)$ and large for regions which have low $P(x)$.
- (ii) The direction of the mean shift vector corresponds to the gradient direction of the probability density function.
- (iii) At the mode, which corresponds to the maximal of the probability density function, the mean shift vector is zeros.

The first two properties can be used to develop a gradient based search methods for the modes during nonparametric clustering because the mean shift vector can be viewed as an estimation of the gradient of the probability density function. The last property guarantees the convergence of the algorithm.

The Mean Shift algorithm itself is an iterative method to find modes (the centres of the regions of high concentration) of multi-dimensional data sets. The mean shift clustering algorithm can be derived from the basic Mean Shift theory as Algorithm 4.

Algorithm 4 Mean Shift Algorithm

- 1: **begin initialise**
 - 2: Choose a radius h as the size of search window.
 - 3: Choose an initial location of the search window.
 - 4: Apply the Mean Shift procedure to find the mode from the initial location.
 - 5: Remove the item between the initial location and $mode + radius$.
 - 6: Set the initial location to $mode + radius$.
 - 7: Repeat steps 4, 5, 6 until the number of the remaining items is less than a preset threshold.
 - 8: **end**
-

As shown above, at the first step the algorithm choose a radius h as the size of the search window, and this window size is used to compute the Mean Shift vector in the area from the initial location to $mode + radius$. This radius defines the search range during computing the Mean Shift vector, so it is very important to choose a proper radius. The algorithm deletes all the items from the initial location to $mode + radius$ every time when

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

a new mode is found. At last the whole process stops when the number of the remaining items is less than a preset threshold. The algorithm can be applied to multi-dimensional data set, while here a 1D data set is used to demonstrate for simplicity. At first the histogram of the data set is built as Figure 2.5.0, and then the initial location is set at the origin. The search window now is defined as from 0 to h , and the algorithm begins to search for the mode from this windows. When the mode is found the bins between the current initial location (the origin) to $mode + radius$ are deleted, and the initial point is reset to $mode + radius$ again. This iteration is repeated (from Figure 2.5.1 to Figure 2.5.5) until convergence.

Figure 2.5: Iterative modification of the histogram by the Mean Shift algorithm

This algorithm can be applied in unsupervised satellite image classification in the way described in Algorithm 5. At first an input satellite is mapped into the feature space from image domain, and here the histogram is used as the feature. Then an adequate number is defined as the radius of the search window, and the Mean Shift algorithm is applied to find the modes in the feature space. When all modes are found these modes will be

Algorithm 5 Mean Shift Clustering Algorithm

- 1: **begin initialise**
 - 2: Map the image domain into the feature space.
 - 3: Define an adequate number of search window and set the initial to origin.
 - 4: Find centres of high density regions by applying the Mean Shift Algorithm.
 - 5: Find regions in the image domain corresponding to high density regions in the feature space.
 - 6: Do some postprocessing on the image domain regions based on predefined constraints and prior assumptions.
 - 7: **end**
-

mapped back to the image domain, and some postprocessing steps such as assigning every pixel to the nearest mode are applied. At this stage each sample point is associated with a cluster centre. To allocate the data points a k-nearest neighbour technique is employed, i.e., each data point belongs to the cluster defined by the majority of its k-nearest sample points.

The clustering algorithm makes use of three parameters: the searching sphere radius h which controls the sensitivity of the decomposition, the initial location for searching, and the threshold corresponding to the minimum number of the remaining items. This algorithm is simple and fast, and for high-dimensional satellite image (for example 6-band landsat image), some dimension reduction techniques (such as PCA) can be used. While PCA may lost some information of the satellite, so a tradeoff need to be considered here between processing speed and classification accuracy.

2.3 Criterion Functions for Clustering Algorithm

From the discussion of the unsupervised classification algorithm, a major issue comes out in the design of the clustering algorithms or unsupervised classification algorithms: *what is the criterion function to be optimised* [18]? Suppose that there is a data set $\mathcal{D} = \{x_1, \dots, x_n\}$ with n samples, and the objective cluster algorithm is to partition it into c disjoint subsets $\mathcal{D}_1, \dots, \mathcal{D}_c$. Each subset represents a cluster, and the

samples in the same cluster should be more similar to each other than they are to the samples in other clusters, at the same time the samples in different clusters should be different to each other as much as possible. One reasonable method to make this into a well-defined problem. A *criterion function* that measures the clustering quality of any partition of the data at each iteration of the refinement process is defined. Thus the clustering problem is to be simplified as to find the specific partition that maximise the criterion function.

2.3.1 Sum-of-Squared Error Criterion

The sum-of-squared-error criterion is the simplest and most widely used criterion function for clustering. Let n_i be the number of samples in cluster $\mathcal{D}_i$ and m_i be the mean value of the i th cluster, where

$$m_i = \frac{1}{n_i} \sum_{x \in \mathcal{D}_i} x \quad (\text{Eq. 2.27})$$

The sum-of-squared error is defined as

$$\mathcal{J}_e = \sum_{i=1}^c \sum_{x \in \mathcal{D}_i} \|x - m_i\|^2 \quad (\text{Eq. 2.28})$$

As described in Eq. 2.28, for a given cluster $\mathcal{D}_i$, the mean vector m_i is regarded as the best representative of the samples in that cluster [7] because it minimises the sum of the squared length of the “error” vector $x - m_i$ in $\mathcal{D}_i$ [18]. $\mathcal{J}_e$ acts as the measurement of the total squared error which represents the n samples $x_1, \dots, x_n$ by the c cluster centres $m_1, \dots, m_c$. The value of $\mathcal{J}_e$ depends on how to group the samples into clusters and the number of clusters. The optimal partitioning is defined as one that it minimises the $\mathcal{J}_e$ criterion. This type of clusterings is often called *minimum variance partitions* [18].

2.3.2 Classification Accuracy

The classification accuracy A of an image classification algorithm depends on the number of samples correctly classified and it is evaluated by the formula:

$$A = \frac{t}{n} \times 100\% \quad (\text{Eq. 2.29})$$

Where t is the number of sample pixels classified correctly, and n is the total number of sample pixels. While for the real satellite image it is very hard to judge whether a point is classified correctly or not because of lacking of the ground truth. So here a Gaussian mixture generator (the matlab implementation is shown in Appendix C) is used to create the data set for testing. The reason why *mixgen* is used is because it can create a data set also with the corresponding ground truth, with which the classification accuracy can be evaluated.

2.3.3 Scatter Criteria

2.3.3.1 The Scatter Matrices

There are some other kinds of criterion functions which are derived from the scatter matrices, for example *the between cluster scatter matrix* and *the within cluster scatter matrix*, and these functions can do very well in discriminant analysis [18].

As described earlier, the mean value of the i th cluster m_i is defined as

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x \quad (\text{Eq. 2.30})$$

The total mean vector of the whole data set m is

$$m = \frac{1}{n} \sum_D x = \frac{1}{n} \sum_{i=1}^c n_i m_i \quad (\text{Eq. 2.31})$$

The scatter matrix for the i th cluster S_i is define as

$$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^t \quad (\text{Eq. 2.32})$$

The *within-cluster scatter matrix* S_w is

$$S_w = \sum_{i=1}^c S_i \quad (\text{Eq. 2.33})$$

and the *between-cluster scatter matrix* S_b is

$$S_B = \sum_{i=1}^c n_i (m_i - m)(m_i - m)^t \quad (\text{Eq. 2.34})$$

and the total scatter matrix S_T is

$$S_T = \sum_{x \in D} (x - m)(x - m)^t \quad (\text{Eq. 2.35})$$

It is obvious that the total scatter matrix is the sum of the *within-cluster scatter matrix* and the *between-cluster scatter matrix*:

$$S_T = S_W + S_B \quad (\text{Eq. 2.36})$$

From the above the total scatter matrix does not depend on how the set of the samples is partitioned while it only depends on the total set of samples. The within-cluster scatter matrices and between-cluster scatter matrices are dependent on the partitioning [18]. From the definitions of the two scatter matrixes, it can be easily concluded that there is an exchangeable relationship between these two matrices: as the between-cluster scatter goes up the within-cluster scatter goes down, and vice versa.

A scalar measure of the “size” of a scatter matrix is needed to be used as a criterion to optimise during the clustering process [45]. The traces of these scatter matrixes are taken into consideration here.

2.3.3.2 Trace Criterion

The trace of a matrix is defined as “the sum of its diagonal elements”, and it is regarded as one of the simplest scalar measure of a scatter matrix [57]. The trace of a scatter matrix is proportional to the sum of the variances in the coordinate directions, so it

serves as the measure of the square of the scattering radius. Thus, two obvious criterion functions to minimise are the trace of S_W and the trace of S_B . Ideally, the samples in a given cluster should be as homogeneous as possible, thus S_W should be as small as possible; likewise, the samples in different clusters should be as distinct from one another as possible, thus S_B should be as large as possible [45]. Above all, these two criteria are defined like in Eq. 2.37 and Eq. 2.38

$$tr[S_W] = \sum_{i=1}^c tr[S_i] = \sum_{i=1}^c \sum_{x \in D_i} \|x - m_i\|^2 \quad (\text{Eq. 2.37})$$

$$tr[S_B] = \sum_{i=1}^c n_i \|m_i - m\|^2 \quad (\text{Eq. 2.38})$$

2.4 Parallel Image Processing Platforms

Nowadays with the development of science and technology in remote sensing domain, the spatial and spectral resolutions are both improved a lot. As a result thousands of Gigabytes images are produced every day [27], so it becomes to be quite necessary for high speed image processing platforms. In this section PC clusters, Computational Grid and Parallel Processing Units of X-Sat are reviewed, and special emphasises are placed on the data processing frameworks and the programming interfaces.

2.4.1 Image Processing Platform on PC Cluster

With the advent of high-speed networks and the availability of powerful high-performance computers, cluster architectures using off-the-shelf components have emerged as one of the most cost-effective platforms for computational intensive applications [36]. Beowulf [46] [40] is such a project to produce high-performance parallel computers using clusters of personal computers and freely available software such as the Linux operating system. It has been proven that the price/performance ratio of the Beowulf system is attractive for academic and research organisations [29].

Message passing is a common programming paradigm widely used on modern parallel computers, especially on PC Clusters. The most important and popular message-passing software packages are MPI and PVM [30], which are discussed in the following subsection.

2.4.1.1 PVM and MPI

PVM (Parallel Virtual Machine) [39] is a self-contained and public-domain software system that was originally designed to enable a cluster of heterogeneous computers to be used as one large-scale, message-passing parallel computer. It integrates a set of software tools and libraries that emulate a general-purpose, flexible, heterogeneous concurrent computing framework on interconnected computers of varied architecture [8].

The Message Passing Interface (MPI) [9] is a standard specification for a library of message-passing functions [30]. The MPI package provides a public-domain, platform-independent standard of message-passing library. The package is a language-independent one and for example it provides C and Java bindings [38]. As the standard in message-passing programming, MPI has gained wide acceptance in the parallel computing community.

2.4.1.2 Parallel Image Classification Platform on PC Cluster

The task of designing a parallel image classification platform for the (RS)2I project ((*RS*)2I stands for the Retrieval System for Remotely Sensed Imagery based on PC Clusters and PPU of X-Sat) can be divided into two part:

- (i) How to share image data among cluster nodes?
- (ii) How to exchange information during processing?

On considering the first question mentioned above a Network File System (NFS) [51] server is used in this platform. NFS is to allow remote hosts to mount partitions onto a

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

particular system and use them as if they were local file systems. This property allows files to be organised in a central location, and provides the functionality of allowing authorised users continuous access to them. (See Figure 2.6).

Figure 2.6: Shareing data between cluster nodes via NFS

There are many programming languages and message passing libraries that can solve the second question, for example PVM [39], MPI [9], Parallel Matlab [6, 2, 5]. All these have the flexibilities for different architectures and different communication modes, and they can also provide several mechanisms to perform point-to-point and collective communications. These packages also support both *single program multiple data* (SPMD) and *multiple program multiple data* (MPMD) modes of parallel/distributed programming [30].

A major application for PC-Clusters is to process the hyper-spectral images in remote sensing, where, because of the high dimensionality of the data, most of the machine-based exploitation procedures are computation-intensive. In the message-passing programming model, such a big problem can be divided into some smaller parts, which are then assign to each computing process, and the different processes exchanges information via the message passing library (As shown in Figure 2.7).

Figure 2.7: Message passing within cluster

2.4.2 Image Processing Platform on Computational Grid

Computational Grid is the new generation of super computer in the 21st century, and it has already attracted the consideration of both the academic community and the industrial community globally. In 1998 Ian Foster and Carl Kesselman defined the *computing grid* as “a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities” [21]. The emphasises of this definition is placed on accessing the computing resource, data and services provided by the Grid when they are needed. Later *Virtual Organisation (VO)* got its definition as “a set of individuals and/or institutions defined by the sharing rules” [23]. With the concept of *VO*, the definition of the grid is refined as “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations” [23]:

The design objectives, target applications, and machine environment determine the architecture of a Grid system. The design objectives are grouped into three different types: improving application performance, data access, and enhanced services [?].

2.4.2.1 Globus Toolkit

The Globus project [1, 20] is developed by the Globus project team to build computational grid and grid-based applications. Globus Toolkit has emerged itself as *the de facto standard of Grid computing*, and it mainly has the following modules:

Grid Security Infrastructure (GSI) [22] enables secure authentication and communication over an open network, for example the Internet.

Globus Resource Management System (GRMS) is developed to support resource allocation and reservation in the computational grid.

Globus Metacomputing Directory Service (MDS) provides the necessary tools to build an LDAP based information infrastructure for computational grids.

GridFTP is a high-performance, secured, reliable data transfer protocol optimised for high-bandwidth wide-area networks [28].

2.4.2.2 MPICH-G2 – A Grid-enabled MPI

MPICH-G2 [3] is a complete implementation of the MPI-1 standard, and uses Globus Toolkit Version 2 services to support efficient and transparent execution in heterogeneous grid environments. The MPICH-G2 toolkit uses a range of the services of Globus Toolkit to handle the various complex issues that arise in heterogeneous, multi-site grid environment, for example the cross-site authentication, the need to deal with multiple schedules with different characteristics, coordinated process creation, heterogeneous communication structures, executable staging, and collection of standard output [32].

2.4.2.3 Parallel Image Processing Platform on Grid

The computational Grid system used in the Grid-(RS)2I¹ project is based on the Globus Toolkit 2.4 [26] and consists of two separate clusters. One of them is a Linux Cluster: HPC-PDPM (11 nodes (22 CPUs): Intel P4 2.4 GHz, Kernel Version: 2.4.18-27.7.xsmp), which is interconnected with a Myrinet with a bandwidth of 1 Gbps; the other one is a

¹*Grid-(RS)2I* stands for the Retrieval System for Remotely Sensed Imagery based on Computational Grid

SUN Cluster (10 nodes (20 CPUs): SUN Ultra Sparc, Kernel Version: Generic_112233-03), which is interconnected with a high speed Ethernet with a bandwidth of 100 Mbps. These two clusters are connected with a Ethernet whose bandwidth is 100 Mbps. These two clusters forms into a hierarchical grid system [13], and the architecture is shown in Figure 2.8.

Figure 2.8: Architecture of the used hierarchical grid system.

Obviously in Figure 2.8, there is a large gap between the speed of the intra-cluster network (1-Gbps Myrinet) and that of the inter-cluster network (100-Mbps Ethernet). To make full use of the network bandwidth and the CPU cycles of the hierarchical grid system, two kinds of hierarchical grid system are to be used in the Grid-(RS)2I projects.

The first kind of parallel programming environment is shown in Figure 2.9, which can be divided into two layers. The upper layer is an MPICH-G2 [3] [19] layer. The programs run on this layer take charge of the inter-cluster communication and assignment of jobs between different clusters, and they run only on the front ends of different clusters. The lower layer is an MPICH [4] layer. The programs run on this layer is the computing power provider of the local cluster, and they run on all nodes within the cluster. This architecture provides a robust platform to make full use of (fast) intra-cluster communication and effective CPU cycles. Data can be exchanged between the MPICH-G2 program and the MPICH program with the Inter-Process Communication (IPC) mechanism [47] inside the front-end node of each cluster (see Figure 2.9).

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Figure 2.9: Hierarchical parallel programming environment consisting of MPICH-G2 and MPICH programs [13].

The second kind of parallel programming environment can also be divided into two layers. The upper layer is an MPICH-G2 [19, 3] layer, too, and the front-end node of each cluster acts as a resource provider which mainly provides the computing power of the Computational Grid. The lower layer consists of an SUN Grid Engine [49] that runs on all nodes within a cluster. This allows the full use of the fast intra-cluster communication and the emergence of a single system image [41] of a specific cluster. Data can be exchanged between the MPICH-G2 layer and the SUN Grid Engine layer by submitting a specific job to the front-end node of each cluster(see Figure 2.10).

Although the Grid can provide huge computing power, but for the time being its response time is still quite slow and the parallel implantations of the different algorithms on the Computational Grid are almost the same as those on PC Cluster. So in this thesis the Computational Grid is only given out as a candidate parallel image processing platform, no further discussion is made.

2.4.3 Parallel Image Processing Platform on X-Sat

The Parallel Processing Unit (PPU) is the high performance parallel computer on board of the micro-satellite X-Sat, and is designed as a reconfigurable parallel architecture for

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

Figure 2.10: Hierarchical parallel programming environment consisting of MPICH-G2 and SUN Grid Engine.

space-borne on-board processing that combines the advantages of powerful computational resources with a straightforward software development [12]. Its main objective is to relax the down-link burden between satellite and ground station through on-board preprocessing like compression and ground cover classification.

Figure 2.11 shows the architecture of the PPU. As shown the PPU consists of two ProASIC APA600s from Actel that host in total 20 processing nodes (PN) connected by high-speed dedicated parallel links [12]. The requirement of multiple CPUs is to achieve the required processing power as well as reliability through redundancy, which means when some CPUs die the system can use other candidate CPUs. Each COTS PN is fairly susceptible to the damaging effects of radiation in orbit, and has a not insignificant probability of ceasing to function during the satellite's lifetime, however the probability of all PNs doing so is much lower.

One of the main objective to use FPGAs here is to provide a flexible network topology among the PNs and the processing capability for realtime task. Each of the PNs consists of an Intel StrongARM SA-1110 processor and 64MBtyes SDRAM. In addition each of the FPGAs is connected to triple voting flash chips that hold the software (i.e., Operating System and some initialisation data) for the PNs. The PPU is also connected with a mass

CHAPTER 2. BASIC CONCEPTS AND RELATED WORK

storage unit (RAM-DISK), which stores the images to process, via two high-speed 160 Mbit/s LVDS connections while the control through the OBC is handled by a redundant controller area network (CAN) [12].

Figure 2.11: Architecture of Parallel Processing Unit [12]

The advantage of this parallel architecture is that it employs a loosely coupled mode which is similar to a PC Cluster, and this makes it straightforward to port parallel application from any PC cluster to the PPU.

Chapter 3

Enhanced ISMC Image Classification Algorithm

The ISMC algorithm has already been reviewed in Chapter 2.2.5, in this chapter to improve the classification performance the Enhanced ISMC algorithm is proposed. To reduce the computing burden in calculating the largest distance in a data set, an algorithm whose complexity is $O(n)$ to find out the approximate largest distance is introduced in Section 3.2. The evaluation of the classification result is presented in Section 3.5. Section 3.3 and Section 3.4 introduce the design of the parallel ISMC algorithm, and also the implementation on PC cluster and PPU.

3.1 Enhanced ISMC Algorithm

The ISMC algorithm distinguishes itself to other clustering algorithms in better partitioning of the data and improved computational efficiency [45]. Although the ISMC algorithm can do well in automatic clustering of satellite-observed scenes, there are still some parts that need improvement. For example, after every splitting step, every cluster does not converge as much as possible. As shown in Figure 3.1, from the first image, it is obvious that this cluster is too large and should be divided into two clusters. In the second image, the ISMC algorithm finds the maximum and the minimum y_{max} and y_{min}

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Figure 3.1: Classification by the ISMC algorithm.

Figure 3.2: Classification by the Enhanced-ISM algorithm.

of the cluster, and divides the big cluster into two according to the hyperplane in the middle of these two points. Then the algorithm regards y_{max} and y_{min} as the two new cluster centres. After this the ISMC algorithm takes this as the basic and proceeds with the processing. The basic idea of the enhancement used here is to make every cluster as dense as possible [36].

The Enhanced-ISM algorithm uses a k-means step to ensure the density of every cluster. As shown in Figure 3.2, after the first step a large cluster is found to be split. The algorithm then finds the two farthest data points y_{max} and y_{min} in the data set and divides this cluster with the middle hyperplane of these two points; In the following step, a k-means procedure is introduced to calculate the two new cluster centres.

There is a similar problem when two too near clusters are merged, when merging, the ISMC algorithm only regards an arbitrary one of the two considering cluster centres as

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

the new cluster for further processing. Thus another k-means step is also added after each merge step in the Enhanced-ISM algorithm. This ensures that every cluster is separated to all others as much as possible, and the pixels in the same cluster are as much alike as possible. The flowchart of the Enhanced-ISM algorithm is presented in Figure 3.3.

The destination of making dense clusters is achieved, but the computing burden is increased as several k-means steps are introduced.

Figure 3.3: Flowchart of the Enhanced-ISM algorithm.

3.2 Approximation of the Largest Distance

In unsupervised image classification algorithms or clustering algorithms, a large part of the computational burden comes from the calculation the pair-wise distance in the whole data set. The traditional method is to calculate the distance between every two points in this data set. The algorithm is straightforward to implement but its computing complexity is $O(n^2)$. To solve this problem many new algorithms were introduced, for example

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

the hyper-cube algorithm used in the ISMC algorithm with a computing complexity of $O(n \times \log n)$ [45]. Here, a new kind of algorithm whose computing complexity is $O(n)$ is presented, and it is described in Algorithm 6:

Algorithm 6 Approximation of the Largest Distance

- 1: **begin**
 - 2: Calculate the mean of the given data set, and mark it as *Centre*;
 - 3: Find the farthest point in the data set to point *Centre*, and mark it as *P1*;
 - 4: Find the farthest point in the data set to point *P1*, and mark it as *P2*;
 - 5: Regard the largest distance in the data set as the distance between *P1* and *P2*;
 - 6: **end**
-

The data set in Figure 3.4 (the blue point) is a 2-dimensional Gaussian distribution [54] consisting of 100 points. Firstly, the algorithm calculates the centre - the mean value of the data set, and then the farthest point to the centre, *P1*. In the next step, the algorithm gets the farthest data point to point *P1*, assume it as *P2*, and regards the distance between *P1* and *P2* as the largest pair-wise distance in the data set. It is trivial to prove that this distance is not the largest one all the time, yet most of the time it is the largest distance. Experiments showed that the algorithm can find the largest pair-wise distance in 96 out of 100 times. The accuracy gained with the approximation method is sufficient since the iterative process is repeated and the errors are averaged out.

3.3 Design of the D-ISM C Algorithm

As the development of sensor technology and the increasing of communication bandwidth between satellite and the ground station, the resolution of remotely sensed images and the updating speed make the volume of image database cumulate day after day, so a method to processing these images rapidly and accurately is desired badly in this area. A natural solution to this problem is to use parallel computers that can run computing tasks parallelly, and nowadays there are many parallel architectures that can provide

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Figure 3.4: Approximative method for finding the largest pair-wise distance.

huge computing power such as PC Cluster, Computational Grid, and Parallel Processing Units. To utilise the power of parallel architectures, the former sequential algorithm needs to be redesigned to be the distributed algorithm, and the design is very important to the processing speed.

Before designing a distributed algorithm, a communication paradigm must be chosen to exchange and update the information between tasks executing on separate computational nodes [8] such as the nodes in a PC-Cluster or the processors in the PPU. Here, according to the architecture of the testbeds involved, a message-passing paradigm is adopted, under which each participating processor has its own dedicated memory. The information is exchanged among the processors to inform about the changes, and many different software models can be used here to implement a distributed algorithm. The software model used here is the master-and-slave model [55] in which one workstation acts as the master while all others are slaves. The master divides the problem into tasks and assigns them to slaves, waits for them to complete their respective work, and finally, collects the results. To make full utilisation of the distributed system and, thus, have better efficiency, during design, the following points should be considered:

- (i) The problem should be partitioned in such a way that the amount of the inter-task

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

communication is minimised [8].

- (ii) The master and the slaves should work simultaneously, to effectively exploit the parallelism, thus wastage of resources is avoided as much as possible [55].

From the sequential version of the ISMC algorithm it can be seen that the most time-consuming part is the allocation of each pixel to the nearest cluster centre, because it requires the calculation of the distances of each pixel in the feature space to all possible cluster centres. Note that the computational requirements grow exponentially with the image size and number of contained clusters [4]. This problem is tackled here by dividing an input image into small equal-sized image blocks (IBs). Each slave's task is working only on the IB assigned to it, along with the necessary communication with the master, while the master's task is to calculate such statistical information as the trace of the between-cluster scatter matrix $tr(S_b)$, the trace of the within-cluster scatter matrix $tr(S_w)$, and the new cluster centres. and to make the decision of splitting or merging. An outline of the proposed D-ISMC algorithm is given in Algorithm 7. 1, the number of involved processes (N_T) and their ranks (task_id) are determined. The master's rank is set to zero, while slaves are assigned nonzero ranks in the range from 1 to $N_T - 1$. Every slave (or the master) gets its own role depending on the value of task_id, and then the corresponding function (master-task or slave-task) is called [36].

Algorithm 7 D-ISMC Algorithm

```

1: begin
2: Determine the number of tasks ( $N_T$ );
3: Determine the rank of the task (task_id);
4: if (task_id is 0) then
5: Master-task();
6: else
7: Slave-task();
8: end if;
9: end

```

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

The pseudo code of master-task is listed in Algorithm 8. The master enters into its initialising phase (Line 1 in Algorithm 8), at this phase the master mainly does the following things one by one:

- (i) Read the *image size* (I_{size});
- (ii) Calculate the size of task ($task_size$);

$$task_size = I_{size}/N_T \quad (\text{Eq. 3.1})$$

- (iii) Compute the initial cluster centre;

$$m_1 = \frac{\sum_{n=1}^N y_n}{N} \quad (\text{Eq. 3.2})$$

- (iv) Send the $task_sizes$ and initial cluster information to the corresponding task;

Then, the master-task enters the splitting-and-merging part (Line 3 to Line 12 in Algorithm 8). In this part, the master-task exchanges the necessary information with the slave-tasks and collects the results from the slaves to determine which cluster has to be split or which two clusters have to be merged. The master-task checks whether the convergence condition has been met, and if so the master sends the finalisation command to all the slave-tasks (Line 13 and 14 in Algorithm 8). The thresholds used here are the same as those described in Chapter 4.

The pseudo code of the slave-task is listed in Algorithm 9. At first, the specific slave-task blocks itself until it received the initialisation information from the master-task (Line 1 in Algorithm 9). After that, the slave-task enters its main task - allocating each pixel in the **IB** of its own to the nearest cluster centre (Line 2 to Line 8 in Algorithm 9). These steps are kept running until the slave process receives the finalisation command from the master-task. Then it begins to label its own IB (Line 9 in Algorithm 9).

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Algorithm 8 Master-Task of D-ISMC

```
1: begin
2: Initialise jobs and assign tasks to slave;
3: While (no convergence)
4: Send new cluster information to all slaves;
5: Split clusters;
6: Remove all clusters with zero elements;
7: Perform k-means;
8: Merge Cluster;
9: Remove all clusters with zero elements;
10: Perform k-means;
11: Receive results and compute new cluster centres;
12: End while
13: Send cluster information to all slaves;
14: Let every slave to label the IB of its own;
```

Algorithm 9 Slave-Task of D-ISMC

```
1: Get IB information from the “master” and read IB data;
2: While (no convergence);
3: Receive cluster centres from the master;
4: Set all elements to nearest cluster centre;
5: Compute the new cluster centres;
6: Return the newly computed cluster information;
7: Wait for master node;
8: End while;
9: Begin to label the IB;
```

3.4 Implementation on Parallel Architectures

3.4.1 Implementation on a PC Cluster

As described in Chapter 3, a PC-cluster is an ideal platform for the implementation of the D-ISMC algorithm, because of its architecture. The whole algorithm is implemented with MPICH-P4, and the intra-cluster network used here is Myrinet with a bandwidth of 3.3 Gbps.

An asynchronous parallel processing model is used here, whereby each slave sends the individual result back to the master as soon as the result is obtained. The master responses to the slave according to the results sent by the most recent two slaves without any blocking [30]. As shown in Figure 3.5, there are four slave-tasks and one master-task involved, where T_t stands for the time to transfer image data from the master process to the different slave processes, T_i stands for the time that the specific slave process needs to process its own IB and T_c stands for the time needed by the master process to process the two recent responses from the slave processed. In the first T_c in Figure 3.5 which is marked by “*”, the master process makes its decision solely according to the first result from the slave process. Firstly the four slave-tasks are assigned jobs one by one, then the slave-tasks begin to work on their own IBs according to Algorithm 1, as soon as every slave-task gets the job of its own it will start to work on it. When the convergence condition is met, the specific slave-task will return the result to the master-task, and the master-task will process it immediately according to the most recent two results it received (including this one). When the master-task completes its job, the result will be sent back to the specific slave-task. This process will run from time to time until the convergence condition is met.

Figure 3.5: Schedule of the D-ISM algorithm on a PC-Cluster

3.4.2 Implementation on the PPU of X-Sat

According to the architecture (shown in Figure 2.11) of the Parallel Processing Units (PPUs) of X-Sat, one of the four FPGAs acts as the master, and the processing units act as slaves. The flow chart of the implementation of the D-ISM algorithm on the PPU of X-Sat is shown in Figure 3.6. Here the main job of the FPGA is to compute the global minimums and maximums of the whole image, decide when to split and when to merge, while the jobs of the processing units which have much more programmable computing resource are mainly on processing the image blocks of their own. The whole parallel process is done in a synchronous manner.

The flow of the implementation on the PPU is summarised as the following:

1. Transfer image data from RAM-Disk to the local memory of the individual *Processing Units (PU)* via *FPGA*;
 - 1.1. Transfer the IBs to the corresponding *PPUs*;
 - 1.2. Calculate the *global mean* of the whole image (done by *FPGA*). By now all image data is stored in the local RAM of the *PPUs*, just the *global mean* is stored in *FPGA*;
2. *Splitting and Merging*;
 - 2.1. *Splitting*.

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Figure 3.6: Flow chart of the D-ISMC implementation on the PPU of X-Sat

- 2.1.1. Every *PU* computes the minimum and maximum of each band of each cluster, and sends the results to the *FPGA*. The reason why the computation is done here is to minimise the data exchanged between the *PU*s and the *FPGA*.
- 2.1.2. The *FPGA* receives these minimums and maximums of each cluster, then the real minimum and maximum of each cluster are computed. After this the master (the *FPGA*) sends the newly computed result back to every *PU*.
- 2.1.3. With the minimum and maximum of each band of one specific cluster, a hyper-cube is constructed and that encloses the whole cluster. Next to each corner the farthest point and nearest point in this cluster is found, and these points with the

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

corresponding distances are send to the *FPGA*.

2.1.4. The *FPGA* receives these nearest and farthest points and the corresponding distances, and then decides which is the nearest one or farthest one to every specific corner. Now for each cluster a set of points is obtained, the size of which is the double of the amount the corners. Then the largest distance between these points is gotten, if this distance is larger than a pre-defined threshold. Then this cluster is split into two, and the cluster centres of the new born cluster are these two points. This is done for every cluster, and the result is sent to each PU. If there is no cluster to split here then go to *Step 2.2*. The constraint can be considered here is that every cluster can only be split once now.

2.1.5. Every PU gets the result after *Step 2.1.4*, and then it assigns each pixel in its own *IB* to the nearest cluster centre, calculate cluster means again. Empty clusters are removed. Go to *Step 2.1.1*.

2.2. *K-means* step.

2.2.1. The *FPGA* sends the cluster centres to each *PU*.

2.2.2. Every *PU* gets the cluster centres from the *FPGA*, and assigns each pixel in this *IB* to the nearest cluster centre, and calculates new cluster centres. Then the result is sent to the *FPGA*.

2.2.3. The *FPGA* gets the newly computed cluster means, and calculates the global means with them.

2.2.4. Check whether the maximum of iteration times is met, if met go to *Step 2.3*, else go to *Step 2.2.1*.

2.3. *Merging*.

- 2.3.1. On each *PU*, the minimum distance between cluster centres is calculated if this distance is less than the pre-defined threshold, then these two clusters will be merged into one. The new cluster centres of this IB will be calculated, and they will be sent to the *FPGA*. If no clusters are merged in this step go to *Step 2.4*.
- 2.3.2. The *FPGA* receives the result from *PU*s, and then the global cluster means are calculated. The new computed cluster means are sent to each *PU* after that. Go to *Step 2.3.1*.
- 2.4. *K-means* step (same as *Step 2.2*).
- 2.5. Check whether the convergence is met, if met go to *Step 2.6*, else go to *Step 2.1.1*.
- 2.6. Output the map and the classification result.

3.5 Evaluation of the Classification Result

Figure 3.7 compares the results of the ISMC, ISMC with Algorithm 6, and EISMIC procedures for the same satellite image shown in Appendix B.1, and they all use the same parameters as [45]. In Figure 3.7, column (a) shows the raw satellite images blocks, column (b) shows the results of the ISMC procedure and four clusters are found, column (c) shows the results of the app-ISMC procedure and four clusters are found, and column (d) shows the results of the EISMIC procedure and also four clusters are found. From the results, although all the three algorithm find the same number of cluster, the classification quality is different. For example, on considering the lowest image block, it is obvious that the ISMC procedure misclassified a region, while the app-ISMC and EISMIC procedures classified it correctly and the EISMIC procedure kept much more details than any other procedure.

To compare the performance of different algorithms, two quantitative measurements are used here. That is the *between cluster scatter matrix* S_b and the *within cluster scatter*

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Figure 3.7: Comparison of classification results : (a) raw satellite data, (b) ISMC classification, (c) app-ISMC classification, (d) EISMIC classification

matrix S_w [18]. As a good clustering algorithm, S_w should be as small as possible, and S_b should be as large as possible. As described in Chapter 2, the trace of S_w and S_b is used to compare the cluster distributions produced by the different algorithms numerically. Plots

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

of the $Tr(S_w)$ Vs different image scenes in Appendix B are shown in Figure 3.8, while that of the $Tr(S_b)$ Vs different image scenes in Appendix B are shown in Figure 3.9. For almost all images used in the experiment, the $Tr(S_w)$ by the app-ISM algorithm with the new approximation algorithm is smaller than that produced by the ISMC algorithm but larger than that produced by the EISM algorithm. Moreover, the $Tr(S_b)$ by the app-ISM algorithm with the new approximation algorithm is larger than that produced by the ISMC algorithm but smaller than that produced by the EISM algorithm. So from this point the performance of the EISM algorithm is better than the other two algorithms.

Figure 3.8: Trace of S_w versus image identifier.

The performance evaluation was done on a DELL Linux workstation equipped with two *Intel^R XeonTM* 2.60GHz CPUs and 1 GBytes of RAM. Execution times for the ISMC, app-ISM and EISM for the images used in this study are plotted versus image

Figure 3.9: Trace of S_b versus image identifier.

identifier in Figure 3.10. Run times for app-ISM and EISM are significantly more consistent than those for ISM procedure, this is because the added k-Means steps consume too much computing power. As can be seen here, although the Enhanced ISM algorithm boosts the classification quality, it takes much more time to run.

Figure 3.11 shows the the processing time versus different image sizes for EISM algorithm. All the images were made from a 3000×3000 SPOT satellite image, and the resizing algorithm uses nearest neighbour interpolation to determine the values of pixels in the output image [25]. Here the x – axis shows the length of different image squares, while the y – axis shows the processing time in second. From the figure, it can be concluded that the processing time of the algorithm increases as the size of images. The reason here is that EIMSC algorithm uses a very efficient method find the cluster centres, and the speed of splitting and merging is also fast, while the added k-Means steps need to review every pixel during processing, and its computational complexity is

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

Figure 3.10: Processing time versus image identifier.

almost $O(n)$.

Figure 3.11: Processing time versus different image sizes for the EISMIC algorithm.

In order to scale how good a parallel algorithm is, the speedup is used and it is defined

CHAPTER 3. ENHANCED ISMC IMAGE CLASSIFICATION ALGORITHM

as:

$$speedup = \frac{executionTime(nProc = 1)}{executionTime(nProc = n)} \quad (\text{Eq. 3.3})$$

where $nProc$ is the number of processing node in the experiment, $executionTime$ is the processing time used in the operation.

The speedup with respect to the execution time of the sequential implementation is shown in Figure 3.12. It can be seen that the speedups gained from the D-ISMC algorithm are almost equal to the ideal case which is indicated as the ideal speedup, because only the cluster information needs to be updated between the separate processing nodes in every step, which is negligible in comparison with the processing time assumed on the image blocks. The satellite image used in this experiment is a 3000×3000 3-band SPOT satellite image. When nine processing nodes are used, the speedup is about eight.

Figure 3.12: Speed up graph for D-ISMC algorithm on a PC Cluster

3.6 Summary

From the results of the different algorithms, it can be concluded that the performance of the EISM algorithm is the best among the three algorithms, but it is also the slowest one; The original ISMC algorithm is the fastest one, while its performance is not so satisfying. So an approximation method to find the largest pair-wise distance is introduced here, it boosts the processing speed and also makes out an accurate enough result. To make the EISM algorithm to be a fast and accurate image classification procedure, and parallel architecture is considered. With the use of D-ISMC algorithm, speedup eight is gained when nine processing nodes are involved and the classification result is the same as that of the EISM procedure.

Chapter 4

Mean-Shift Based Image Classification Algorithm

This chapter presents a practical algorithm for unsupervised nonparametric classification of large satellite imageries. The method is based on the mean shift procedure which is mentioned in Chapter 2.2.6, and to improve the classification accuracy a special filter - the majority voting filter [34] is used here, which make full use of spatial aspects to get the representative cluster in the neighbourhood. After an introduction to the mean shift clustering algorithm, an unsupervised satellite image classification system is described, followed by its performance evaluation. Finally, the parallel implementation of the unsupervised satellite image classification algorithm is presented.

4.1 Overlapping Issue in Classification Algorithms

Feature space analysis is a fundamental problem for many low/high level vision tasks, i.e. image classification [15], moving object tracking [16], image retrieval [15] etc. For image classification, the pixels of the image are mapped into the feature space. Classification problem then becomes *the task of finding modes in the mapped feature space* [16]. For moving object tracking, the tracking target object is represented by a specific distribution in the feature space, i.e. colour, texture. The tracking problem then turns out to be the

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

task of searching for locations whose distribution in feature space is closest to the tracking target object distribution in the following frames [16]. For image retrieval, each image is represented as a point in the feature space, i.e. colour, texture etc. The retrieval problem is then entitled as a task of locating the images whose feature space representation (a point) is closed to the example query images [15].

With the increasing availability of computing power, it is possible to handle the feature space analysis problem which is of high dimension and contains millions of data points. Many algorithms have been proposed for feature space analysis problem for various vision tasks in recent years [14]. Here, the focus is put on the problem of image classification through feature space analysis. As described above, the goal here is to find the modes of the data points in the feature space. This can be achieved through a clustering procedure. In the previous chapter, the EISMC clustering algorithm for image classification is described, which is a variation of the splitting and merging algorithm with some k-means steps added in. In that method, the feature space is represented as a mixture of Gaussian distributions. The algorithm needs to find the number of mixing components, k , and also the modes of the data set. Although, the EISMC algorithm and its variant are very useful, they have two shortcomings. Firstly, the thresholds for splitting and merging need to be defined in advance. If the thresholds are not proper, the clustering result may be erroneous due to the invalid assumption. Secondly, the accuracy of the algorithm depends on the percentage of each class in the scene to be separated [45].

Due to the shortcomings of the EISMC algorithm, another clustering algorithm proposed recently is also concerned in this study: mean shift algorithm. It is a robust clustering technique which neither requires priori knowledge of the number of clusters nor constrains the shape of the clusters. This is also an iterative algorithm like k-means or EISMC algorithm, but instead of estimating the mean of each cluster, it estimates the modes of the underlying multivariate distributions in the feature space. The number

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

of clusters is automatically obtained by finding the centre of the densest region in the feature space. In the following section, the mean shift theory, its property and implement details are described.

In unsupervised classification algorithms, one important issue is *how to handle the overlapping problem in feature space*. For example if two elliptic Gaussian clouds embed each other deeply or lightly (as shown in Figure 4.1), how to classify them correctly? To make this issue intuitive, a data set as large as $256 \times 256 \times 2$ is created with the Gaussian Mixture Generator. This data set is displayed in a 2-D space in Figure 4.1, and because of the two clouds are overlapped into each other, here the two cluster clouds are labelled in different colours and shown in two sub-images. And to display it as a colour image,

Figure 4.1: Data set in 2D space

all the values in the third band are set to 255, so the data set can be displayed as colour image in Figure 4.2.

Figure 4.3.a shows the ground truth of the artificial data set, which means the left part belongs to one cluster while the right part belongs to another cluster. Figure 4.3.b shows the map of the points in the overlapping area which means the points in the overlapping part (shown in Figure 4.1) is denoted as a different colour in both of the two parts. Thus

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.2: 256x256 image created with Gaussian Mixture Generator

here an interesting question is raised: *Is the classification algorithm able to handle this overlapping issue to classify the data set correctly?*

Figure 4.3: class description: (a) ground truth map, (b) and overlapping map.

To measure how much two clusters are overlapping each other, a concept of *overlap-*

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

ping degree needs to be defined. A single 1-D Gaussian distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \quad (\text{Eq. 4.1})$$

where μ is the mean of the distribution and σ is the standard deviation. From the property of Gaussian Distribution, 68.26% points fall in the 1σ area, 95.44% in the 2σ area, 99.72% in the 3σ area. For the problem, the number of points within 2σ is considered as accurate enough. The concept of overlapping is defined as follows:

- point which falls into the within- 2σ area of the two clusters.
- point which falls into the within- 2σ area of its own cluster, and falls in the within- 2σ area of the other cluster at the same time.

The points that fulfil the above conditions are called overlapping points. And *the overlapping rate* is defined as,

$$R_o = \frac{N_o}{N_a} \times 100\% \quad (\text{Eq. 4.2})$$

Where N_o is the number of the points in the overlapping parts, and N_a is the total number of points in the data set.

Next the Mean-Shift algorithm with the radius of search window $h = 40$ (in Algorithm 4) is applied on this data set, and the algorithm finds two clusters (the number is correct). Figure 4.4 is the classification map of the result. Figure 4.5 shows the result in feature space. Here the classification accuracy is 96.28%, and 2437 points are misclassified.

From the result it can be concluded that for the Mean Shift algorithm (Figure 4.5), overlapping is not allowed and the data set is just separated roughly with a plane. To handle the overlapping area of clusters in the feature space and to improve the classification accuracy, the spatial information needs to be used. The position information cannot be used directly because the position can be regarded as a uniform distribution while

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.4: Classification map of the standard Mean Shift algorithm

Figure 4.5: 2D feature space of the classification result the Mean Shift algorithm.

the data set should obey some Gaussian distributions, and it is destroyed if the uniform distribution is used directly.

The basic idea of the Majority Voting Filter [34] is to use the neighbourhood information to handle the overlapping issue, and it is applied onto the classification map. In

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

the classification map, every pixel is replaced as the cluster index of its own, and it is meaningless to sort the index of the cluster. The Majority Voting Filter handles these two situations:

- (i) find the dominant part in the window replace the centre item of the window with it.
- (ii) if the parts in the window can match each other just do nothing to the window.

Figure 4.6 shows the two scenarios that can be handled by the Majority Voting Filter. For scenario $\langle a \rangle$, it can be seen that the dominate part in this window is “b” so the centre item is replaced with “b” when the filter is applied. For scenario $\langle b \rangle$, all the parts in the window match each other, that means their amounts are the same, so the filter does nothing on it and leaves things as they are. The window size here is defined as: firstly, a minimum region size r needs to be predefined, which means if a region is smaller than this then it is deleted and merged into its neighbour. Secondly, the window size W is defined as

$$W = \lceil 2 \times r + 1 \rceil \quad (\text{Eq. 4.3})$$

and the dominate filter runs in this way described in Algorithm 10.

Algorithm 10 Dominate Filter Algorithm

```

1: begin initialise
2: Get input-map and the minimum-good-region-size.
3: Calculate size of the window according to the minimum-good-region-size.
4: Begin to slide the window.
5: if the parts in the window can match each other.
6: then do noting
7: else
8: find the dominant part in the window and the assign it to the map item.
9: end
10: end
```

b	b	b	b	b	a	b	c	d	e
a	b	c	b	e	a	b	c	d	e
b	b	c	d	b	a	b	c	d	e
a	b	c	b	b	a	b	c	d	e
b	b	b	d	e	a	b	c	d	e
<a>									

Figure 4.6: Two scenarios handled by the Dominate Filter

Figure 4.7 shows the result map after the dominate filter is applied, and here only 24 pixels are misclassified. Figure 4.8 shows the classification result in 2D space (it is displayed in two sub-images to show the overlapping), as expected, the misclassification of the points in the overlap issue is greatly alleviated when the Dominate Filter is used.

4.2 Implementation of Mean Shift Algorithm on Parallel Architectures

In this section, the rough design of the distributed Mean Shift classification algorithm is given out in Subsection 4.2.1. Then the implementation on PC-Cluster is described in Subsection 4.2.2.

4.2.1 Design of the Distributed Mean Shift Classification Algorithm

From the Mean Shift clustering algorithm described in Algorithm 5, it can be concluded that an intensive part of computing power is consumed in the postprocessing step (to classify each pixel to the nearest cluster centre). To improve the processing speed a par-

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.7: Classification map of the modified Mean Shift algorithm.

Figure 4.8: Classification result of the modified Mean Shift algorithm in 2D space

allel paradigm is used to distribute the computing job onto a parallel architecture. Like the design of the D-ISMC algorithm, the distributed Mean Shift classification algorithm also employs the Master-and-Slave parallel programming model [55]. From the sequential

version of the Mean Shift classification algorithm, the most time-consuming part is the allocation of each pixel to the nearest cluster centre, because it requires the calculation of the distances.

Figure 4.9 displays the major parallelization steps of the distributed Mean Shift classification algorithm. In order to distribute the computational workload, the raw satellite needs to be partitioned in to some smaller strips according to the number of processing nodes. Then each process on separate processing node is then able to perform computations concurrently using the local data of their own, at the mean time, the processes can communicate with each other using the inter-process communication mechanism (i.e., MPI, PVM). During the computational procedure, there are situations in which information pertaining to the distributed data is needed. The whole Mean Shift process is finalised when the pre-defined convergence condition is met. Finally, the local data is gathered and reduced for global use, and then the Dominate Filter is applied onto the result as a post-processing step.

4.2.2 Implementation on a PC Cluster

The pseudo code of the main routine of the distributed Mean Shift algorithm is shown in Algorithm 11. When a D-MeanShift process is created, the process firstly finds its own identity according to `task_id`, if it is the master then the Master-task is executed, else the Slave-task is executed.

Algorithm 12 shows the routine of the Master-task. The Master-task first do the initialising processing on the whole image, and assign tasks to the slaves, and at the same time it keeps a copy of the whole image. Then it applies Mean Shift algorithm onto the image, and get the class centres. Next these class centres are sent to the slaves, and the master enters some iterative steps, gathering class information from slaves, computing new class information, until the convergence condition is met. When the

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.9: Parallelisation steps of D-MeanShift

global classification map is sent to the master, the master applies the Dominate Filter as a post-processing step, at last the classification result is saved.

Algorithm 13 shows the routine of the Slave-task. Every slave reads the image block of its own first, and then begins to wait for the class information from the master. When

Algorithm 11 D-MeanShift Algorithm

- 1: **begin**
 - 2: Determine the number of tasks (N_T);
 - 3: Determine the rank of the task (task_id);
 - 4: **if** (task_id is 0) **then**
 - 5: Master-task();
 - 6: **else**
 - 7: Slave-task();
 - 8: **end if**;
 - 9: **end**
-

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Algorithm 12 Master-Task of D-MeanShift

```

1: begin
2: Initialise jobs and assign tasks to slaves;
3: do the Mean Shift processing on the whole image (no convergence);
4: While (no convergence)
5: Send result cluster information to all slaves;
6: Gather local cluster information from all slaves;
7: Compute the global cluster information;
8: End While
9: Gather the classification result maps from each processing node;
10: Perform the Dominate Filter onto the global result map;
11: Save Result;
12: end

```

the class information arrives, the slave also enters an iteration, classifying all the elements to the nearest class centres, computing the new class information, sending it back to the master. When the convergence condition is met by the master the iteration is finalised, and the slave sends the local classification result map back to the master.

Algorithm 13 Slave-Task of D-MeanShift

```

1: Get IB information from the “master” and read IB data;
2: While (no convergence);
3: Receive cluster centres from the master;
4: Set all elements to nearest cluster centres;
5: Compute the new cluster centres;
6: Return the newly computed cluster information;
7: Wait for master node;
8: End while;
9: Send the local classification result map back to the Master;

```

4.3 Evaluation of the Classification Result

The first experiment was carried out for nine data sets created with the Gaussian Mixture Generator (Appendix C), and each of them contains two classes. Using the Gaussian Mixture Generator the corresponding ground truth was gained when the data set was created, so the overlapping rate is also known at the same time. Figure 4.10 shows

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

the result of the first experiment, the x – axis shows the overlapping rate, while the y – axis shows the classification accuracy rate. The green curve in the figure show the classification accuracy of the Mean Shift algorithm, and it can be seen that when the overlapping rate is smaller than 60%, the accuracy can be remains as acceptable as larger than 90%. While when the Dominate Filter was used as the postprocessing step the classification accuracy (shown as the red curve) can also remains larger than 90% even when the overlapping rate is as large as 80%, which means the two clusters are overlapped heavily in the feature space.

Figure 4.10: Plot of Classification Accuracy

Figure 4.11 compares the results of the EISM procedure and the Mean Shift procedure with Dominate Filter whose window size is 3×3 . Both algorithms perform on the

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

same satellite image shown in Appendix B.1. The reason for the window size of 3×3 is to minimise the amount of blurring at the image's borders. In Figure 4.11, column (a) shows the raw satellite images blocks, column (b) shows the results of the EISMC procedure and four clusters are found, and column (c) shows the results of the Mean Shift procedure and six clusters are found. From the results, both the number of clusters and the classification quality are different. The EISMC procedure keeps much more detail than the Mean Shift procedure, while the Mean Shift procedure ignores some edge information because the Dominate Filter is used.

Figure 4.12 and Figure 4.13 show the plots of the trace of within cluster scatter matrix S_w and between cluster scatter matrix S_b versus image identifiers which are listed in Appendix B. As described in Chapter 2, for a good classification algorithm the $tr(S_w)$ value should be as large as possible, and the $tr(S_b)$ value should be as small as possible. From the figures, for almost all images used in the experiment, the $Tr(S_b)$ by the Mean Shift algorithm with the new approximation algorithm is larger than that produced by the ISMC algorithm and even a little larger than that produced by the EISMC algorithm. Moreover, the $Tr(S_w)$ by the Mean Shift algorithm with the new approximation algorithm is much smaller than that produced by the ISMC algorithm and even a little smaller than or approach that produced by the EISMC algorithm. So from this point the performance of the Mean Shift algorithm is no worse than the EISMC algorithm.

Figure 4.14 shows the processing time versus different image sizes. All the images were made from the same 3000×3000 SPOT satellite image as the one mentioned in the previous chapter, and the resizing algorithm is also the same one. Here $x - axis$ shows the length of different image squares, while $y - axis$ shows the processing time in second. As can be seen from the figure, the processing time curve does not rise so steep as that shown in Figure 3.11. This is because in the mean shift procedure, the time used to find modes in different image size are almost the same, because the algorithm makes a

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.11: Comparison of classification results: (a) raw image data, (b) EISMIC classification, (c) Mean Shift classification

histogram for every image. While once the histogram is done, the size of computing are all the same which is $256 \times 256 \times 256$ large, so the time consumed in this part is almost the same. When all modes are found, the following step is to assign every pixel to its

Figure 4.12: Trace of S_w versus image identifier

corresponding cluster, and this processing time is concern with the image size. So as the image size increases, the processing time increases at the same time but the degree is not so large as that of the EISMC algorithm.

Figure 4.15 shows the speedup graph of the distributed Mean Shift algorithm on the PC-Cluster. This experiment was carried on a 3-band satellite image with the size of 3000×3000 pixels, and from the graph a speedup as large as seven was gained when nine processors were used.

4.4 Summary

In this chapter, the basic Mean Shift theory is reviewed, and a Mean Shift satellite image classification algorithm is also given out. To solve the overlapping issue in spectral feature space and to improve the classification accuracy, a concept of overlapping rate is defined and a dominate filter is proposed to make use of the neighbourhood spatial information.

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.13: Trace of S_b versus image identifier

Figure 4.14: Processing time for different image sizes.

At last a distributed Mean Shift classification algorithm is described, and a good speedup graph is gained as a result.

CHAPTER 4. MEAN-SHIFT BASED IMAGE CLASSIFICATION ALGORITHM

Figure 4.15: D-MeanShift speedup graph

Chapter 5

Recommendations for Future Developments

By the time when this thesis was completed, the PPU for X-Sat was still under development, so the implementations of the algorithms mentioned above could not be done completely. According to the architecture and characteristics of the PPU some recommendations regarding corresponding implementations in the future are listed below.

5.1 Floating Point Computing

In the design of the PPU, every Processing Node (PN) is implemented as a StrongARM SA1110 processor which has no Floating Point Unit (FPU). Hence, the current implementations of the mentioned algorithms on the PC cluster have to be revised to satisfy the situation. Most of the utilised floating point computing can be replaced with integer computing through a scaling approach. For example, if the desired accuracy is set to be $1/1000$, all the values involved in computing have to be up scaled by at least 1000 depending on the actual number of involved operations and the expected error propagation. After all the computation, the result can be down scaled with a corresponding value. With this method some accuracy may be lost and overflows may occur, so a trade off needs to be found for a satisfying result.

5.2 Histogram Building and Information Exchange

In the implementation of the Mean Shift algorithm on the PPU, a very time consuming part is the construction of the global histogram. To reduce the computing burden, some software-hardware partitioning [53] techniques can be used. At the beginning when the whole image is loaded into the separate PNs, the histogram can be built at the same time by the FPGAs. Finally, after the entire image is transferred the histogram is sent to each PN. The advantage is a significantly reduced number of memory access. Moreover, as on the PN's side, after each iteration of the algorithm the tasks that are executed by the master node can be performed on the FPFA. Thus, an additional PN would become available for computations.

5.3 Accelerating Cleaning-up Stage

During the postprocessing a dominate filter with a window size of 3×3 pixels is applied to the classification result. To speedup the whole processing speed a smaller window size can also be considered, for example a 1×3 window. The immediate benefit is that this operation can be performed in real-time on the FPGA, while the result is streamed back to the RAM-Disk [50] [56]. However, the disadvantage is a reduced accuracy, so a trade-off between processing speed and classification accuracy needs to be considered here.

Chapter 6

Conclusions

In this study, the main emphasis was put on the extension of two unsupervised image classification algorithms for satellite images based on machine learning theory, together with their corresponding implementations on parallel architectures, i.e. a PC cluster and outlines for the Parallel Processing Unit of the X-Sat mission.

The first algorithm is the Enhanced-ISM algorithm. To improve the classification accuracy of the ISM algorithm, two k-means steps were added after the splitting step and the merging step, respectively. The classification results are compared with those of the k-Means algorithm as well as the ISM algorithm. The findings were that the classification accuracy and the processing speed were both increased.

The second algorithm is the Mean Shift algorithm. To handle the overlapping issue in the feature space, the spatial information of every pixel is taken into consideration by the so-called dominate filter. With the dominate filter, the overlapping of cluster clouds in feature space is allowed and higher classification accuracy can be gained.

The two above algorithms were both implemented on a particular PC cluster and satisfying speedups were gained. Because the parallel processing unit was still incomplete, so only a rough design could be given without an actual implementation and experimental results.

As part of this study, two papers were published:

CHAPTER 6. CONCLUSIONS

- Liu Bo, Timo Bretschneider, *D-ISMC: A distributed unsupervised classification algorithm for optical satellite imagery*, Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Vol. 6, pp. 3413-3415, 2003
- Liu Bo, Timo Bretschneider, *A distributed non-parametric unsupervised classification algorithm for remotely sensed optical imagery*, Proceedings of the Asian Conference on Remote Sensing, Vol. 1, pp. 426-431, 2004

References

- [1] Globus Website. <http://www.globus.org>.
- [2] MPI toolbox for matlab - MPITB. <http://atc.ugr.es/javier-bin/mpitbeng>.
- [3] MPICH-G2 project in high-performance computing laboratory of North Illinois University. <http://www3.niu.edu/mpi>.
- [4] MPICH project in mathematics and computer science division of Argonne National Laboratory. <http://www-unix.mcs.anl.gov/mpi/>.
- [5] Parallel programming with MatlabMPI. <http://www.ll.mit.edu/MatlabMPI/>.
- [6] PVM toolbox for matlab - PVMTB. <http://atc.ugr.es/javier-bin/pvmtbeng>.
- [7] M.B. Al-Daoud, N.B. Venkateswarlu, and S.A. Roberts. Fast k-means clustering algorithms. <http://citeseer.nj.nec.com/al-daoud95fast.html>.
- [8] G. R. Andrews. *Foundations of multithreaded, parallel, and distributed programming*. Persion Education, 1st edition, 2000.
- [9] Argonne National Laboratory. The message passing interface (MPI) Standard. <http://www.mcs.anl.gov/mpi>.
- [10] G. Ball and D. Hall. A cluster technique for summarizing multivariate data. *Behav. Sci*, 12:153–155, 1978.

REFERENCES

BIBLIOGRAPHY

-
- [11] T. Bretschneider. Singapore's satellite mission x-sat. *Proceeding of the IAA Symposium on Small Satellites for Earth Observation*, pages 105–108, 2003.
 - [12] T. Bretschneider, B. Ramesh, V. Gupta, and I. McLoughlin. Low-cost space-borne processing on a reconfigurable parallel architecture. *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms*, pages 93–99, 2004.
 - [13] C. Chen and B. Schmidt. Computing large-scale alignment on a multi-cluster. *IEEE Cluster Computing, Hong Kong, IEEE, 2003*, page 38.
 - [14] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, page 750, 1997.
 - [15] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell.*, 24:603–619, 2002.
 - [16] D. I. Comaniciu. Nonparametric robust methods for computer vision, Jan, 2000. <http://www.caip.rutgers.edu/riul/research/theses/comaniciuthesis.html>.
 - [17] J. Donnay, M. J. Barnsley, and P. A. Longley, editors. *Remote Sensing and Urban Analysis*. St. Edmundsbury, 2001.
 - [18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. A Wiley-Interscience Publication, 2nd edition, 2001.
 - [19] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke. Wide-area implementation of the message passing interface. *Parallel Computing*, 24(12):1735–1749, 1998.

REFERENCES

BIBLIOGRAPHY

-
- [20] I. Foster and C. Kesselman. The globus projects: A statue report. *Proceeding of the Heterogeneous Computing Workshop*, 1998.
 - [21] I. Foster and C. Kesselman. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers, Inc, 1998.
 - [22] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational Grids. pages 83–92, 1998.
 - [23] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: enabling scalable virtual organizations. *International Journal Supercomputer Applications*, 15(3), 2001.
 - [24] J. Gauvain and C. Lee. Maximum a-posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions SAP*, 2:291–298.
 - [25] P. J. Gibson and C. H. Power. *Introduction Remote Sensing Digital Image Processing and Applications*. Routledge, 2000.
 - [26] Globus project. <http://www.globus.org>.
 - [27] B. Gorham. Mapping agricultural landuse in the mississippi alluvial valley of arkansas. http://www.cast.uark.edu/local/lulc/lulc_home.html.
 - [28] GridFTP. <http://www.globus.org/datagrid/gridftp.html>.
 - [29] D.A. Grove, P.D. Coddington, K.A. Hawick, and F.A. Vaughan. Cluster computing with imacs and power macintoshes. *Proceeding of the conference on Parallel and Distributed Computing Systems*.
 - [30] K. Hwang and Z. Xu. *Scalable Parallel Computing - Technology, Architecture, Programming*. The McGraw-Hill Companies, 1st edition, 1998.

REFERENCES

BIBLIOGRAPHY

-
- [31] X. Jia and J. A. Richards. Efficient transmission and classification of hyperspectral image data. *IEEE Transactions on Geoscience and Remote Sensing*, 41(2):1129 – 1131, May 2003.
 - [32] N. T. Karonis. Mpich-g2: A grid-enabled implementation of the message passing interface. <http://citeseer.nj.nec.com/554400.html>.
 - [33] G. Karypis, E. (Sam) Han, and V. Kumar NEWS. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68 – 75, 1999.
 - [34] L. Lam and C. Y. Suen. Application of majority voting to pattern recognition: An analysis of its behaviour and performance. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 27(3):553 – 568, 1997.
 - [35] T. M. Lillesand and R. W. Kiefer. *Remote Sensing and Image Interpretation*. John Wiley and Sons, Inc, 3rd edition, 1994.
 - [36] B. Liu and T. Bretschneider. D-ISMC: A distributed unsupervised classification algorithm for optical satellite imagery. *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, pages 3413–3415, 2003.
 - [37] U. Maulik and S. Bandyopadhyay. Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE Transactions on Geoscience and Remote Sensing*, 41(5):1075 – 1081, May 2003.
 - [38] MPI standard 2.0. <http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html>.
 - [39] Oak Ridge National Laboratory. PVM: Parallel Virtual Machine. <http://www.epm.ornl.gov/pvm/pvmhome.html>.

REFERENCES

BIBLIOGRAPHY

-
- [40] Center of Excellence in Space Data and Goddard Space Flight Center Information Sciences (CESDIS). Beowulf project at cesdis. <http://www.beowulf.org>.
 - [41] Ed R. Buyya. *High Performance Cluster Computing: Architectures and Systems, Vol 1*. Prentice Hall PTR, 1999.
 - [42] J. A. Rice, editor. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.
 - [43] J. A. Richards. *Remote Sensing Digital Image Analysis : An Introduction*. Springer-Verlag, 2nd edition, 1993.
 - [44] K. Saitwal, M. R. Azimi-Sadjadi, and D. Reinke. A multichannel temporally adaptive system for continuous cloud classification from satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 41(5):1098 – 1104, May 2003.
 - [45] J. J. Simpson, T. J. McIntire, and M. Sienko. An improved hybrid clustering algorithm for natural scenes. *IEEE Transactions on Geoscience and Remote Sensing*, 38(2):1016 – 1032, March 2000.
 - [46] T. Sterling and D. J. Becker. Beowulf: A parallel workstation for scientific computation. *Proceeding of International Conference on Parallel Processing*, 1995.
 - [47] W. R. Stevens. *Advanced programming in the UNIX environment*. Pearson Education, 1993.
 - [48] W. Sun, V. Heidt, P. Gong, and G. Xu. Information fusion for rural land-use classification with high-resolution satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4):883 – 890, April 2003.
 - [49] SUN Grid Engine. <http://www.sun.com/software/gridware/sge.html>.

REFERENCES

BIBLIOGRAPHY

-
- [50] R. Susilo and T. Bretschneider. On the realtime satellite image compression of x-sat. *Proceedings of the International Conference on Information, Communications and Signal Processing and IEEE Pacific-Rim Conference on Multimedia*, 1:474–478, 2003.
 - [51] The Internet Engineering Task Force. Network file system (NFS) version 4 protocol (RFC 3530). <http://www.ietf.org/rfc/rfc3530.txt>.
 - [52] I. L. Thomas, V. M. Benning, and N. P. Ching. *Classification of Remotely Sensed Images*. Adam Hilger, Bristol, 1987.
 - [53] F. Vahid. modifying min-cut for hardware and software functional partitioning. *Codes/CASHE'97*, pages 43–48, 1997.
 - [54] S. Verbeek. Gaussian mixtures generator. <http://carol.wins.uva.nl/~jverbeek/gm>.
 - [55] B. Wilkinson and M. Allen. *Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers*. Pearson Education, 1st edition, 1999.
 - [56] D. Xue, T.C. Low, T. Bretschneider, and I. McLoughlin. Remote real-time oil spill detection in space (under review).
 - [57] Y. Zhang and B. Guindon. Quantitative assessment of a haze suppression methodology for satellite imagery: Effect on land cover classification performance. *IEEE Transactions on Geoscience and Remote Sensing*, 41(5):1082 – 1089, May 2003.

Appendix A

6-band Landsat Test Image

Figure A.1: 1024x768 6-band LANDSAT image

Appendix B

Other Test Images

Figure B.1: 517x472 3-band IKONOS satellite image

Figure B.2: 934x500 3-band LANDSAT satellite image

Figure B.3: 512x512 3-band LANDSAT satellite image

Figure B.4: 704x503 3-band MOMS satellite image

Figure B.5: 271x329 3-band SPOT satellite image

Figure B.6: 429x436 3-band SPOT satellite image

Figure B.7: 801x867 3-band SPOT satellite image

Appendix C

matlab source code for function mixgen

```
function [X, T, pt, map, Lik] = mixgen(n, m, k, d, c, e)
%mixgen - Gaussian mixture generator
%
%[X, T] = mixgen(n, m, k, d, c, e)
% n - size of training set
% m - size of test set
% k - number of components
% d - dimension
% c - separation degree
% e - maximum eccentricity
%returns
% X - training set (n x d)
% T - test set (m x d)

% Nikos Vlassis, 2000
% for definitions see (Dasgupta, 1999)
```

```

R = zeros(k, d^2);

% mixing weights
while 1
    W = randn(k,1);
    W = W / sum(W);
    if all(W > 1/(4*k))
        break;
    end
end

% create c-separated Gaussian clusters of maximum eccentricity e
trials = 1; while 1
    X = [];
    T = [];
    M = randn(k,d)*sqrt(k)*sqrt(c)*trials/10;
    Trace = zeros(k, 1);
    for j = 1: k
        % to make a random rotation on the speific cluster
        U = rand(d,d)-0.5;
        U = sqrtm(inv(U*U')) * U;
        L = diag(rand(d,1)*(e-1)+1).^2/100;
        msg = 1;
        while msg
            [C, msg] = chol(U*L*U');
        end
    end
end

```

```

R(j, :)=C(:)';
%
nj = ceil(n*W(j));
Xj = randn(nj, d) * C;
X = [X; repmat(M(j, :), nj, 1) + Xj];
Trace(j) = trace(cov(Xj));
mj_(j) = nj;
%
mj = ceil(m*W(j));
Tj = randn(mj, d) * C;
%
cc = 1;
for k_=1:d
    cc = cc*L(k_,k_); %
end
mm = 2;
pt(j) = (2*pi)^(-d/2) / cc^(1/2) * exp(-0.5*2*mm^2);
%
T = [T; repmat(M(j, :),mj ,1) + Tj];
%
% mj_(j) = mj;
%
end
% check degree of separation
error = 0;
for i = 1: k-1

```

```
        for j = i+1: k
            if norm(M(i, :)-M(j, :)) < c * sqrt(max(Trace(i), Trace(j)))
                error = 1;
            end
        end
    end
end
if ~error
    break;
end
trials = trials + 1;
end

base = 1; for i = 1:k
    map(base:sum(mj_(1: i))) = i;
    base = base + mj_(i);
end

Lik = em_gauss(X, M, R);
```

Appendix D

matlab source code for function emgauss

```
function L = em_gauss(X, M, R)
%em_gauss - compute likelihoods for all points and all components
%
%L = em_gauss(X,M,R)
% X - (n x d) matrix of input data
% M - (k x d) matrix of components means
% R - (k x d^2) matrix of Cholesky submatrices of components covariances
%      in vector reshaped format. To get the covariance of component k:
%      Rk = reshape(R(k,:),d,d); S = Rk'*Rk;
%returns
% L - (n x k) likelihoods of points x_n belonging to component k

% Nikos Vlassis, 2000

[n,d] = size(X); k = size(M, 1);

L = zeros(n, k); for j = 1:k
```

```
% Cholesky triangular matrix of component's covariance matrix
Rj = reshape(R(j, :), d, d);

% We need to compute the Mahalanobis distances between all inputs
% and the mean of component j; using the Cholesky form of covariances
% this becomes the Euclidean norm of some new vectors
New = (X - repmat(M(j, :), n, 1)) * inv(Rj);
Mah = sum(New.^2, 2);

L(:,j) = (2*pi)^(-d/2) / det(Rj) * exp(-0.5*Mah);
end
%
```