Detection And Classification Of Buried Radioactive Materials

Wei Wei

DETECTION AND CLASSIFICATION OF BURIED RADIOACTIVE MATERIALS

By

Wei Wei

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

December 2011

DETECTION AND CLASSIFICATION OF BURIED RADIOACTIVE MATERIALS

By

Wei Wei

Approved:

Name: Wei Wei

Date of Degree: December 9, 2011

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Nicolas H. Younan

Dissertation Director: Dr. Jenny Q. Du

Title of Study: DETECTION AND CLASSIFICATION OF BURIED RADIOACTIVE MATERIALS

Pages in Study: 97

Candidate for Degree of Doctor of Philosophy

This dissertation develops new approaches for detection and classification of buried radioactive materials. Different spectral transformation methods are proposed to effectively suppress noise and to better distinguish signal features in the transformed space. The contributions of this dissertation are detailed as follows.

1) Propose an unsupervised method for buried radioactive material detection. In the experiments, the original Reed-Xiaoli (RX) algorithm performs similarly as the gross count (GC) method; however, the constrained energy minimization (CEM) method performs better if using feature vectors selected from the RX output. Thus, an unsupervised method is developed by combining the RX and CEM methods, which can efficiently suppress the background noise when applied to the dimensionality-reduced data from principle component analysis (PCA).

2) Propose an approach for buried target detection and classification, which applies spectral transformation followed by noise-adjusted PCA (NAPCA). To meet the requirement of practical survey mapping, we focus on the circumstance when sensor

dwell time is very short. The results show that spectral transformation can alleviate the effects from spectral noisy variation and background clutters, while NAPCA, a better choice than PCA, can extract key features for the following detection and classification.

3) Propose a particle swarm optimization (PSO)-based system to automatically determine the optimal partition for spectral transformation. Two PSOs are incorporated in the system with the outer one being responsible for selecting the optimal number of bins and the inner one for optimal bin-widths. The experimental results demonstrate that using variable bin-widths is better than a fixed bin-width, and PSO can provide better results than the traditional Powell's method.

4) Develop parallel implementation schemes for the PSO-based spectral partition algorithm. Both cluster and graphics processing units (GPU) implementation are designed. The computational burden of serial version has been greatly reduced. The experimental results also show that GPU algorithm has similar speedup as cluster-based algorithm.

ACKNOWLEDGEMENTS

I acknowledge with sincere gratitude the guidance provided by my major advisor Dr. Nicolas H. Younan and my dissertation director Dr. Jenny Q. Du. Without their invaluable guidance and help, this dissertation could not be materialized.

I thank my committee members, Dr. Yi Su and Dr. Erdem Topsakal, for their guidance and fruitful discussion. I would also like to extend my sincere thanks to Dr. Charles Waggoner and Ms. Donna Rogers in the Institute for Clean Energy Technology at Mississippi State University for providing the lab data. I would like to acknowledge U.S. Army Engineer Research and Development Center at Vicksburg, Mississippi, for providing financial support throughout the research.

Finally, special gratitude goes to my parents and my wife for their support and encouragement.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS ii

LIST OF TABLES vi

LIST OF FIGURES viii

CHAPTER

I. INTRODUCTION 1

1.1 Background 1
1.2 Motivation 2
1.2.1 Unsupervised anomaly detection 2
1.2.2 Noise-adjusted principle component analysis 2
1.2.3 Optimized spectral transformation 3
1.2.4 Parallel computation for particle swarm optimization 3
1.3 Contribution 3

II. EXPERIMENT DATA 5

2.1 NaI dataset 5
2.2 Laboratory LaBr dataset 8
2.3 T1 and T2 field datasets 10

III. UNSUPERVISED TARGET DETECTION 12

3.1 Introduction 12
3.2 Proposed Method 12
3.2.1 Classical RX algorithm for anomaly detection 12
3.2.2 Constrained energy minimization algorithm 14
3.3 Experiments 15
3.4 Conclusion 20

IV. NOISE-ADJUSTED PCA FOR SPECTRAL TRANSFORMATION 21

4.1 Introduction 21
4.2 Proposed Method 22

4.2.1 Spectral transformation based feature extraction....22
4.2.1.1 Spectral Bin Energy (SBE)....23
4.2.1.2 Spectral Bin Difference (SBD)....24
4.2.1.3 Spectral Bin Ratios (SBR)....25
4.2.1.4 Spectral Comparison Ratios (SCR)....26
4.2.2 PCA and NAPCA....27
4.2.3 Target detection and evaluation....29
4.3 Experiments....31
4.3.1 Experiment Using the Entire Dataset....31
4.3.2 Uncertainty Analysis....35
4.3.3 Experiment Using Data Containing Difficult Classes Only....40
4.4 Conclusion....41

V. OPTIMIZED SPECTRAL TRANSFORMATION....43

5.1 Introduction....43
5.2 Proposed Method....45
5.2.1 Unconstrained PSO....45
5.2.2 Constrained PSO....47
5.2.3 A simple example....49
5.2.4 Powell's direction set method....50
5.2.5 Uniform bin-width optimization....50
5.2.6 PSO searching for variable bin-widths....51
5.2.7 PSO searching for the optimal number of bins....52
5.2.8 Overall algorithm with two PSOs....53
5.3 Experiments....54
5.3.1 Data and implementation....54
5.3.2 GC and *k*-NN using the original data....57
5.3.3 Uniform bin-width and Powell's method....58
5.3.4 Bin optimization using PSO....58
5.4 Conclusion....64

VI. PARALLEL OPTIMIZATION....66

6.1 Introduction....66
6.2 Proposed Method....68
6.2.1 Parallel algorithm for one PSO on clusters....68
6.2.2 Parallel algorithm for one PSO on GPU....70
6.2.2.1 Data organization....71
6.2.2.2 Random number generation....71
6.2.2.3 Overall algorithm of GPU-implemented PSO....72
6.2.2.4 Parallelization design on GPU....73
6.2.3 Parallel hardware and software....75
6.3 Experiments....76
6.3.1 Cluster parallel implementation....76
6.3.2 GPU parallel implementation....80

6.3.2.1 Running time and speedup versus number of bins 80
6.3.2.2 Running time and speedup versus swarm size...................82
6.4 Conclusion 84

VII. CONCLUSION AND FUTURE WORK 86

REFERENCES 89

LIST OF TABLES

TABLE Page

3.1 The probability of detection and false alarmed rate when the threshold is set as 50% maximum value in dataset T1 20

3.2 The probability of detection and false alarmed rate when the threshold is set as 50% maximum value in dataset T2 20

4.1 Detection and classification accuracy (%) of different methods for the entire dataset...32

4.2 Classification accuracy (%) of seven target classes 33

4.3 Detection and classification accuracy (%) of 0.25 s data..................................34

4.4 Detection and classification accuracy (%) of 0.1 s data....................................35

4.5 F-test for the mean accuracies of the ten methods 39

4.6 T-test for the mean accuracies of the four groups 40

4.7 Detection and classification accuracy (%) of different methods for the dataset containing seven difficult classes...41

5.1 The performance of GC and original k-NN on the two datasets........................55

5.2 The resulting performance when the objective is overall detection accuracy..59

5.3 The resulting performance when the objective is overall classification accuracy..60

5.4 The resulting performance when the objective is multi-objective function...61

5.5 Classification accuracy of DU classes 62

6.1 Accuracy from the best uniform partitions 78

6.2 Detection accuracy from parallel PSO-based methods 78

6.3 Classification accuracy from parallel PSO-based methods 79
6.4 Results of parallel bins selection running time (in second) 82
6.5 Results of parallel PSO running time (in second) 84

LIST OF FIGURES

FIGURE Page

2.1 Original target (buried 15 cm deep) and background spectra. 6

2.2 Original target (buried 23 cm deep) and background spectra. 7

2.3 Original target (buried 30 cm deep) and background spectra. 8

2.4 Spectra of 36 g target. 9

2.5 Spectra of 16 g target. 9

2.6 Spectra of 8 g target. 10

2.7 T1 data illustration (GC). 10

2.8 T2 data illustration (GC). 11

3.1 Target detection for T1 real data. 16

3.2 Target detection for T2 real data. 17

3.3 ROC curve for T1 18

3.4 ROC curve for T2 19

4.1 Original target (buried 15 cm deep) and background spectra (1 s dwell time). 23

4.2 After applying SBE transform to the spectra in Fig. 4.1. 24

4.3 After applying SBD transform to the spectra in Fig. 4.1. 25

4.4 After applying SBR transform to the spectra in Fig. 4.1 26

4.5 After applying SCR transform to the spectra in Fig. 4.1 27

4.6 Decision boundary determined by training data. 30

4.7 Boxplots for 24-fold cross-validation using the entire dataset. 36

5.1 Population distribution at different iterations. (a) Iteration = 1. (b) Iteration = 25. (c) Iteration = 50. (d) Iteration = 75. 49

5.2 The proposed adaptive optimization system. 52

5.3 A learning curve for the SCR transform. 55

5.4 Bin boundaries from five PSO runs for SCR. 56

5.5 Boxplot of detection accuracy (OD) for the testing data..................................63

5.6 Boxplot of classification accuracy (OC) for the testing data. 63

5.7 Boxplot of multi-objective function value (0.5 OD + 0.5 OC) of the testing data...64

6.1 The parallel PSO algorithm diagram..69

6.2 The diagram for the GPU-implemented PSO bins selection.............................73

6.3 The parallel speedup for two objective functions. 77

6.4 Comparision on OD and OC using the testing data. 80

6.5 Speedup performance with different number of bins selected by Cluster 81

6.6 Speedup performance with different number of bins selected by GPU 82

6.7 Speedup performance with different swarm size for Cluster 83

6.8 Speedup performance with different swarm size for GPU 83

CHAPTER I

INTRODUCTION

1.1 Background

Detection and discrimination of radioactive objects have many important applications, such as illicit cargo detection at border crossings [1]-[2], buried target detection within battlefields [3], and nuclear threat discrimination from benign sources [4]. Several approaches have been developed [4]-[6]. Many of these methods use a gamma spectrometer to count the number of emitted gamma photons; detection or classification is achieved based on the measurements of these photons at different energy levels. It is assumed that the collected energy spectrum is significantly different between target and non-target measurements. For instance, one of the most common and simple criteria is the gross count (GC) of photons within a certain spectral range [9]. Another method involves computing the ratio of an unknown measurement with a known and benign measurement, which is referred to as the spectral comparison ratio (SCR) method [4][10]. Characteristics of the ratio can help determine whether the unknown measurement is similar to that of benign measurement; then target discrimination can be achieved.

Due to low energy counts and strong background clutters, the performance of the aforementioned techniques may be poor when a target, e.g., depleted uranium, is buried. Under such circumstance, it is important to develop algorithms that can effectively suppress noise and background interference. In this dissertation, we present several advanced statistical signal processing methods for the detection of buried radioactive materials; these methods can also classify targets buried at different depths even when benign radioactive materials are present.

1.2 Motivation

1.2.1 Unsupervised anomaly detection

In practical applications, the spectra of target and non-target may be unknown. The classical anomaly detection methods are well suited for this situation. The Reed-Xiaoli (RX) algorithm is based on exploiting the difference between a spectral signature and its neighbors. The distance measure is the Mahalanobis distance. Another advanced method is to combine the RX algorithm with the constrained energy minimization (CEM) method. The CEM is a supervised method requiring target spectrum, which can be obtained by selection of the RX output. The CEM performance may be improved by using the data from principle component analysis (PCA)-based dimensionality reduction.

1.2.2 Noise-adjusted principle component analysis

We are more interested in the features in an energy spectrum curve rather than GC. Spectral transformation may help feature extraction. We will investigate three spectral transformation methods, including spectral bin energy (SBE), spectral bin ratio (SBR), and SCR, which can normalize the contribution from background and eliminate the trivial variation from noise. In addition, PCA and noise-adjusted PCA (NAPCA) have the capability of suppressing noise. The difference is NAPCA ranks the PCs by signal-to-noise ratio (SNR) while PCA ranks them by variance. We will investigate the effects of these methods.

1.2.3 Optimized spectral transformation

Since an energy spectrum is usually sparse, we need to combine certain energy channels to gather enough information for feature selection. However, how to partition a spectrum into a number of windows or bins is a challenging problem. A good partition will lead to improved detection and classification accuracy, but inappropriate partition may result in even worse accuracy than without partition. For simplicity purpose, it is assumed that bin partition is non-overlapping and for all the channels. Thus, the number of bins and their bin-widths need to be optimized. An adaptive optimization system using particle swarm optimization (PSO) will help to determine both the optimal number of bins and the corresponding bin-widths simultaneously.

1.2.4 Parallel computation for particle swarm optimization

With modem computational facilities, such as massively parallel processors, the computing time of the PSO algorithm can be greatly reduced. A parallel PSO algorithm distributes computational burden to parallel running processing units such that the algorithm is executed in a timely manner. Compared to the serial version, the parallel PSO can greatly enhance the training speed of the adaptive optimization system.

1.3 Contribution

The specific contributions in this dissertation are summarized as below:

1. Assess the classical anomaly detection methods on buried radioactive material detection. The original RX algorithm performs similarly as the GC method. However, the CEM method performs better if using feature vectors selected from the RX output. This unsupervised method can

suppress the background noise by using the dimensionality-reduced data from PCA.

2. Propose an approach for buried target detection and classification, which applies spectral transformation followed by NAPCA. To meet the requirement of practical survey mapping, we focus on the circumstance when sensor dwell time is very short. The results show that spectral transformation can alleviate the effects from spectral noisy variation and background clutters, while NAPCA, a better choice than PCA, can extract key features for the following detection and classification.
3. Propose an adaptive optimization system with PSO to automatically determine the optimal number of bins and the corresponding optimal varied bin-widths for energy spectral transformation. Two PSOs are incorporated in the system with the outer one being responsible for selecting the optimal number of bins and the inner one for optimal bin-widths. The experimental results demonstrate that using variable bin-widths is better than a fixed bin-width, and PSO can provide better results than the traditional Powell's method.
4. Develop parallel implementation schemes for the PSO-based bin partition algorithm. Both cluster and graphics processing units (GPU) implementations are designed for parallel PSO-based spectral transformation. The computational burden of serial version has been greatly reduced. The experimental results show that GPU has similar speedup as cluster-based algorithm.

CHAPTER II

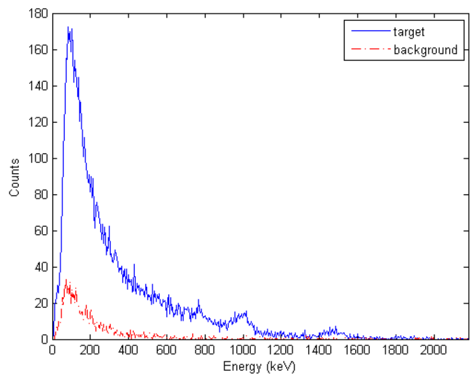
EXPERIMENT DATA

This research focuses on the analysis of data collected by gamma ray spectrometers. In this chapter, we introduce both the laboratory data and real field data used in the experiments.

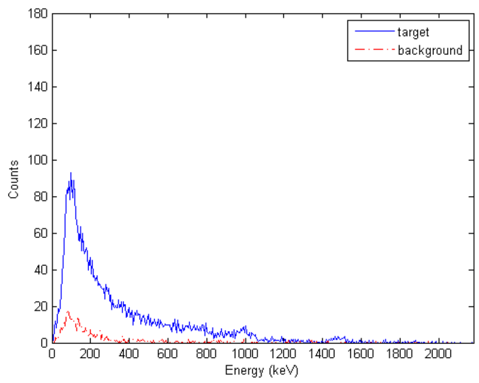
2.1 NaI dataset

Sodium iodide (NaI) is by far the most widely used material for scintillators. It is available in a single crystal form or a more rugged polycrystalline form (used in high vibration environments, e.g., wireline logging in the oil industry). It also has other applications including nuclear medicine, basic research, environmental monitoring, and aerial surveys [85]. Some researchers use NaI scintillator for portal monitors applications. Experimental data and computer simulations are presented for gamma-ray detection for homeland security applications at international borders [1][2][5]. In our studies, laboratory data was collected using a 10×10×40 cm NaI scintillation detector. The measured spectra covered the energy range from 0 keV to 2160.0 keV. The target was depleted uranium with 4.3 kg mass. The background consisted of construction sand. Natural ore was also present, considered a benign material.

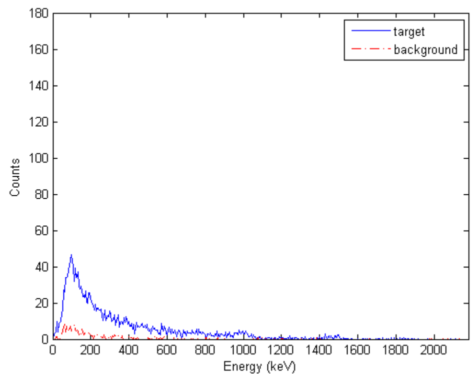
Figs. 2.1-2.3 show example spectra of targets buried at 15cm, 23cm, and 30cm depth, respectively. Counting time was changed from 1 s, 0.5 s, 0.25 s, to 0.1 s. We can see that as the depth is increased and counting time is decreased, the features around 768keV and 1001keV disappear, and the difference between the target spectrum and background spectrum become insignificant. Therefore, detection of buried targets from the data collected with short counting time is a very difficult task.



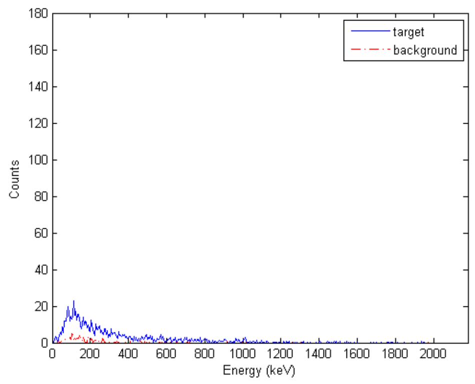
(a) 15 cm depth and 1 s dwell time



(b) 15 cm depth and 0.5 s dwell time

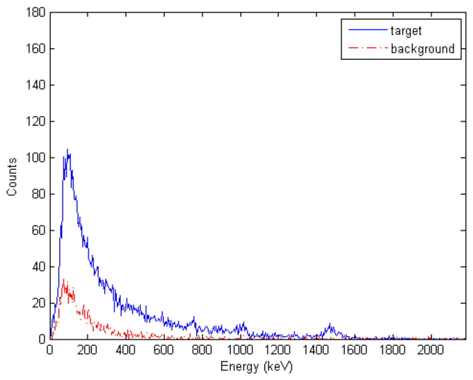


(c) 15 cm depth and 0.25 s dwell time

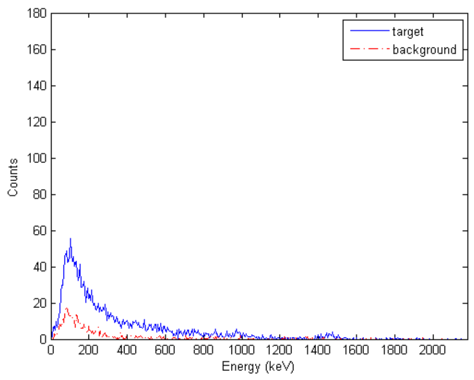


(d) 15 cm depth and 0.1 s dwell time

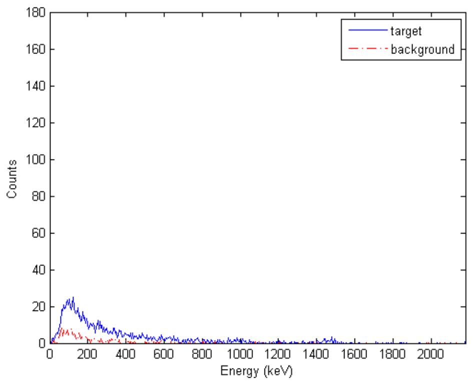
Figure 2.1 Original target (buried 15 cm deep) and background spectra.



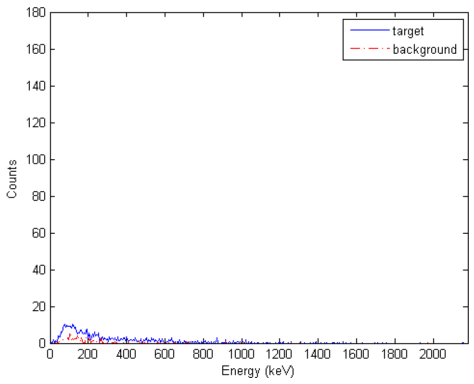
(a) 23 cm depth and 1 s dwell time



(b) 23 cm depth and 0.5 s dwell time

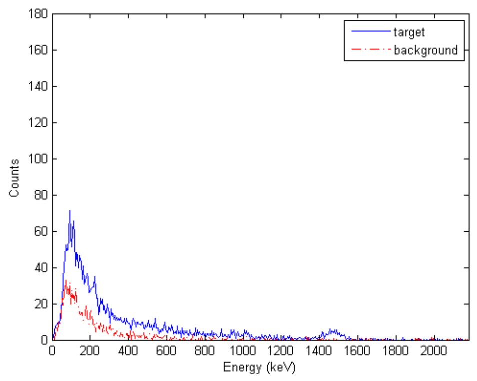


(c) 23 cm depth and 0.25 s dwell time

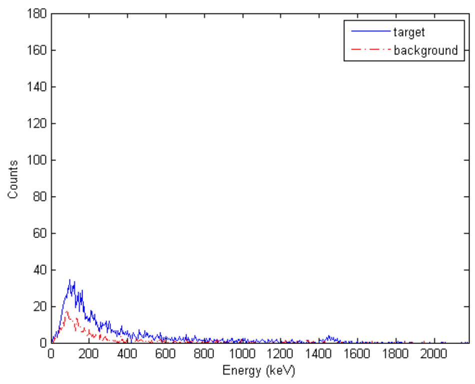


(d) 23 cm depth and 0.1 s dwell time

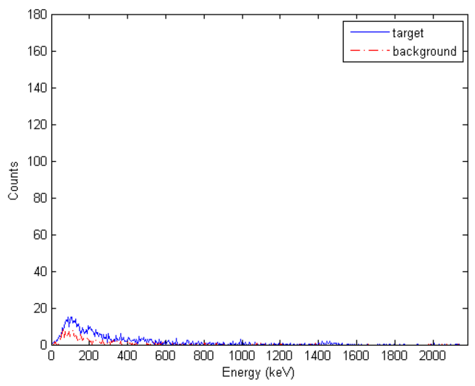
Figure 2.2	Original target (buried 23 cm deep) and background spectra.



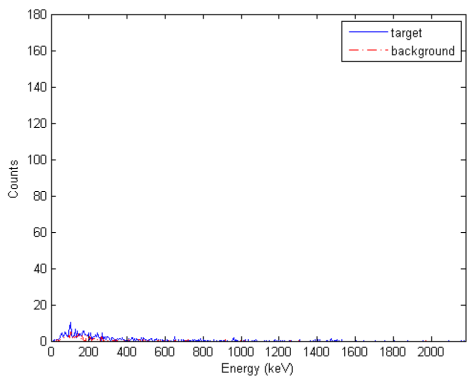
(a) 30 cm depth and 1 s dwell time



(b) 30 cm depth and 0.5 s dwell time



(c) 30 cm depth and 0.25 s dwell time



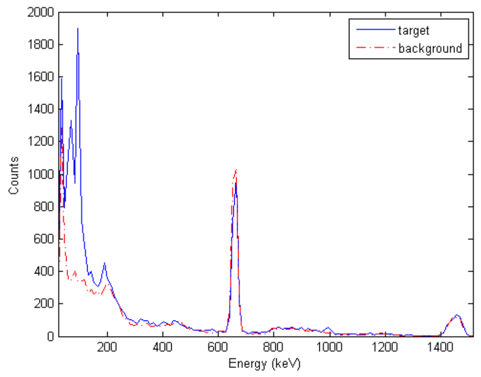
(d) 30 cm depth and 0.1 s dwell time

Figure 2.3 Original target (buried 30 cm deep) and background spectra.

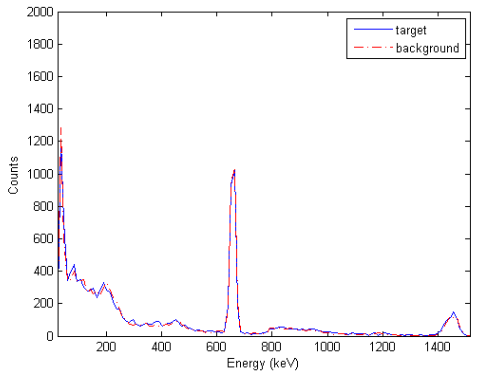
2.2 Laboratory LaBr dataset

Lanthanum bromide (LaBr) scintillators offer improved energy resolution than NaI scintillators and excellent temperature characteristics. Due to its high resolution, LaBr scintillators perform well in the recent gamma spectroscopy-based detection and identification systems used in the homeland security market. In our study, a dataset was collected from 21.1 to 1516.8 keV with 127 channels by an LaBr scintillator. It consisted of ten classes with different mass of 8, 16 and 36 g, different depths of 0, 15 and 30 cm, and background (i.e., construction sand). Each class had 50 measurements with 10 s counting period. The calibration source is Cs-137, yielding significant peaks at 33 keV and 662 keV.

Figs. 2.4-2.6 show example spectra of 32 g, 16g, and 8 g targets, respectively. When targets are buried 30cm deep, their spectra are quite similar to those of background.

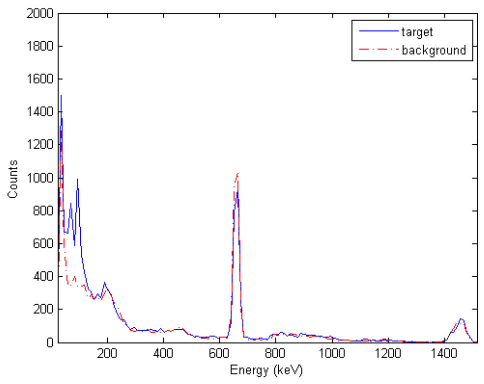


(a) Buried at 0 cm

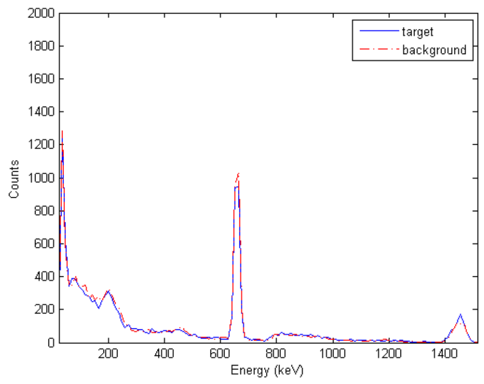


(b) Buried at 30 cm

Figure 2.4 Spectra of 36 g target.

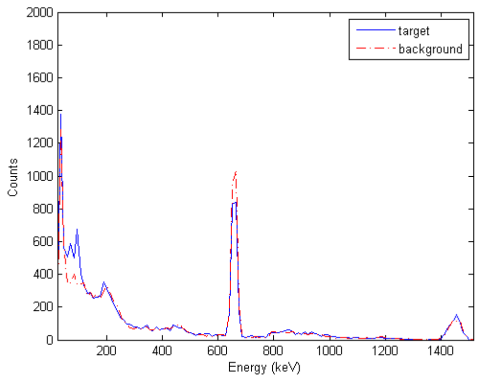


(a) Buried at 0 cm

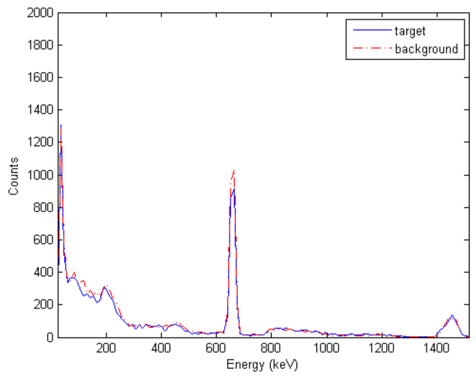


(b) Buried at 30 cm

Figure 2.5 Spectra of 16 g target.



(a) Buried at 0 cm



(b) Buried at 30 cm

Figure 2.6 Spectra of 8 g target.

2.3 T1 and T2 field datasets

Two datasets T1 and T2 used in our experiment were collected from a real field with an NaI scintillation detector. T1 dataset consists of 128 channels while T2 has 1024 channels. In Figs. 2.7-2.8, these datasets are displayed in color based on GC, where a red pixel represents a potential target and a blue pixel represents background area.

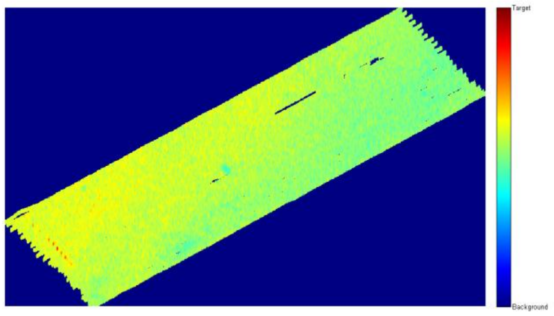


Figure 2.7 T1 data illustration (GC).

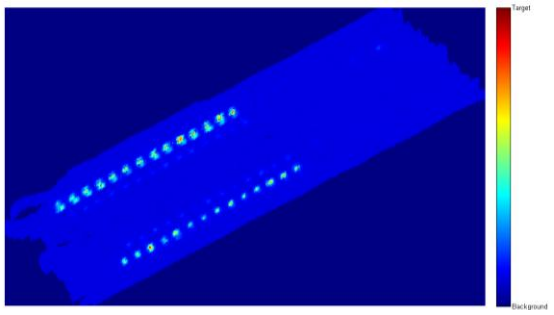


Figure 2.8 T2 data illustration (GC).

CHAPTER III

UNSUPERVISED TARGET DETECTION

3.1 Introduction

Matched filter methods play an important role in target detection. These algorithms are of interest because: (1) they can outperform other existing algorithms due to their capability of background suppression; (2) they are suitable to radioactive materials detection in an unknown circumstance since they require least prior information. An example is the constrained energy minimization (CEM) filter, which has good merit on maximizing the target signature response while suppressing the undesired background signature response [13],[14]-[18]. The idea was first derived from the minimum variance distortionless response (MVDR) beamformer in array processing [15],[16] and was later used in chemical remote sensing [17].

The well-known RX algorithm [11],[12] is a detector for anomaly detection. It is originally developed for multispectral imagery by Reed and Yu in 1993 [19]. The RX-algorithm is based on exploiting the difference between the spectral signatures and its neighbors. It actually is a matched filter in an unsupervised fashion.

3.2 Proposed Method

3.2.1 Classical RX algorithm for anomaly detection

In the conventional RX-algorithm, a nonstationary local mean is subtracted from each spectral pixel within a fixed window. The local mean is obtained by sliding a double concentric window, which consists of a small inner window centered within a larger outer window over each pixel in an image, and the mean is calculated from the spectral pixels falling between the inner and the outer window. The size of the inner window is usually assumed to be the size of the target of interest. The residual signal after mean subtraction is assumed to approximate a zero-mean Gaussian random process. Let each input signal vector be denoted by $\mathbf{x} = [x_1, x_2, ..., x_n]^T$. A two-hypothesis test is formulated as

$$H_0 : \mathbf{x} = \mathbf{n} \tag{3.1}$$

$$H_1 : \mathbf{x} = a\mathbf{s} + \mathbf{n} \tag{3.2}$$

where **n** is noise vector represents the background noise process. And **s** is feature signal represents the anomaly signal. *a* is constant which larger than 0 under hypothesis H_1 and equals to 0 under H_0.

The target signature and background covariance are assumed to be unknown. This model assumes that the data come from two normal probability density functions with the same covariance matrix but different means [61]. Under H_0, the data (background clutter) are modeled as $N(\mathbf{0}, \mathbf{C}_b)$, and under H_1 the data are modeled as $N(\mathbf{s}, \mathbf{C}_b)$. It should be noticed that an important assumption in the RX-algorithm is that the background and target have the same covariance matrix. Generally, this is not a valid model if a particular target structure is to be detected. A more appropriate model would have two different covariance structures — one for anomaly (which could be target or background clutter) and one for background. However, the covariance structure for the anomaly cannot be estimated in reality, since the statistical structure of the anomaly signals cannot be defined. Therefore, the same covariance structure for anomaly and background is adopted. The basic RX algorithm is the benchmark anomaly detection algorithm and is can be written as below:

$$w_{RX} = (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{R}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{3.3}$$

where **R** is the background covariance matrix estimated from the surrounding background data, and $\boldsymbol{\mu}$ is the estimated background clutter sample mean [62]. A threshold η is to be assigned such that detection can be achieved on the RX output.

3.2.2 Constrained energy minimization algorithm

The key to the CEM algorithm is to determine a weight vector $\mathbf{w} = [w_1, w_2, \cdots w_k]^T$ that suppresses the unknown and undesired background data while enhancing that of the known target signature vector **d**. The CEM operator is defined by two constraints. The first constraint is to minimize the total output energy. The energy of an individual signal summed across the energy range can be represented by a scalar value y_i.

$$y_i = \sum_{k=1}^{l} w_k r_{ik} \quad i=1,2,...,q \tag{3.4}$$

where **w** is the vector of weights and q is the total number of pixels, and r_{ik} is the energy counts recorded in the *k*-th channel for the *i*th signal vector. The formula can be rewritten as vector notation:

$$y_i = \mathbf{w}^T \mathbf{r}_i \tag{3.5}$$

The second constraint is that when applied to the target spectrum, $y_i = \mathbf{w}^T \mathbf{d} = 1$. This constrained minimization problem can be solved and the CEM detector is

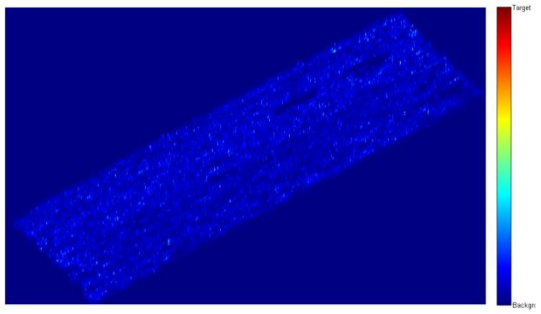
$$w_{CEM} = \frac{\mathbf{d}^T \mathbf{R}^{-1}}{(\mathbf{d}^T \mathbf{R}^{-1} \mathbf{d})} \tag{3.6}$$

where **R** is the covariance matrix of the pixel vectors. A threshold η is required to complete target detection [13].

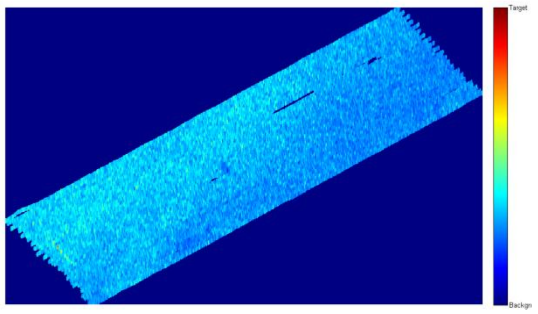
3.3 Experiments

We implemented the popular method GC as benchmark method as shown in Figs. 2.7-2.8. The CEM method was employed to compare with GC. However, CEM is a supervised method; how to choose the feature vector **d** is challenging. Here, we applied an unsupervised method, such as the RX algorithm, to choose the top 10 vectors with the largest outputs, and calculated their mean as the **d** for the CEM. We also applied PCA to reduce data dimensionality (only the first two PCs were used), followed by the RX and CEM-based target detection.

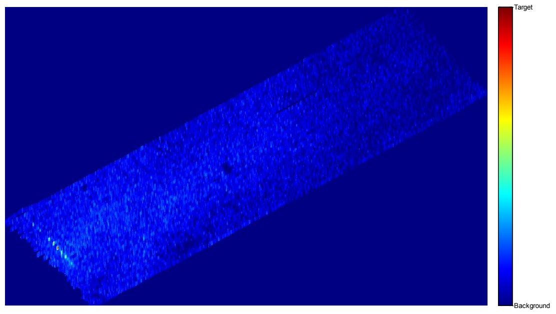
Figs. 3.1-3.2 show the detection maps for T1 and T2 datasets. Compared to Figs. 2.7-2.8, the RX algorithm itself did not offer much advantage. However, when the RX and CEM were combined in Fig. 3.1(c), the detection maps were significantly improved in terms of background suppression; the performance was further improved if PCA was used for dimensionality reduction as a preprocessing step in Figs. 3.1(d) and (e). For the T2 dataset, improvement was obvious when using PCA in Figs. 3.2 (d) and (e).



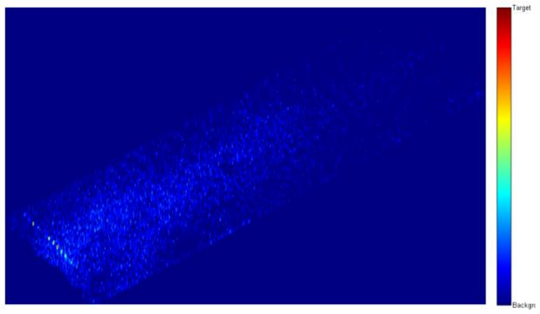
(a) RX method with a dual window
(inner window 7x7, outer window 11x11)



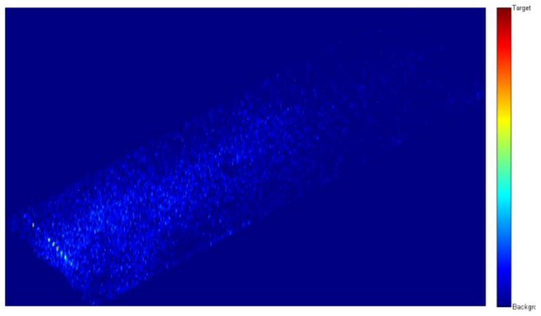
(b) RX method
(all pixels are used for background estimation)



(c) CEM method
(Feature selected by the RX method in (b))

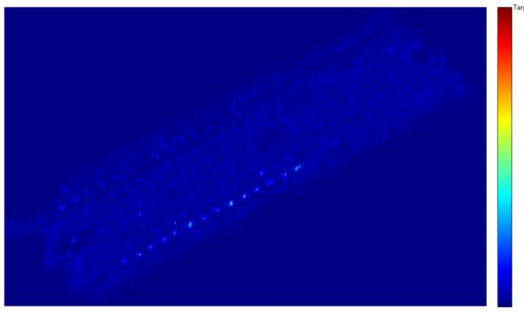


(d) CEM method after PCA
(Feature selected by the RX method in (b))

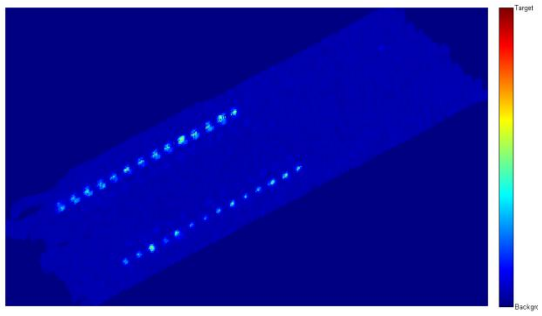


e) CEM method after energy windowing and PCA
(Feature selected by the RX method in (b))

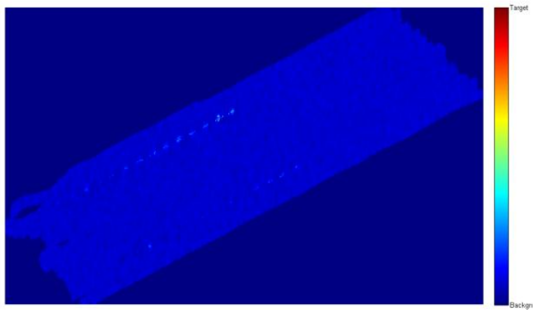
Figure 3.1 Target detection for T1 real data.



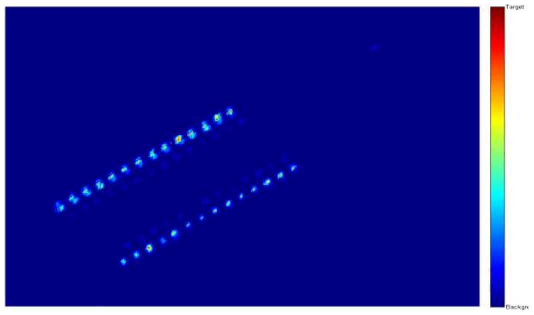
(a) RX method with a dual window
(inner window 7x7, outer window 11x11)



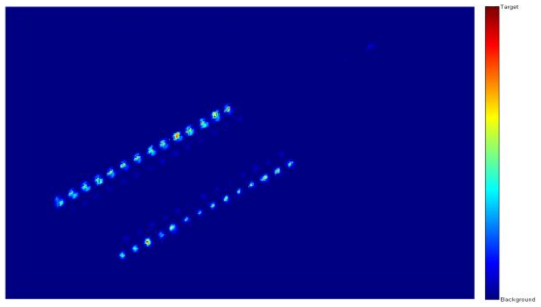
(b) RX method
(all pixels are used for background estimation)



(c) CEM method
(Feature selected by the RX method in (b))



(d) CEM method after PCA
(Feature selected by the RX method in (b))



(e) CEM method after energy windowing and PCA
(Feature selected by the RX method in (b))

Figure 3.2 Target detection for T2 real data.

Receiver operating characteristics (ROC) curves represent detection probability versus false-alarm rates. It could also provide quantitative performance comparison. P_d and P_f are the corresponding probability of detection and false alarm rate, respectively, which can be defined as:

$$P_d = \frac{N_d}{N_t}, P_f = \frac{N_f}{N_c}$$

where N_d is the number of detected target samples, N_t is the number of total target samples, N_f is the number of false alarms, and N_c represents the number of detected samples. As shown in Figs. 3.3-3.4, the CEM with energy windowing and PCA showed significantly improved performance over the other three methods; the CEM method after energy windowing and PCA significantly outperformed the conventional RX and CEM alone at lower false-alarm rates.

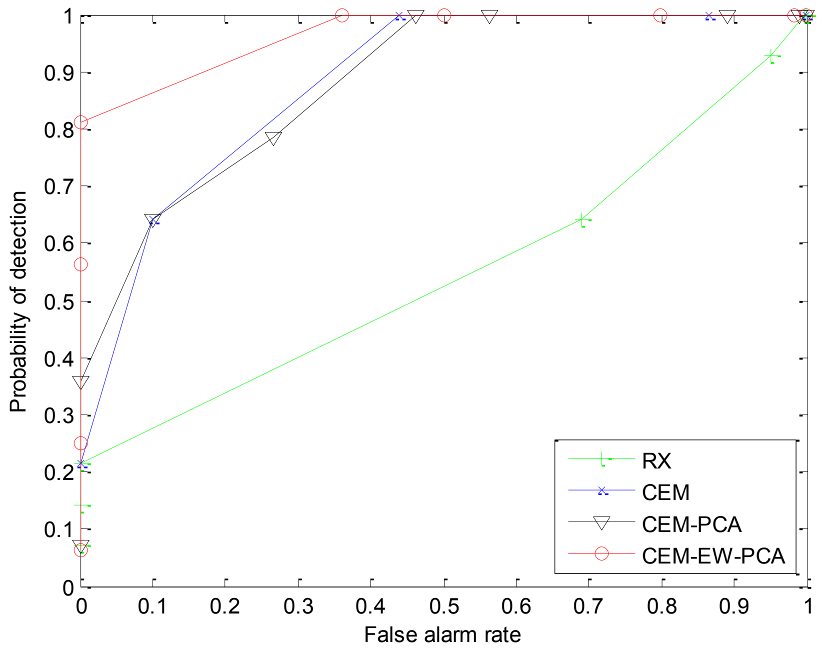


Figure 3.3 ROC curve for T1

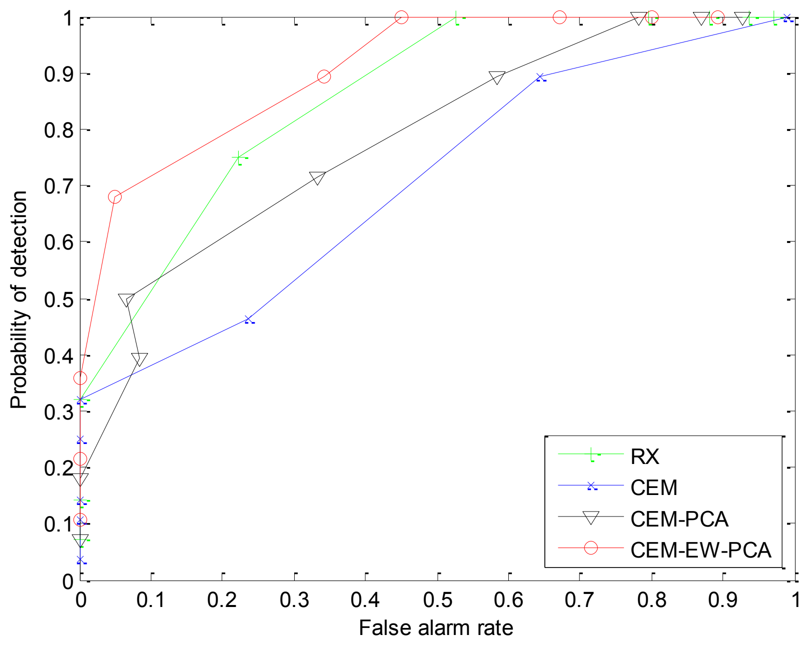


Figure 3.4 ROC curve for T2

We further set 50% of the maximal output value as the cut-off threshold to test the performance of different algorithms on target detection and background suppression. Tables 3.1-3.2 list the probability of detection and false alarm rate using the RX, CEM, CEM after PCA, and CEM after energy windowing and PCA. For T1 dataset, the false alarm rates for RX and CEM were 0.95 and 0.99. But for CEM after taking the PCA or energy windowing, the false alarm was decreased to 0.46 and 0.36. This shows our algorithm performed well on background suppression. For T2 dataset, CEM method could achieve the minimum false alarm rate but its detection probability is also low. But for CEM after PCA or energy windowing, false alarm rate is low; in the mean time, their probability of detection is higher.

Table 3.1 The probability of detection and false alarmed rate when the threshold is set as 50% maximum value in dataset T1

RX		CEM		CEM after PCA		CEM after energy window and PCA	
P_d	P_f	P_d	P_f	P_d	P_f	P_d	P_f
0.93	0.95	1.00	0.99	1.00	0.46	1.00	0.36

Table 3.2 The probability of detection and false alarmed rate when the threshold is set as 50% maximum value in dataset T2

RX		CEM		CEM after PCA		CEM after energy window and PCA	
P_d	P_f	P_d	P_f	P_d	P_f	P_d	P_f
0.75	0.22	0.14	0	0.50	0.07	0.68	0.05

3.4 Conclusion

We implement the RX and CEM detection method in T1 and T2 datasets. CEM is a supervised method requiring target signature being known in advance, so we applied an unsupervised method, such as the RX algorithm, to choose target feature for the CEM. We also applied PCA to reduce data dimensionality, followed by the RX and CEM-based target detection. Another way is applying energy windowing to transform the feature space. Using the energy-windowing-transformed data, the detection performance was significantly improved in terms of better background suppression. The performance improvement can be further illustrated using ROC-based quantitative analysis.

CHAPTER IV

NOISE-ADJUSTED PCA FOR SPECTRAL TRANSFORMATION

4.1 Introduction

PCA is a popular multivariate statistical technique. It was invented in 1901 by Karl Pearson [21]. The goal is to extract the important information from the data and to represent it as a set of new orthogonal bases called principal components (PCs). The number of PCs should be less than or equal to the number of original data dimension. This transformation is defined in such a way that the first principal component has the highest variance among all the variables. PCA is a versatile technique and has been used widely in signal processing for various applications such as dimensionality reduction, data compression, and feature extraction.

However, it is obvious that both signal and noise can contribute to data variance; it is possible for a PC with lower ranking to include more important signal features than a PC with higher ranking. In order to deal with this problem, Green *et al.* [22] developed a maximum noise fraction (MNF) transformation based on maximization of SNR, so that the transformed principal components are ranked by SNR rather than variance as used in a PCA. This MNF transformation was later developed by Lee *et al.* in [23]. Based on a two stage processes which consists of a noise-whitening process and a PCA to achieve what the MNF transform does, this new derived transform is referred to as a noise adjusted principal components analysis (NAPCA). Since noise variances in different bands are whitened by NAPCA prior to PCA, maximizing the noise-whitened data variance results in maximizing SNR. Therefore, NAPCA is essentially equivalent to MNF and can be viewed as a variant of MNF. Some researchers found that SNR is a better metric than variance to gauge the actual signal information contained in major PCs [24]. In this way, a PC with higher ranking always contains more signal information and less noise than a PC with lower ranking in NAPCA transformed data.

4.2 Proposed Method

4.2.1 Spectral transformation based feature extraction

Previous methods of radioactive target detection mainly analyze the number of gamma-ray counts in certain energy channels or windows [1]-[4],[9],[10]. Such deterministic methods have disadvantages. First, the measurement hardware always has some degree of variability and uncertainty in the number of counts detected, introducing noise to the collected data. Second, when measurements are sparse, it is more difficult to observe the features (e.g., peaks) due to counting randomness. In addition, when the target is buried, the background has a significant interfering impact on the collected spectrum. Fig. 4.1 shows a buried target and background spectra. To deal with such practical spectra more efficiently, we deploy statistical approaches in our research; in addition to energy windowing or bin partition, the collected spectra are transformed into another subspace before applying a statistical detection or classification algorithm such that target features become more distinguishable.

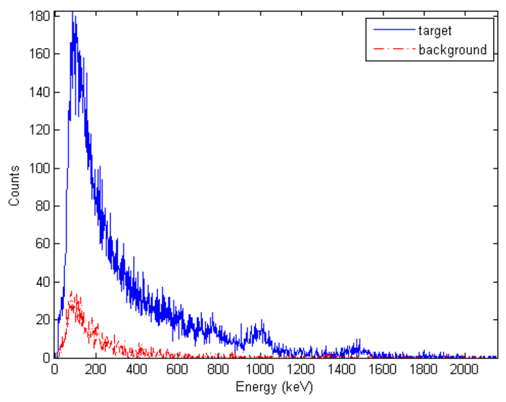


Figure 4.1 Original target (buried 15 cm deep) and background spectra (1 s dwell time).

4.2.1.1 Spectral Bin Energy (SBE)

In the spectral bin energy (SBE) transform in Eq. (4.1), the input spectral measurements are divided into *J* bins:

$$S_{SBE}(j) = \sum_{i \in bin(j)} P(i) \quad \text{for} \quad j = 1,2,\ldots J \tag{4.1}$$

where *P(i)* represents the count in the *i*-th channel. The sum of energy counts within each bin is computed, reducing the data dimensionality. Actually, it is the basic energy windowing method. This transform is especially useful for spectrum measurements that are significantly sparse, such as those measured over very short time periods, because it summarizes the information within the spectrum that may not be immediately obvious. It also helps smooth out the noise. The spectra after SBE transform are shown in Fig. 4.2.

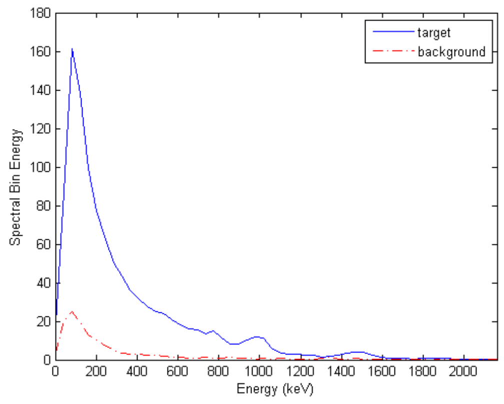


Figure 4.2 After applying SBE transform to the spectra in Fig. 4.1.

4.2.1.2 Spectral Bin Difference (SBD)

The spectral bin difference (SBD) transform computes the difference of total counts in each bin between a current and a previously measured background spectrum. The spectrum is divided into *J* bins. The SBD transform is defined as

$$S_{SBD}(k) = S_{SBE}(j) - S_{SBE}^{b}(j) \quad \text{for} \quad j = 1,2,\ldots J \tag{4.2}$$

where S_{SBE}^{b} and S_{SBE} are the SBE-transformed known background and observed measurements, respectively. Fig. 4.3 illustrates the difference between a background measurement and a target measurement. Notice that the SBD-transformed spectrum is close to 0 when the measurement is similar to that of background. On the other hand, the difference for the target measurement is dramatically dissimilar and more distinguishable from the SBD-transformed background measurement.

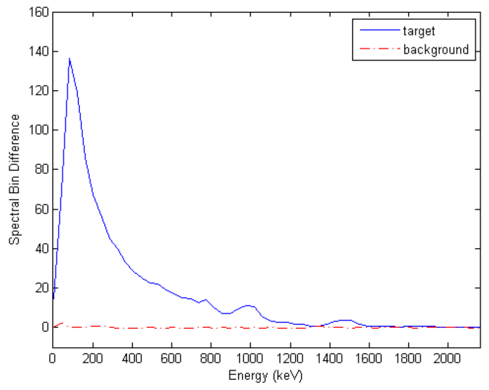


Figure 4.3 After applying SBD transform to the spectra in Fig. 4.1.

4.2.1.3 Spectral Bin Ratios (SBR)

Spectral bin ratios (SBR) method transforms the spectrum based on its ratio with a previously measured background spectrum. Similar to SBD, the spectrum is divided into several bins. The energy within each bin is computed for both the observed spectrum and the known background spectrum, and the ratio of these two bins is computed as:

$$S_{SBR}(j)=\frac{S_{SBE}(j)}{S_{SBE}^{b}(j)} \quad \text{for} \quad j=1,2,\cdots,J \tag{4.3}$$

Fig. 4.4 illustrates a comparison of the ratios of the background measurement and the target measurement in Fig. 4.1 using another background measurement (e.g, mean measurement of background). Notice that the background ratio is close to 1 across the spectrum. This is expected since all background measurements should have a similar fraction of energy in each energy bin. On the other hand, the ratio for target measurement is different tremendously and easier to be separated from the background ratio.

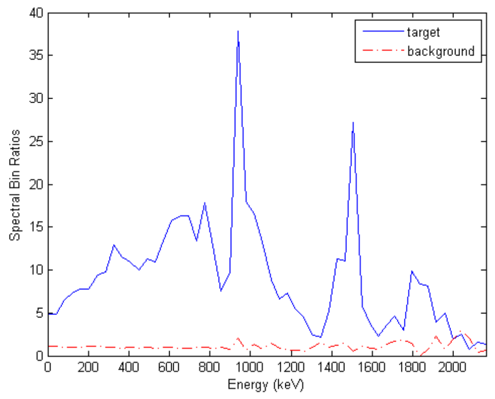


Figure 4.4 After applying SBR transform to the spectra in Fig. 4.1.

4.2.1.4 Spectral Comparison Ratios (SCR)

The fourth transform we investigate is the spectral comparison ratios (SCR) method [9],[10]. As with the SBD and SBR methods, a previously measured background spectrum is required. Both the known background spectrum and observed spectrum are divided into *J* bins. One bin (usually the first bin) is chosen as a reference. The SCR can then be computed using the following equation:

$$S_{SCR}(j-1) = S_{SBE}(1) - \frac{S^b_{SBE}(1)}{S^b_{SBE}(j)} S_{SBE}(j) \quad \text{for} \quad j = 2,3,\ldots J \tag{4.4}$$

Given an observed spectral measurement, this method measures how closely the spectrum matches that of the background on the basis of ratios. If the observed spectrum is very close to the calibrated background measurement, the SCR should be close to 0 across the spectrum. Otherwise, if the measurement is of the target, the magnitude of the SCR should be significantly different from 0. Such differences offer a set of features that can be used to better discriminate the target from the background measurements. Fig. 4.5 shows the resulting spectra in Fig. 4.1 after the SCR transform, where the background spectrum is close to 0 while the target spectrum is rapidly changed but unequal to 0.

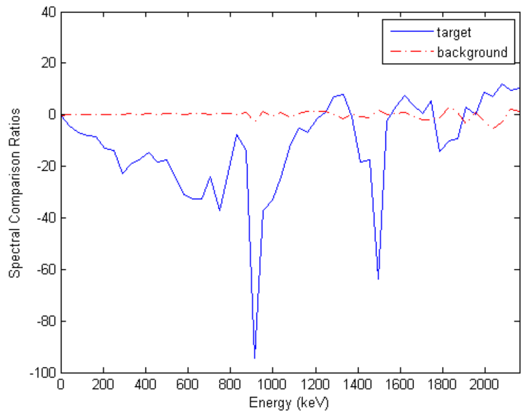


Figure 4.5 After applying SCR transform to the spectra in Fig. 4.1.

4.2.2 PCA and NAPCA

PCA ranks PCs in terms of data variance. Consider the observation model

$$\mathbf{r} = \mathbf{x} + \mathbf{n} \tag{4.5}$$

where r is an energy spectral measurement with data dimensionality L (i.e., the number of spectral channels), $\mathbf{x}$ is a signal vector, and n represents the uncorrelated additive noise. Let $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_L]$ and $\mathbf{\Lambda} = diag\{\lambda_1, \lambda_2, \cdots, \lambda_L\}$ be the eigenvector and eigenvalue matrices of the data covariance matrix $\mathbf{\Sigma}$, where $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_L$ are L eigenvectors of size $L \times 1$ and $\lambda_1, \lambda_2, \cdots, \lambda_L$ are the corresponding L eigenvalues, i.e.,

$$\mathbf{V}^{\mathrm{T}}\mathbf{\Sigma}\mathbf{V} = \mathbf{\Lambda} \tag{4.6}$$

Then, the PC images can be calculated from

$$\mathbf{r}_{\mathbf{PCA}} = \mathbf{\Lambda}^{-1/2}\mathbf{V}^{\mathrm{T}}(\mathbf{r} - \mathbf{m}) \tag{4.7}$$

where $\mathbf{m}$ is the data mean. Assume $\lambda_1 \geq \lambda_2 \geq, \cdots, \geq \lambda_L$, the variances of the L PC images of the transformed data using $\mathbf{r}_{\mathbf{PCA}}$ are $\lambda_1, \lambda_2, \cdots, \lambda_L$, respectively.

NAPCA ranks PCs in terms of SNR. It can be performed with two steps. The first step conducts noise-whitening to the original data, and the second step performs the ordinary PCA to the noise-whitened data. Since the noise variance is unity in the noise-whitened data, the resultant PCs are in the order of SNR. Let $\mathbf{\Sigma_n}$ be the noise covariance matrix and **F** be the noise-whitening matrix such that

$$\mathbf{F}^{\mathbf{T}} \sum_{\mathbf{n}} \mathbf{F} = \mathbf{I} \tag{4.8}$$

where **I** is the identity matrix. Transforming **Σ** by **F**, i.e.,

$$\mathbf{F}^{\mathbf{T}} \sum \mathbf{F} = \sum_{\mathbf{n\_adj}} \tag{4.9}$$

where $\mathbf{\Sigma_{n\_adj}}$ is the covariance matrix with the noise being whitened. Finding a matrix **G** such that

$$\mathbf{G}^{\mathbf{T}} \sum_{\mathbf{n\_adj}} \mathbf{G} = \mathbf{I} \tag{4.10}$$

Then, the operator for NAPCA can be constructed by

$$\mathbf{r}_{\mathbf{NAPCA}} = \mathbf{G}^{\mathbf{T}}\mathbf{F}^{\mathbf{T}}(\mathbf{r} - \mathbf{m}) \tag{4.11}$$

The major difficulty in performing NAPCA is having an accurate noise covariance matrix $\mathbf{\Sigma_n}$, which is difficult in general. The following method is adopted in our research for its simplicity and effectiveness [25]. Let **Σ** be decomposed as

$$\sum = \mathbf{DED} \tag{4.12}$$

where $\mathbf{D} = diag\{\sigma_1, \sigma_2, \cdots, \sigma_L\}$ is a diagonal matrix with σ_l^2 being the diagonal elements of **Σ**, which is the variance of the *l*-th original channel, and **E** is the correlation coefficient matrix whose *ij*-th element represents the correlation coefficient between the *i*-th and *j*-th channels. Similarly, in analogy with the decomposition of **Σ**, its inverse $\mathbf{\Sigma^{-1}}$ can be also decomposed as

$$\sum^{-1} = \mathbf{D}_{\Sigma^{-1}}\mathbf{E}_{\Sigma^{-1}}\mathbf{D}_{\Sigma^{-1}} \tag{4.13}$$

where $\mathbf{D}_{\Sigma^{-1}} = diag\{\zeta_1, \zeta_2, \cdots, \zeta_L\}$ is a diagonal matrix with ζ_l^2 being the diagonal elements of $\mathbf{\Sigma^{-1}}$ and $\mathbf{E}_{\Sigma^{-1}}$ is a matrix similar to **E** with the diagonal elements being one and all the off-diagonal elements being within $(-1, 1)$. It turns out that ζ_l^2 is the reciprocal of a good noise variance estimate of the *l*-th channel. Therefore, the noise covariance matrix $\mathbf{\Sigma_n}$ can be estimated by a diagonal matrix $\mathbf{\Sigma_n} = diag\left\{\zeta_1^{-2}, \zeta_2^{-2}, \cdots, \zeta_L^{-2}\right\}$. It is worth mentioning that data dimensionality may be reduced to *J* if energy bins are applied in spectral transformation.

4.2.3 Target detection and evaluation

To extract the primary features from data contaminated by noise and background clutter, PCA or NAPCA can be applied to the transformed spectra. The first several PCs are kept and used in the detection and classification step. The *k*NN clustering technique is applied to the PCs. For detection, the reduced set of features can be classified into two classes, e.g., target and non-target, using the *k*NN. Fig. 4.6 illustrates an example that shows the decision boundary created by the *k*NN method. In this case, there are a total of four background measurements and eight target measurements. The decision boundary consists of the points whose average distances to the *k* nearest target samples and to the *k* nearest background samples are the same. We choose *k* to be 2 in this example. All the test measurements to one side of the decision boundary will be detected as the background, and all measurements to the other side will be detected as the target. Similarly, multiple-class classification is achieved with multiple boundaries.

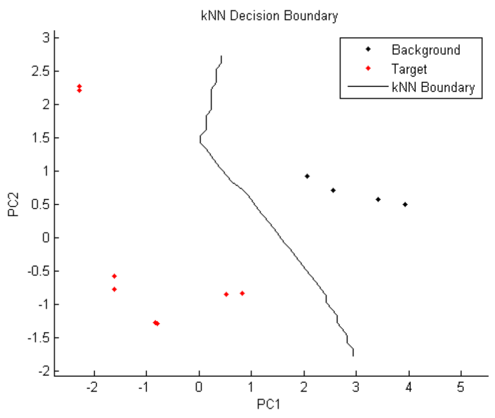


Figure 4.6 Decision boundary determined by training data.

Detection performance is quantified with target detection (TD) accuracy, non-target detection (NTD) accuracy, and overall detection (OD) accuracy. In addition, targets buried at different depth can be considered as different classes, and classification accuracy can be quantified using target classification (TC) accuracy and overall classification (OC) accuracy. The five metrics are defined as:

$$\text{TD} = \frac{\text{number of accurately detected target samples}}{\text{number of target samples}} \tag{4.14}$$

$$\text{NTD} = \frac{\text{number of accurately detected nontarget samples}}{\text{number of nontarget samples}} \tag{4.15}$$

$$\text{OD} = \frac{\text{number of accurately detected samples}}{\text{number of overall samples}} \tag{4.16}$$

$$\text{TC} = \frac{\text{number of accurately classified target samples}}{\text{number of target samples}} \tag{4.17}$$

$$\text{OC} = \frac{\text{number of accurately classified samples}}{\text{number of overall samples}} \tag{4.18}$$

The T-fold cross-validation divided the original samples into T subsamples. Of the T subsamples, a single subsample was taken for validation and the remaining $T-1$ subsamples were used as training data. The cross-validation process was then repeated T

times, with each of the T subsamples used exactly once as the validation data. The T results from the folds were averaged to produce a single estimation. All samples were used for both training and validation, and each sample was used for validation exactly once.

4.3 Experiments

4.3.1 Experiment Using the Entire Dataset

*k*NN ($k = 4$) was applied for T-fold cross-validation ($T = 24$). The T-fold cross-validation divided the original samples into T subsamples. Of the T subsamples, a single subsample was taken for validation and the remaining $T - 1$ subsamples were used as training data. The cross-validation process was then repeated T times, with each of the T subsamples used exactly once as the validation data. The T results from the folds were averaged to produce a single estimation. All samples were used for both training and validation, and each sample was used for validation exactly once.

Table 4.1 Detection and classification accuracy (%) of different methods for the entire dataset

Methods	TD	NTD	OD	TC	OC
GC	83.7	63.5	77.6	80.3	63.8
SBE	91.6	83.3	89.1	81.8	75.5
SBR	90.5	40.3	75.4	69.0	54.6
SCR	89.3	69.4	83.3	77.4	70.4
SBE-PCA	93.5	88.9	92.1	72.6	71.3
SBR-PCA	73.8	62.5	70.4	48.8	41.7
SCR-PCA	89.3	73.6	84.6	64.9	61.7
SBE-NAPCA	94.3	88.3	92.5	87.1	77.0
SBR-NAPCA	93.9	87.3	91.9	87.1	76.7
SCR-NAPCA	93.2	77.8	88.6	81.6	75.7

Table 4.1 tabulates the average detection accuracy of cross-validation by considering all the seven target classes as a single class and all the three non-target classes as the other. The NAPCA-based methods provided higher detection accuracy than the PCA-based methods. In particular, the three spectral transformation methods in conjunction with NAPCA could improve the performance of the methods applied on the original data. For instance, the SBR transform could not improve the performance if using PCA; however, it could result in significant improvement with NAPCA. As for the SCR method, the overall classification accuracy was lower than the NAPCA-based method, but similar to the PCA-based method; with PCA (i.e., SCR-PCA), the overall detection accuracy was increased from 83.3% to 84.6%; if NAPCA was employed (i.e.,

SCR-NAPCA), it was further increased to 88.6%. The GC method generally yielded low accuracy.

Table 4.2 Classification accuracy (%) of seven target classes

	15cm	23cm	30cm	45cm	60cm	75cm	90cm
GC	100.0	100.0	100.0	95.2	23.8	47.6	95.2
SBE	100.0	77.3	54.5	90.9	50.0	100.0	100.0
SBR	100.0	79.2	79.2	54.2	12.5	87.5	70.8
SCR	100.0	70.8	79.2	83.3	37.5	95.8	75.0
SBE-PCA	83.3	45.8	33.3	91.7	54.2	100.0	100.0
SBR-PCA	91.7	50.0	87.5	20.8	12.5	62.5	16.7
SCR-PCA	87.5	37.5	66.7	54.2	50.0	100.0	58.3
SBE-NAPCA	95.0	95.0	75.0	100.0	50.0	95.0	100.0
SBR-NAPCA	95.2	95.2	76.2	100.0	47.6	95.2	100.0
SCR-NAPCA	90.5	61.9	66.7	95.2	66.7	100.0	90.5

Classification was also conducted where targets buried at different depths were considered different classes. As shown in Table 4.1, NAPCA with SBE provided the highest target classification accuracy (i.e., 87.1%) when classifying the target buried at seven different depths and the highest overall classification accuracy (i.e., 77.0%) when classifying all the ten classes including natural ore and background.

The detailed classification results of the seven target classes are listed in Table 4.2 (corresponding to the TC in Table I). The 60 cm DU was difficult to be classified because the spectra were close to those of the natural ore buried 45 cm and 75 cm deep.

Interestingly, the four methods applied on the original data (i.e., GC, SBE, SBR, SCR) could provide 100% accuracy for 15 cm DU, while the NAPCA-based methods yielded better results for all the classes. This means the NAPCA-generated feature space is optimal in terms of all the classes but may not for a specific class.

Table 4.3 Detection and classification accuracy (%) of 0.25 s data

Methods	TD	NTD	OD	TC	OC
GC	83.3	55.6	75.0	73.8	60.0
SBE	85.7	75.0	82.5	71.4	67.5
SBR	88.1	11.1	65.0	47.6	33.3
SCR	82.1	50.0	72.5	57.1	47.5
SBE-PCA	90.5	83.3	88.3	76.2	66.7
SBR-PCA	78.6	50.0	70.0	45.2	41.7
SCR-PCA	78.6	72.2	76.7	50.0	48.3
SBE-NAPCA	92.9	88.9	91.7	78.6	70.0
SBR-NAPCA	92.9	88.9	91.7	78.6	70.0
SCR-NAPCA	85.7	77.8	83.3	76.2	73.3

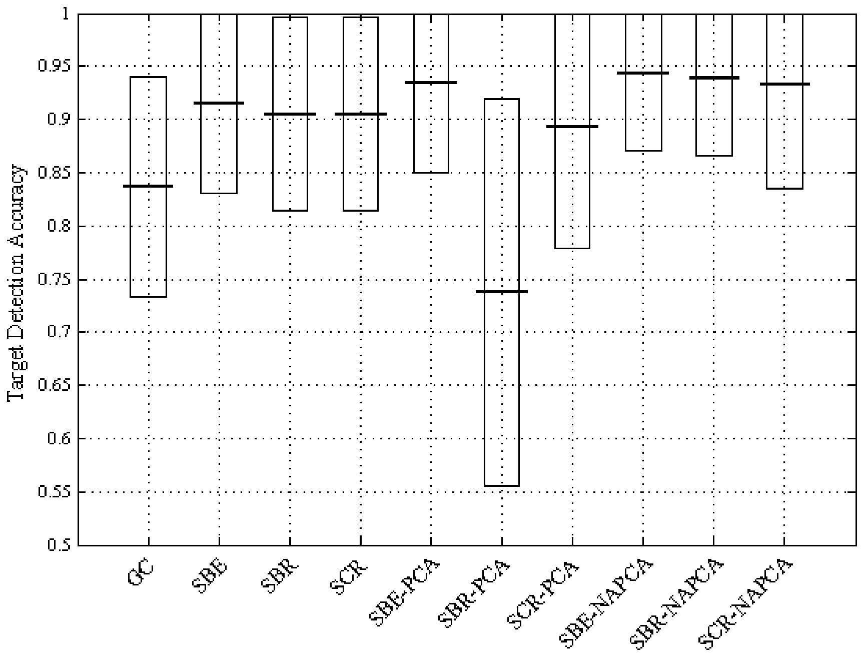
The accuracy for 0.25 s and 0.1 s data were presented in Table 4.3 and 4.4, respectively. For 0.25 s data, two NAPCA-based methods could provide 90% overall detection accuracy and 70% overall classification accuracy. For 0.1 s data, using NAPCA, the overall detection accuracy could be above 80%; however, the target classification and the overall classification accuracy were only around 60% and 50%, respectively, due to low energy counts when sensor dwell time was as short as 0.1 s. In this case, the GC method seemed to be quite stable.

Table 4.4 Detection and classification accuracy (%) of 0.1 s data

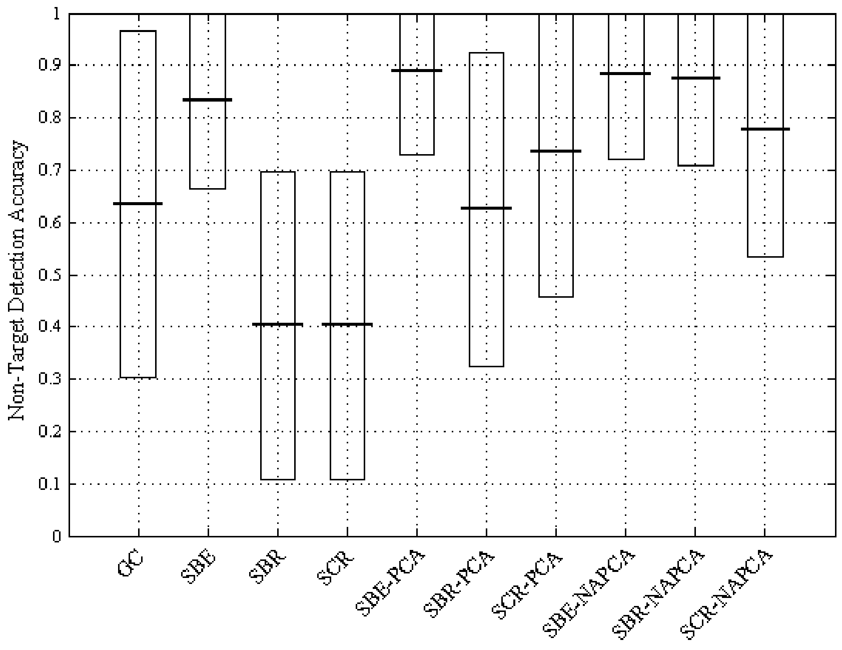
Methods	TD	NTD	OD	TC	OC
GC	78.6	66.7	75.0	71.4	50.0
SBE	81.0	44.4	70.0	54.8	50.0
SBR	92.9	22.2	71.7	31.0	26.7
SCR	88.1	22.2	68.3	40.5	33.3
SBE-PCA	81.0	50.0	71.7	47.6	45.0
SBR-PCA	71.4	44.4	63.3	35.7	33.3
SCR-PCA	81.0	44.4	70.0	57.1	51.7
SBE-NAPCA	85.7	86.7	86.0	62.9	50.0
SBR-NAPCA	85.7	86.7	86.0	62.9	50.0
SCR-NAPCA	82.9	80.0	82.0	60.0	50.0

4.3.2 Uncertainty Analysis

Tables 4.5-4.7 show the average accuracy in the 24-fold cross-validations. In order to better describe the accuracy statistics when using different training and test samples, boxplots were generated in Fig. 4.7 showing the mean and standard deviation for each method (corresponding to Table 4.5). From Fig. 4.7(a), we can see that GC and SBR-PCA were worse than other six methods; SBE-NAPCA and SBR-NAPCA were the best because the mean accuracy was among the largest and the standard deviation was the smallest. Similarly, in Fig. 4.7(b)-(e), SBE-NAPCA, SBR-NAPCA, and SCR-NAPCA were better than their counterparts, and ranked among the best methods.

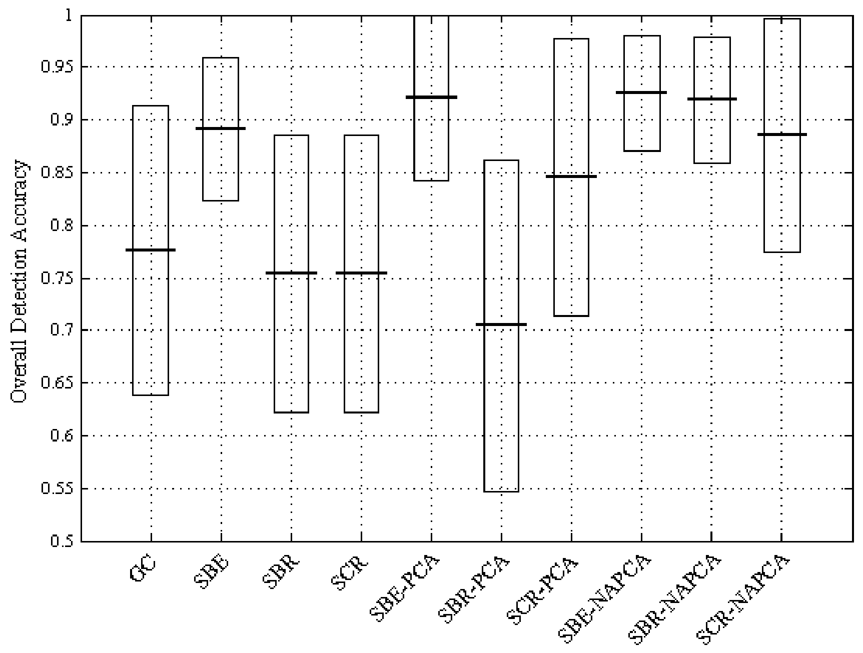


(a) Target Detection (TD) Accuracy

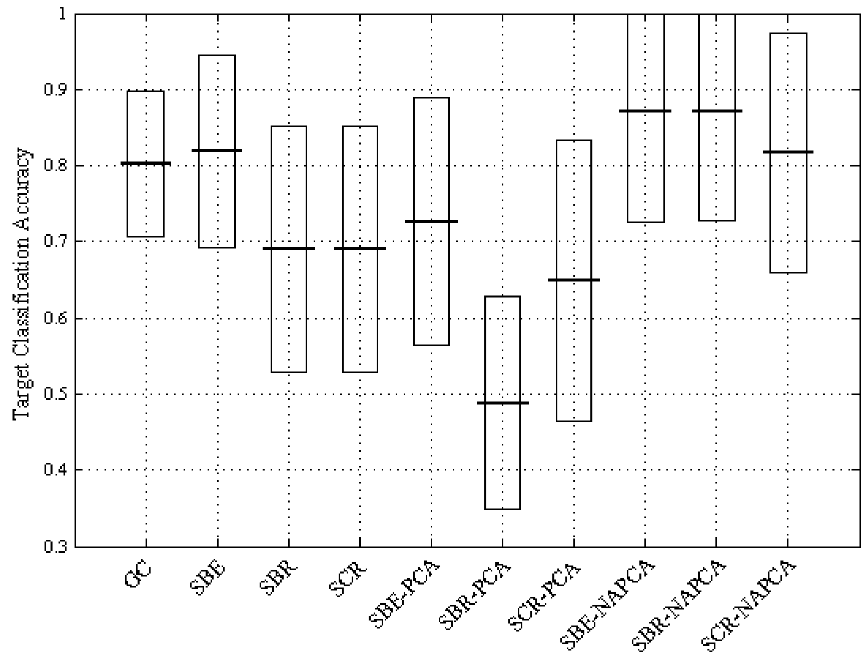


(b) Non-target Detection (NTD) Accuracy

Figure 4.7 Boxplots for 24-fold cross-validation using the entire dataset.

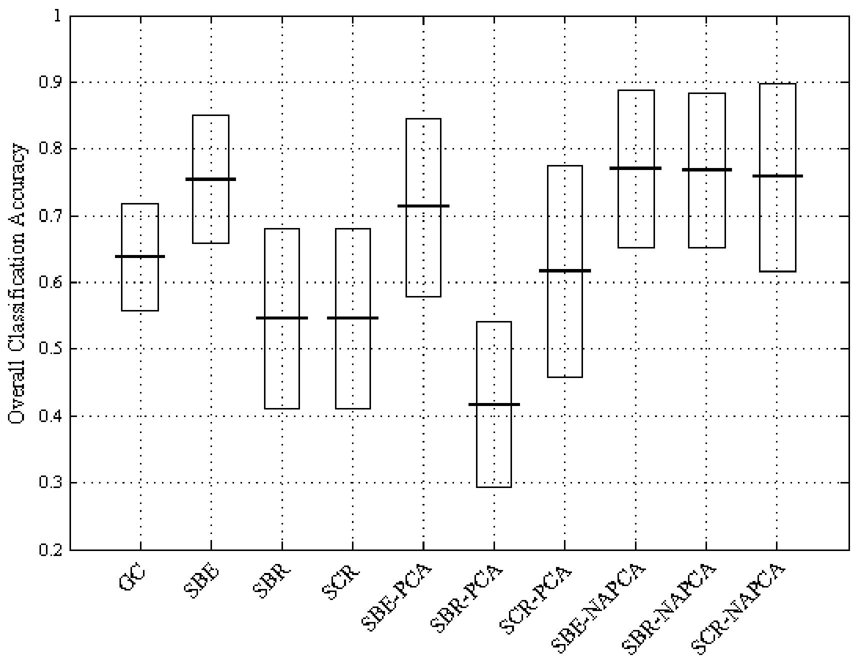


(c) Overall Detection (OD) Accuracy



(d) Classification Accuracy of Seven Target Classes (TC)

Figure 4.7 (continued)



(e) Overall Classification Accuracy of Ten Classes (OC)

Figure 4.7 (continued)

The ANOVA (analysis of variance) F-test was employed to quantify the statistically significant difference between the mean accuracies of the ten methods (denoted as μ_i, $i = 1, \ldots, 10$) with the hypothesis test being formulated as

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_{10}$$

$$H_1 : \textit{not all the } \mu_i \textit{ are equal} \tag{4.19}$$

The results are shown in Table 4.5 with significance level being set to be $\alpha = 0.05$ as usual. We can see that all the P values are less than 0.0001, much smaller than $\alpha = 0.05$, indicating that H_0 is rejected. This means there really exist significant differences among the mean accuracies of the ten methods. Based on the F values, Table 4.5 also indicates that performance discrepancy is the most obvious for OC and the least obvious for TD, which is the same as illustrated in Fig. 4.7. Here, error degree of freedom is 230, treatment degree of freedom is 9, and total degree of freedom is 239.

Table 4.5 F-test for the mean accuracies of the ten methods

	TD	NTD	OD	TC	OC
F Value	8.8835	13.3817	13.1613	15.2013	22.3821
P Value	<0.0001	<0.0001	<0.0001	<0.0001	<0.0001

T-test was also used to analyze the significance of performance discrepancy between different groups: group1 (G1) {GC}; group2 (G2): {SBE, SBR, SCR}, group3 (G3): {SBE-PCA, SBR-PCA, SCR-PCA}, group4 (G4): {SBE-NAPCA, SBR-NAPCA, SCR-NAPCA}. The G_i-G_j test is

$$H_0 : m_i = m_j$$

$$H_1 : m_i \neq m_j \tag{4.20}$$

where m_i is the mean accuracy of G_i (i = 1, 2, 3, 4). The significance level being set to be $\alpha = 0.05$ as usual. Small samples inferences for two samples are considered. The degree of freedom equals the sum of populations of two samples minus 2. If the P value is smaller than $\alpha = 0.05$, H_0 is rejected which means there exists material difference between the performance of the two groups under test. The t-test results were shown in Table 4.6 where the P-values less than $\alpha = 0.05$ were highlighted. As we can see, in all the tests related to the NAPCA group G_4, the significance was very obvious. For instance, the test between G_2 and G_4 showed that all the accuracies except TC demonstrated great improvement, which means applying NAPCA was better than the original ones; the test between G_3 and G_4 also showed great improvement except NTD, which means NAPCA was a better choice than PCA.

Table 4.6 T-test for the mean accuracies of the four groups

		TD	NTD	OD	TC	OC
G1-G2	T Value	2.4381	0.0340	1.2705	1.1516	0.7784
	P Value	0.0187	0.9730	0.2103	0.2554	0.4403
G1-G3	T Value	0.4810	1.3129	1.1242	4.2020	1.3643
	P Value	0.6328	0.1957	0.2668	0.0001	0.1791
G1-G4	T Value	3.7721	2.6538	4.1059	1.3817	4.2150
	P Value	0.0005	0.0109	0.0002	0.1737	0.0001
G2-G3	T Value	1.3408	1.3167	0.0167	2.7425	1.7053
	P Value	0.1866	0.1945	0.9868	0.0087	0.0949
G2-G4	T Value	1.3974	2.7160	2.8236	2.1044	2.4557
	P Value	0.1690	0.0093	0.0070	0.0408	0.0179
G3-G4	T Value	2.3041	1.3689	2.4237	4.7176	4.0286
	P Value	0.0258	0.1777	0.0194	0.0001	0.0002

4.3.3 Experiment Using Data Containing Difficult Classes Only

This experiment was conducted when the three easy classes: DU buried 15 cm, 23 cm, and 30 cm deep, were removed. Table 4.6 lists the average detection accuracy of cross-validation, where NAPCA could improve the performance of SBE, SBR, and SCR while PCA did not necessarily bring about improvement. Table 4.7 also lists the average classification accuracy for the seven classes (with four target classes), where NAPCA-based methods were among the best.

Table 4.7 Detection and classification accuracy (%) of different methods for the dataset containing seven difficult classes

Methods	TD	NTD	OD	TC	OC
GC	75.0	66.7	71.4	69.8	51.2
SBE	85.9	84.1	85.1	85.9	72.8
SBR	83.3	41.7	65.5	63.5	45.8
SCR	81.3	72.2	77.4	72.9	66.1
SBE-PCA	87.5	81.9	85.1	85.4	73.2
SBR-PCA	56.3	63.9	59.5	22.9	23.2
SCR-PCA	83.3	77.8	81.0	64.6	61.9
SBE-NAPCA	89.1	84.1	87.0	87.0	73.9
SBR-NAPCA	86.5	83.3	85.1	84.4	71.4
SCR-NAPCA	85.4	80.6	83.3	83.3	67.9

4.4 Conclusion

In this chapter, we propose an approach for buried depleted uranium detection and classification, which applies spectral transformation followed by PCA or NAPCA. To meet the requirement of practical survey mapping, we focus on the circumstance when sensor dwell time is very short (i.e., less than 1 s). In this case, the gamma spectroscopy collected by an NaI detector can be sparse, random, and dominated by energy counts from the background. We believe an appropriate spectral transform can alleviate the effects from spectral noisy variation and background clutters, while NAPCA, a better choice than PCA, can extract major features for the following detection and classification. Thus, it can generally improve the target detection and classification performance after a certain spectral transform is applied.

For SBR and SCR, a known background measurement is needed. For real field data, background may be changed with geolocation. Under such circumstance, SBR and SCR should be applied locally by using a local background measurement. Even for the data collected at the same location (e.g., the lab data), the background measurement is changed with time due to counting uncertainty when sensor dwell time is very short. This is why a background measurement normalized by another background measurement using SBR or SCR is not a constant 1 or 0. Such variation shows the importance of employing a statistical approach (e.g., PCA and NAPCA) instead of the traditional deterministic approach. It also motivates us to employ a multi-dimensional approach, where all the energy channels/bins are explored simultaneously for decision-making rather than a single or a few channels only.

It is worth mentioning that the performance of all the three spectral transforms is varied with the bin-width selection. How to automatically select an optimal bin-width for each transform is under investigation. However, using the NAPCA-transformed data, these three spectral transforms generally can provide better detection and classification than other methods, such as GC, applied on the original data.

CHAPTER V

OPTIMIZED SPECTRAL TRANSFORMATION

5.1 Introduction

In this chapter, we will apply Particle swarm optimization (PSO) to search for the optimal number of bins and bin-widths in spectral transformation. PSO is an evolutionary computation technique proposed and developed by Kennedy and Eberhart [27]-[31].The PSO uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for global optimal solutions. It is proved to be a very efficient optimization algorithm by searching the entire problem space.

Because PSO is easy to implement, it has been widely used in many engineering optimization problems and proved to be a very efficient optimization algorithm [32]-[37]. Huang *et al.* [38] proposed a method that embeds the sequential search into the evolution optimization of PSO and genetic algorithm (GA) for better ability of the fine tune in local search space and thus behaves well in both global and local situations. Monteiro *et al.* [39] proposed a method for feature selection algorithm based on particle swarm optimization for processing remotely acquired hyperspectral data. Their method utilizing two swarms of particles in order to optimize simultaneously a desired performance criterion and the number of selected features. Gao *et al.* [40] presented a new method for hyperspectral image classification. It combines support vector machine (SVM), PSO and GA together. Its aim is to improve the classification accuracy and reduce the computation consumption based on heuristic algorithms.

Besides the standard PSO, theoretical studies and performance improvements of the algorithm have been merged. The inertia weight ω was introduced by Shi and Eberhart [27]. They proposed an ω linearly decreasing with iterations. Fuzzy adaptive ω was proposed in [41], and a random version setting ω to $0.5 + \mathit{rand}(.)/2$ was experimented in [42] for dynamic system optimization, where rand(.) denotes a uniform random variable within [0, 1]. In addition to the inertia weight and the constriction factor, the acceleration coefficients c_1 and c_2 are also important role in PSO. Kennedy and Eberhart [27] suggested a fixed value of 2.0 while Clerc [59] suggests using the constant c_1 and c_2 which equals to 1.49445. Ratnaweera *et al.* [58] proposed a PSO algorithm with linearly time-varying acceleration coefficients (HPSO-TVAC). The idea of this algorithm is setting a larger c_1 and a smaller c_2 at the beginning and they were gradually reversed during the search.

Another active research trend in PSO is hybrid PSO, which combines PSO with other evolutionary paradigms. Angeline [42] first introduced into PSO a selection operation similar to that in a GA. The method of hybridization of GA and PSO has been used in for recurrent artificial neural network design [43]. In addition to the normal GA operators, e.g., selection [42], crossover [45], and mutation [46], other techniques such as local search [47] and even differential evolution [48] method have been employed to combine with PSO. Cooperative approach, self-organizing hierarchical technique, deflection, stretching, and repulsion techniques [49] have also studied to combine with traditional PSO to enhance performance. In the area of biology, some researchers introduced niche [50] and speciation [51] techniques into PSO to prevent the swarm from crowding too closely and to explore more optimal solutions compared without this technique.

In addition to study on parameter selection, PSO topological structures are also widely researched on. The LPSO with a ring topological structure and the von Neumann topological structure PSO (VPSO) have been proposed by Kennedy and Mendes [52],[53] to enhance the system's performance in solving multimodal problems. Further, dynamically changing neighborhood structures have been proposed by Hu and Eberhart [54] and Liang and Suganthan [55] to overcome the deficiency of unchanged neighbor range. Moreover, in the "fully informed particle swarm" (FIPS) algorithm [56], the information of the entire neighborhood is used to guide the particles. The CLPSO [57] lets the particle use different *pBest*'s to update steps of particles for improved performance in multimodal applications.

5.2 Proposed Method

5.2.1 Unconstrained PSO

The update of particles is accomplished by the equation (5.1) which calculates the new velocity for each particles based on the previous velocity (V_{id}), the particle's location (p_{id} or *pBest*) which is best for the objective function and the particle's location among the neighborhood (p_{ld} or *pBest*) or globally (p_{gd} or *gBest*) that is best for the objective function. These particles are all potential solutions and therefore equation (5.2) is used to update their locations in the solution space. There are two random numbers c_1 and c_2 are independently generated. And the inertia weight ω is used as the scalar of previous velocity V_{id} which provides improved performance in various applications.

$$V_{id} = \omega \times V_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times rand() \times (p_{gd} - x_{id}) \tag{5.1}$$

$$x_{id} = x_{id} + V_{id} \tag{5.2}$$

The detailed PSO algorithm is as follows:

1. Generate a population of particles with random positions and velocities in the problem space.
2. Evaluate each particle for the optimization fitness function.
3. Compare each of the fitness function values with particle's *pBest* value. If its fitness function value is better than *pBest*, then repalce the current particle fitness function value by the current fitness value. And set the *pBest* location by the current particle's location.
4. Compare fitness values of all the *pBest* in the population with the previous *gBest* value. If the current *gBest* is better than the previous *gBest* value, then replace the *gBest* value with the current *gBest* value and their locations.
5. Change the velocity and position of the particle according to equation (5.1) and (5.2) separately.
6. Loop to step 2 until certain criterion is met, usually a sufficiently good fitness value or a maximum number of iterations has finished.

Here the particle's velocity on each dimension has been restricted by a maximum velocity $V_{\max}$. If the velocity on any dimension has exceed the maximum velocity which is user defined parameter, then that dimension's velocity is set to $V_{\max}$ or $-V_{\max}$. On the view of the role of $V_{\max}$, larger value give the ability of particles to search globally while smaller value make the particles search in neighborhood. Here we use the dynamic range as the $V_{\max}$.

The learning constants, c_1 and c_2 in equation (5.1), scale the weight of stochastic acceleration of each particle towards the *pBest* and *gBest*. Therefore, the setting of these acceleration constants controls the steps of particles to the *pBest* and *gBest*. Large values may lead the particles move past the target area, while small values may make the particles wander around the target area before it reached to. Early experiments have set the constants c_1 and c_2 to 2.0 for almost all applications. But later research have been done showing the proper choice of constraint factor may affect the convergence of particle swarm problem. Clerc [59] suggests using the constant c_1 and c_2 which equals to 1.49445.

The population size selection is another problem but not much important. And usually population size is between 20 to 50. It was realized that smaller population is common for PSO since the smaller burden that population brings for computation.

The use of inertia weight in equations (5.1) has provided an improved role in most applications. In original paper [27], the inertia weights decrease linearly from 0.9 to 0.4 during all the iterations. And it effects the exploration of particles globally or locally. Larger value also brings the quick convergence for fitness function on average. A slight change made for inertia weight for fitting the need of tracking and optimizing the dynamic systems [42]. The weight ω in equation (5.1) was set to [rand(.)/2.0 + 0.5]. It ranges between 0.5 and 1.0 but with a mean value of 0.75.

5.2.2 Constrained PSO

The original PSO focuses on unconstrained optimization problems. Various constrained PSO algorithms were developed to facilitate constrained optimization processes [60]. In order to handle constrained problems, the original PSO needs to be modified. The most straightforward modification is to keep the feasible solutions from multiple solutions, which is the one adopted in this research. Thus, all the particles fly through the entire problem space, but only the feasible solutions are tracked and updated in PSO.

In order to facilitate this process, all the particles need to be initialized by feasible solutions. The detailed steps are shown below. Two main modifications have been made to the original PSO. First change is in the initialization process, particles are randomly produced until all the particle populations meet the constraints. Second change is in the update of *pBest*; only those particles in the feasible space are used to update *pBest*.

1. Generate a population of particles with random positions and velocities in the feasible space
2. Evaluate each particle for the optimization fitness function.
3. Compare each of the fitness function values with particle's *pBest* value. If its fitness function value is better than *pBest* and if this particle is in the feasible space, then repalce the current particle fitness function value by the current fitness value. And set the *pBest* location by the current particle's location.
4. Compare fitness values of all the *pBest* in the population with the previous *gBest* value. If the current *gBest* is better than the previous *gBest* value, then replace the *gBest* value with the current *gBest* value and their locations.
5. Change the velocity and position of the particle according to equation (5.1) and (5.2) separately.
6. Loop to step 2) until certain criterion is met, usually a sufficiently good fitness value or a maximum number of iterations has finished.

This constrained PSO costs a lot of time in the initialization step because the feasible space is small when data dimensionality is large. Many random generations must be created before the constraints are relaxed for all the initial particles.

5.2.3 A simple example

In the process of PSO, a large population of particles moves across the solution space. Their update is based on the *pBest* and *gBest*. To illustrate swarm movement, an unconstrained PSO was implemented on a sphere function:

$$f(x) = (x_1 - r)^2 + (x_2 - r)^2 \tag{5.3}$$

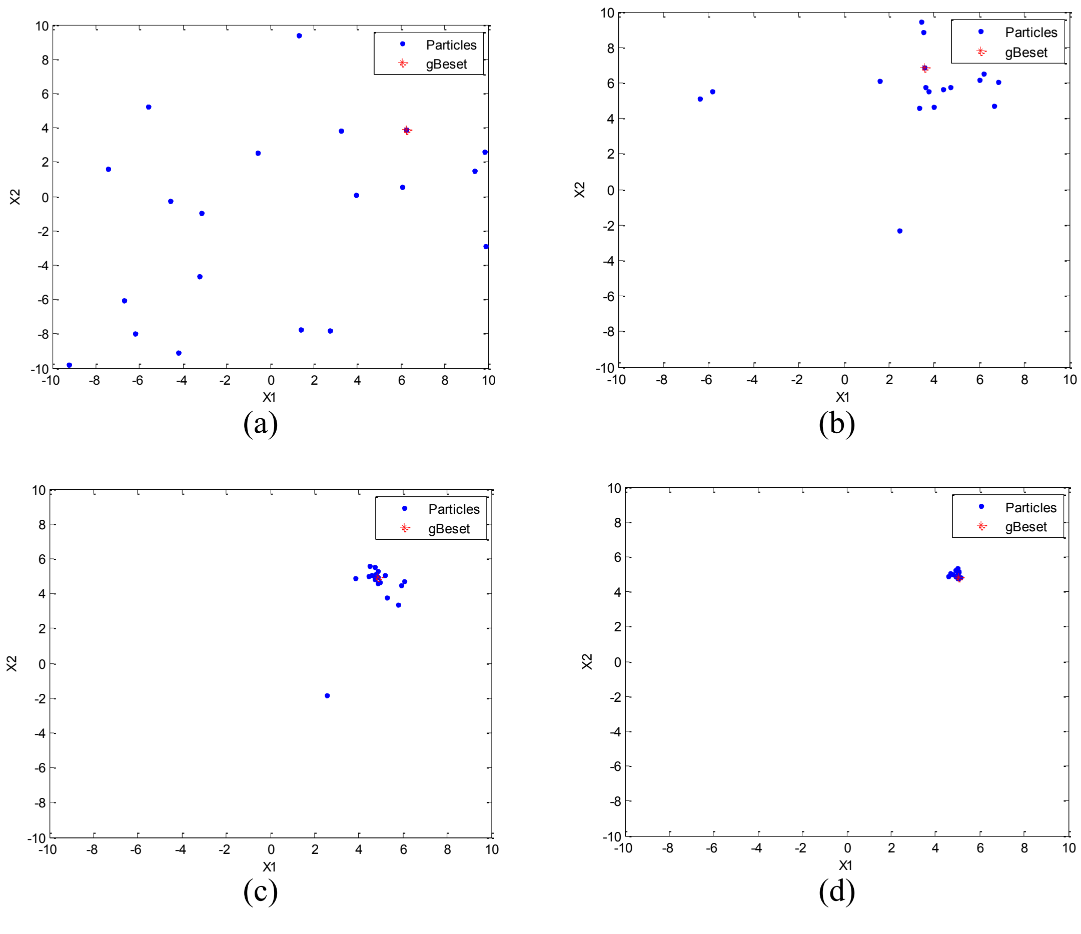


Figure 5.1 Population distribution at different iterations. (a) Iteration = 1. (b) Iteration = 25. (c) Iteration = 50. (d) Iteration = 75.

We set the value of r to be 5 in equation (5.3), so the optimum solution for the objective function is $[x_1, x_2] = [5, 5]$. 20 particles were randomly initialized and 100 iterations were executed. The population distributions in various generations were observed, as shown in Fig. 5.1. From the four subfigures, we could see that the initial particles were spread sparsely in the whole problem space at iteration 1. Then the particles started to be pulled by the updating procedure to the optimal fitness value regions from iteration 25 to iteration 75. Finally, all the particles were gathered at the optimum point.

5.2.4 Powell's direction set method

The Powell's method [86] is an optimization method that is suitable to find the optimum solution that is not far away from the starting searching point when the derivative is difficult to compute. It minimizes along one direction after another. In each iteration, a one-dimensional minimization method (e.g., the Brent's method) is employed. The goal is to minimize the objective function through a set of linear, dependent, and conjugate directions. However, this optimization method is easily to converge to local minima, particularly when the objective function is multi-dimensional. In practical applications, a number of initial values can be tried to find a better solution. However, it is still difficult to find a global optimal solution.

5.2.5 Uniform bin-width optimization

As mentioned earlier, each channel is partitioned into one and only one bin. Thus, we first consider the simplest situation where all the bins in the four transforms in Chapter IV have uniform widths. Then, our goal is to optimize the number of bins that can offer the optimal fitness value for the data. We can use the exhaustive method to search all the possible number of bins and select the best results with the corresponding bin-width. This approach is mainly for comparison purpose.

5.2.6 PSO searching for variable bin-widths

Different from the Powell's method, PSO has much better global optimum searching ability. This is because the PSO algorithm can perform optimization in a multi-dimensional searching space. PSO searches the solution space by starting from randomly distributed *particles* like swarm. Here, possible solutions are called *particles*, and recursive solution update is called *velocity*. It is very similar to other evolutionary computation algorithms, but has relatively fast convergence. It shares some characteristics with evolutionary techniques: 1) it uses a large size of random particles as initials; 2) the optimum objective function value is determined by iteratively updating the generations; and 3) evolution adaptation uses the previous generations, and particles are flown through the problem space following the current best solution.

Assume J bins are to be searched and M particles are randomly initialized. Let a particle x_{id} of size $(J-1)\times 1$ include channel indices $\{c'_1, c'_2, \cdots, c'_{J-1}\}$ selected as bin boundaries, and let v_{id} be the update for selected channel indices. In addition, the historically best local solution for a particle is p_{id}, and the historically best global solution among all the M particles is p_{gd}. The detailed recursion can be expressed by Eqs. (5.3) and (5.4). It calculates the new update for each particle based on the previous update v_{id}, p_{id} that it has reached so far so best for the objective function, and p_{gd} that has reached so far so best for the objective function. In Eq. (5.3), two random numbers ρ_1 and ρ_2 are independently generated; r_1 and r_2 represent two random numbers, which controls the contributions from local and global best solutions; and the inertia weight w is used as the scalar of previous velocity v_{id} which provides improved convergence performance in various applications [8].

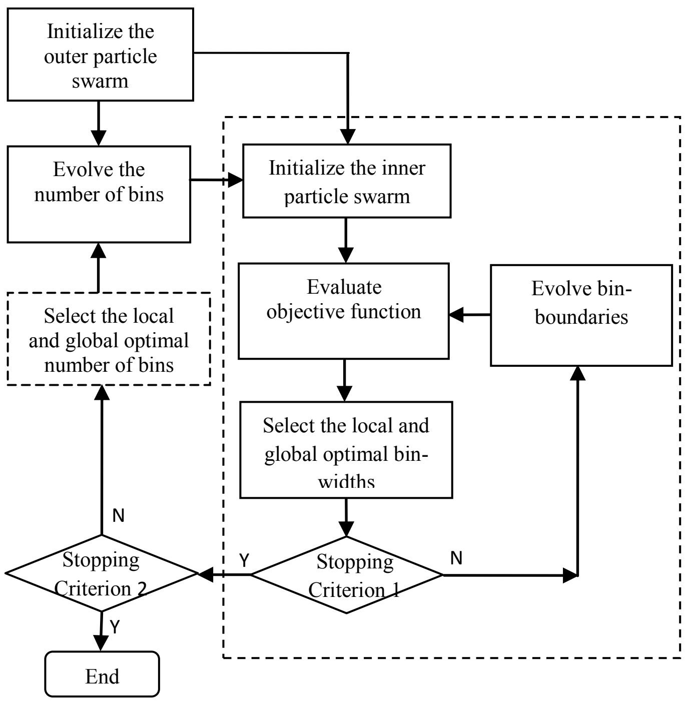

Initialize the outer particle swarm
Evolve the number of bins
Select the local and global optimal number of bins
Initialize the inner particle swarm
Evaluate objective function
Evolve bin-boundaries
Select the local and global optimal bin-widths
Stopping Criterion 1
Stopping Criterion 2
N
Y
N
Y
End

Figure 5.2 The proposed adaptive optimization system.

$$\mathbf{v}_{id} = w \times \mathbf{v}_{id} + \rho_1 \times r_1 \times (\mathbf{p}_{id} - \mathbf{x}_{id}) + \rho_2 \times r_2 \times (\mathbf{p}_{gd} - \mathbf{x}_{id}) \tag{5.4}$$

$$\mathbf{x}_{id} = \mathbf{x}_{id} + \mathbf{v}_{id} \tag{5.5}$$

After convergence, it is needed to round off the solutions to adapt the continuous PSO to a discrete form because the channel indices are discrete values.

5.2.7 PSO searching for the optimal number of bins

If the optimal number of bins is known, we can search for variable bin-widths using the technique described above. Unfortunately, it is difficult to decide the optimal number of bins for a given dataset. Here, we propose an adaptive system to searching for the optimal number of bins and varied bin-widths simultaneously. As shown in Fig. 6, two different optimization processes are incorporated in this system. The inner PSO (in the dashed-line box) searches the optimal varied bin-widths with a certain number of bins, which is one of the particles in the outer PSO (the loop on the left side of Fig. 6). After the inner PSO is converged, the outer PSO updates its particles based on the solutions corresponding to bin-widths produced by the inner one. Once the particles in the outer PSO are updated, the entire process of the inner PSO is executed again. The stopping criterion can be chosen as: the change of the best solution (i.e., g_{id}) between two consecutive iterations is less than a threshold, or g_{id} does not change after a certain number of iterations. Here, we terminate the iterations for the inner PSO if g_{id} does not change after 200 iterations, 100 iterations for the outer PSO. Particles in the inner PSO are multi-dimensional vectors, requiring more iterations for convergence.

Note that the same PSO algorithm in Section III. D is used to search for the optimal number of bins, except that the vectors in Eqs. (5.4) and (5.5) are scalars now; specifically, x_{id} represents a possible solution for the number of bins J, and v_{id} is the update for the value of J in each iteration.

5.2.8 Overall algorithm with two PSOs

To summarize, the automatic searching algorithm using two PSOs can be described as follows.

1) Randomly initial M particles, say, M =20, for the outer PSO, denoted as $\left\{x_{id}^{outer,k}\right\}_{k=1}^{M}$, respectively. Set k =0.

2) Let k = k+ 1, for the specific number of bins represented by $x_{id}^{outer,k}$, randomly initial M particles for the inner PSO denoted as $\left\{\mathbf{x}_{id}^{inner,j}\right\}_{j=1}^{M}$, and each includes ($x_{id}^{outer,k}-1$)

boundary channel indices for partition. In each iteration, run the following steps for the inner PSO update.

2.1) For each inner particle, evaluate the selected objective function.

2.2) Determine the global best particle p_{gd}.

2.3) For each particle, determine its historically local best solution p_{id}.

2.4) Use Eqs. (10) and (11) to update all the inner particles.

2.5) Repeat steps 2.1)-2.4) until convergence. Keep the p_{gd} as the partition corresponding to $x_{id}^{outer,k}$.

3) If $k < M$, go to Step 2). If $k = M$, check if the outer PSO is converged. If yes, terminate the algorithm; the p_{gd} is the optimal number of bins and its corresponding partition found by the inner PSO is the optimal bin-widths. Otherwise, run the following steps for the outer PSO.

3.1) For each outer particle, retrieve the objective function based on the corresponding bin partitions found by the inner PSO.

3.2) Determine the global best particle p_{gd}.

3.3) For each outer particle, determine its historically local best solution p_{id}.

3.4) Use Eqs. (10)-(11) to update all the outer particles.

3.5) Set k =0 and go to Step 2).

5.3 Experiments

5.3.1 Data and implementation

Laboratory data was collected using a 10×10×40 cm sodium iodide (NaI) scintillation detector. The measured spectra covered the energy range from 0 keV to 2160.0 keV with 1011 channels. The target has a cylindrical shape with 4.3 kg mass. The background consisted of construction sand. Natural ore was considered a benign material, scattered in a liter-size plastic bag. The distance between the detector and sand surface was about 15cm. The detector was above the center of the target and ore, which were parallel to the detector cart.

The target was buried at 15 cm, 23 cm, 30 cm, 45 cm, 60 cm, 75 cm, and 90 cm. Natural ore was buried at 45 cm and 75 cm depth. For each class, 24 samples were taken evenly by four different dwell times: 1 s, 0.5 s, 0.25 s, to 0.1 s. In the experiment, all the measurements were normalized into equivalent 1 s dwell time.

Table 5.1 The performance of GC and original k-NN on the two datasets

	Training Data		Testing Data	
	OD	OC	OD	OC
GC	0.833	0.608	0.667	0.550
Original	0.875	0.792	0.850	0.767

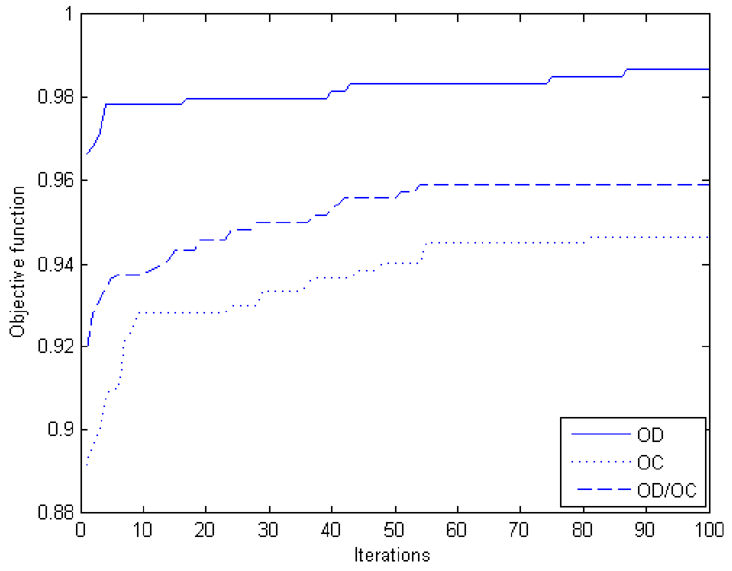
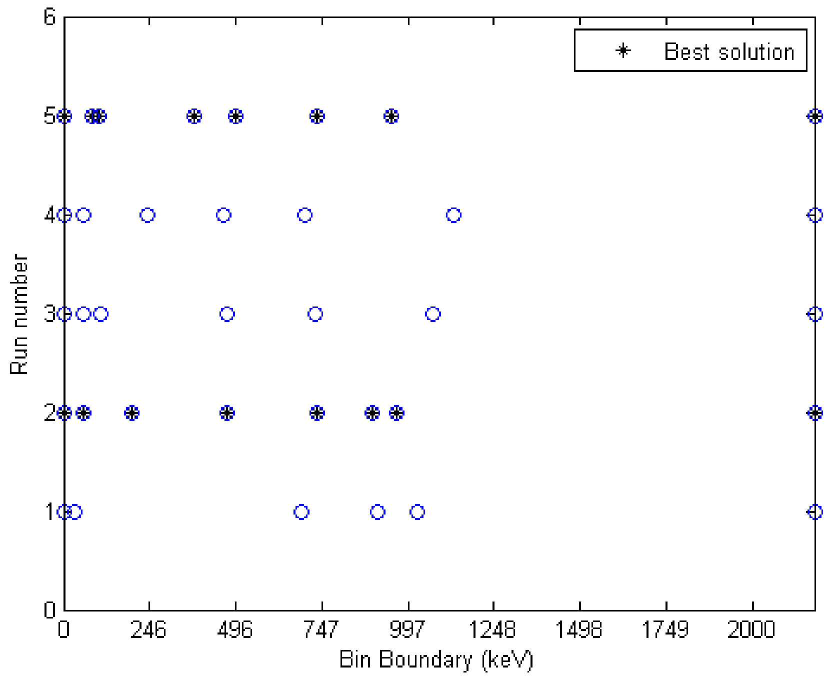
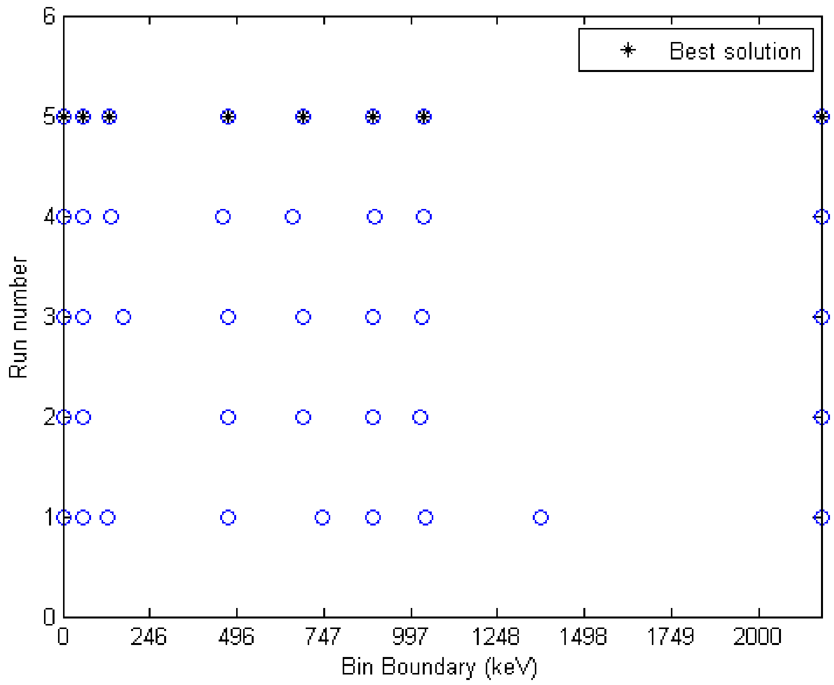


Figure 5.3 A learning curve for the SCR transform.

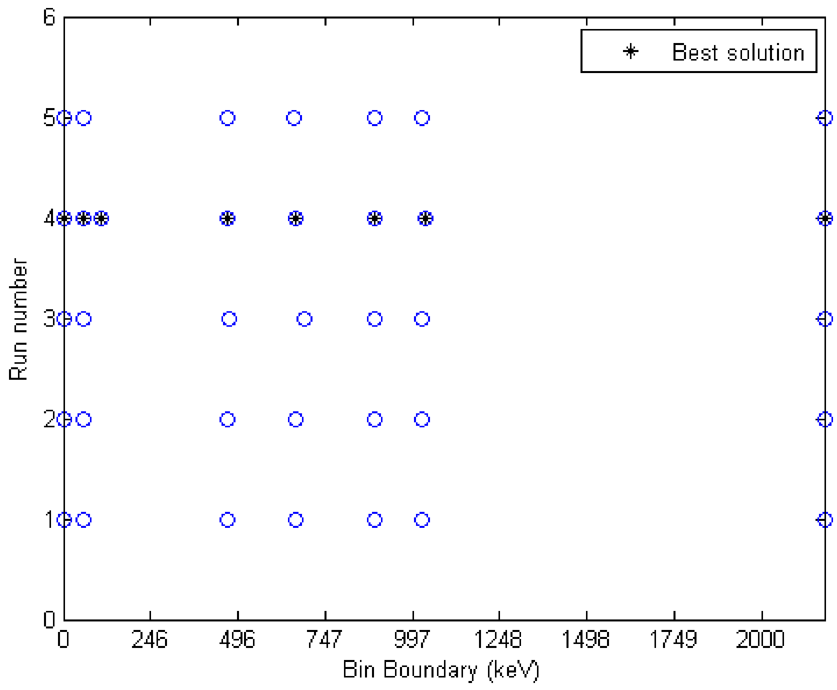


(a) SCR (OD)



(b) SCR (OC)

Figure 5.4 Bin boundaries from five PSO runs for SCR.



(C) SCR (0.5OD + 0.5OC)

Figure 5.4 (continued)

The lab data were divided into two parts with equal size. Each part contained 120 samples. Within the 120 samples, each class had 12 samples; among the 12 samples, each dwell time had 3 samples. We treated the first part of the data as training data and used for system training purpose, and the second part were for testing. 1-NN with 3-fold cross validation was applied for detection and classification in both training and testing process. OD was calculated when all the seven target classes were treated as a single class and natural ore and background as the other, while OC was computed when the ten classes were considered as individual classes.

5.3.2 GC and *k*-NN using the original data

For comparison purpose, Table 5.1 shows the performance of GC and *k*-NN using the original data without spectral transformation on the training data and testing data. On the testing data, GC yielded OD = 0.667 and OC = 0.550, while *k*-NN provided OD = 0.850 and OC = 0.767, better than GC.

5.3.3 Uniform bin-width and Powell's method

For bin partition, first we considered the simplest situation where all the bins in the aforementioned transforms had uniform bin-widths. If the bin-width is the same for all the bins, then exhaustive search is doable to select the optimal number of bins. The optimal number of bins was searched using the training data, then applied for the testing data. Similarly, Powell's method and PSO system were trained with the training data, then tested on the testing data. Their results with the three objects are tabulated in Tables 5.2, 5.3 and 5.4.

5.3.4 Bin optimization using PSO

For the PSO method, an example of learning curve is plotted in Fig. 5.3, where we can see the increase of the objective function with the number of iterations. The bin partition results (i.e., bin boundaries) from five runs for SCR are illustrated in Fig. 5.4. Each run may yield different partitions, and the one marked in ○ is the one providing the best objective function among the five. In Fig. 5.4(a), two partitions provided the same OD values for the training data. Due to the problem complexity, there may exist many local and global optima. Partitions are changed with the objective function; and they are also changed with transforms (and classifier employed).

Table 5.2 The resulting performance when the objective is overall detection accuracy

	Training Data		Testing Data	
	OD	OC	OD	OC
Uniform SCR	0.892	0.792	0.833	0.742
Uniform SBE	0.942	0.875	0.858	0.800
Uniform SBD	0.942	0.867	0.875	0.792
Uniform SBR	0.958	0.833	0.912	0.825
Varied SCR (Powell)	0.858	0.732	0.827	0.737
Varied SBE (Powell)	0.938	0.852	0.875	0.810
Varied SBD (Powell)	0.932	0.847	0.872	0.787
Varied SBR (Powell)	0.915	0.775	0.870	0.768
Varied SCR (PSO)	0.986	0.863	0.880	0.780
Varied SBE (PSO)	0.997	0.887	0.902	0.828
Varied SBD (PSO)	0.993	0.893	0.897	0.828
Varied SBR (PSO)	1.000	0.908	0.950	0.841

The OD and OC results are shown in Tables 5.2, 5.3, and 5.4. Since five repeated runs were implemented, the mean values were presented in the tables. Table 5.2 summarized OD when OD was the objective. For the four spectral transforms, the uniform bin-width provided moderate OD because it fixed all bin-widths to the same and could not divide the spectrum adaptively based on energy peaks or features. However, the varied bin-width optimization would adjust the optimal number of bins and their corresponding widths so that an energy window could adaptively capture the interest energy peaks and combine them together. As the consequence, OD was improved. Comparing the results of Powell's method with that of PSO, the four spectral transforms had all increased their OD. SBR provided the best result with 100% OD for the training corresponding bin data and 95% for the testing data. OC was derived with the same partitions. Compared to GC and the case using the original spectra in Table 5.1, bin partitioning did improve the overall detection and classification performance.

Table 5.3 The resulting performance when the objective is overall classification accuracy

	Training Data		Testing Data	
	OD	OC	OD	OC
Uniform SCR	0.858	0.800	0.800	0.725
Uniform SBE	0.930	0.875	0.890	0.825
Uniform SBD	0.930	0.875	0.890	0.825
Uniform SBR	0.942	0.850	0.925	0.825
Varied SCR (Powell)	0.848	0.750	0.820	0.733
Varied SBE (Powell)	0.923	0.847	0.898	0.841
Varied SBD (Powell)	0.927	0.850	0.893	0.828
Varied SBR (Powell)	0.920	0.803	0.907	0.797
Varied SCR (PSO)	0.978	0.947	0.890	0.793
Varied SBE (PSO)	0.971	0.963	0.900	0.842
Varied SBD (PSO)	0.975	0.963	0.900	0.850
Varied SBR (PSO)	0.998	0.991	0.937	0.848

Table 5.3 summarized OC when OC was the objective. The varied bin-width PSO still provided a higher accuracy than the exhaustively searched uniform bin-width. Again, OD was derived with the corresponding bin parameters. Comparing Table 5.2 and 5.3, we can see that both our optimization methods enhanced the desired accuracy function because it was our criterion function in the optimization process. This is why the OD in Table 5.2 is larger than the counterpart in Table 5.3, and OC in Table 5.3 is larger than

that in Table 5.2. PSO still could achieve better results comparing with the Powell's method and the uniform partition.

Table 5.4 The resulting performance when the objective is multi-objective function

	Training Data			Testing Data		
	OD	OC	0.5OD + 0.5OC	OD	OC	0.5OD + 0.5OC
Uniform SCR	0.892	0.792	0.842	0.833	0.742	0.788
Uniform SBE	0.942	0.875	0.908	0.858	0.800	0.829
Uniform SBD	0.942	0.875	0.908	0.858	0.800	0.829
Uniform SBR	0.958	0.833	0.896	0.917	0.825	0.871
Varied SCR (Powell)	0.872	0.773	0.822	0.831	0.743	0.787
Varied SBE (Powell)	0.925	0.855	0.890	0.891	0.811	0.852
Varied SBD (Powell)	0.928	0.848	0.888	0.904	0.825	0.864
Varied SBR (Powell)	0.918	0.806	0.862	0.880	0.793	0.836
Varied SCR (PSO)	0.983	0.935	0.959	0.893	0.792	0.843
Varied SBE (PSO)	0.985	0.958	0.972	0.910	0.848	0.879
Varied SBD (PSO)	0.983	0.964	0.974	0.906	0.844	0.875
Varied SBR (PSO)	1.000	0.993	0.997	0.950	0.858	0.904

Table 5.4 summarized the multi-objective function values, retrieved OD and OC, when the multi-objective function was the searching criterion. Compared with the results in Tables 5.2 and 5.3, we notice that OD values were close to those when OD was the single objective to be optimized; similarly, OC values were close to those when OC was the single objective. But both criterions would come to a balance comparing the results when they were set as single criterion functions. Again, PSO still outperforms the Powell's method in optimization of multi-objective function for the four spectral transforms.

Table 5.5 Classification accuracy of DU classes

	15	23	30	45	60	75	90
GC	100.0	100.0	100.0	91.7	16.7	16.7	75.0
Original	100.0	100.0	100.0	83.3	33.3	83.3	83.3
Uniform SCR	100.0	100.0	100.0	100.0	33.3	91.7	41.7
Uniform SBE	100.0	100.0	100.0	100.0	33.3	66.7	100.0
Uniform SBD	100.0	100.0	100.0	100.0	33.3	66.7	100.0
Uniform SBR	100.0	100.0	100.0	83.3	33.3	91.7	83.3
Varied SCR (Powell)	100.0	95.0	100.0	88.3	43.3	93.3	33.3
Varied SBE (Powell)	100.0	100.0	100.0	98.3	35.0	85.0	100.0
Varied SBD (Powell)	100.0	100.0	100.0	100.0	37.5	100.0	100.0
Varied SBR (Powell)	100.0	100.0	100.0	81.7	38.3	96.7	86.7
Varied SCR (PSO)	100.0	100.0	100.0	90.0	71.7	91.7	60.0
Varied SBE (PSO)	100.0	100.0	100.0	98.3	50.0	85.0	100.0
Varied SBD (PSO)	100.0	100.0	100.0	100.0	44.4	88.9	100.0
Varied SBR (PSO)	100.0	100.0	100.0	100.0	58.3	91.7	91.7

The mean values of five runs were presented in the tables. To better show the performance statistics, boxplots with the information of mean and standard deviation were drawn in Figs. 5.5-5.7. They further confirmed that the multi-objective PSO (denoted as "OD/OC') provided comparable performance to the one optimized by a single-objective PSO (denoted as "OD" or "OC"). However, as shown in Fig. 5.7, the multi-objective PSO provided a better joint performance.

Table 5.5 showed the classification accuracy (OC) of DU classes at different depth for testing data (with the multi-objective), where PSO-based approaches performed the best.

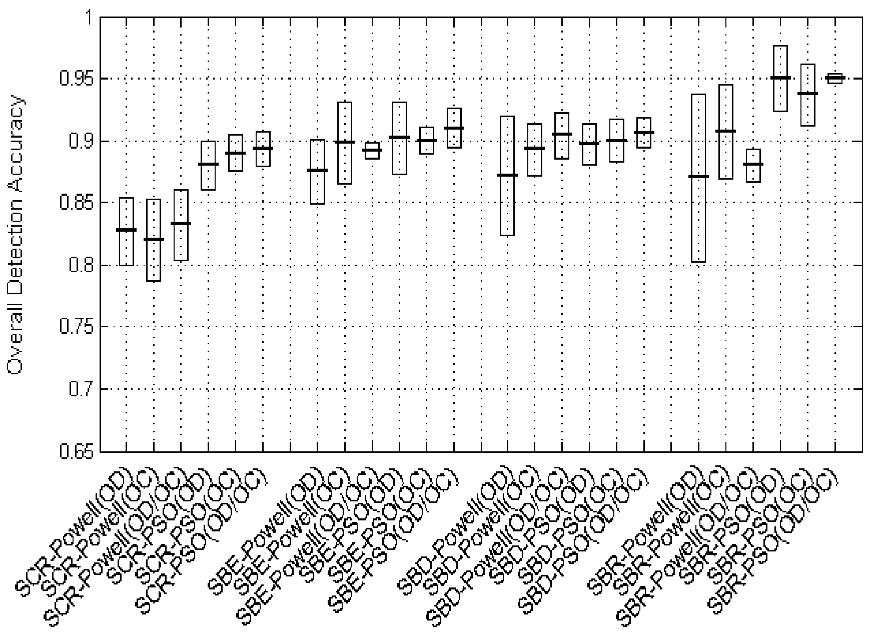


Figure 5.5 Boxplot of detection accuracy (OD) for the testing data.

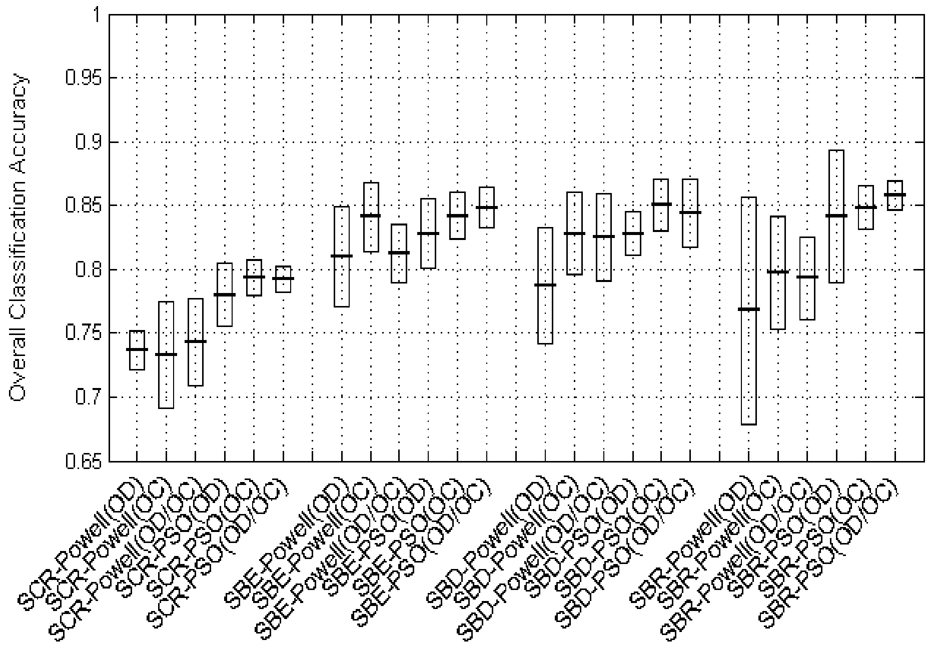


Figure 5.6 Boxplot of classification accuracy (OC) for the testing data.

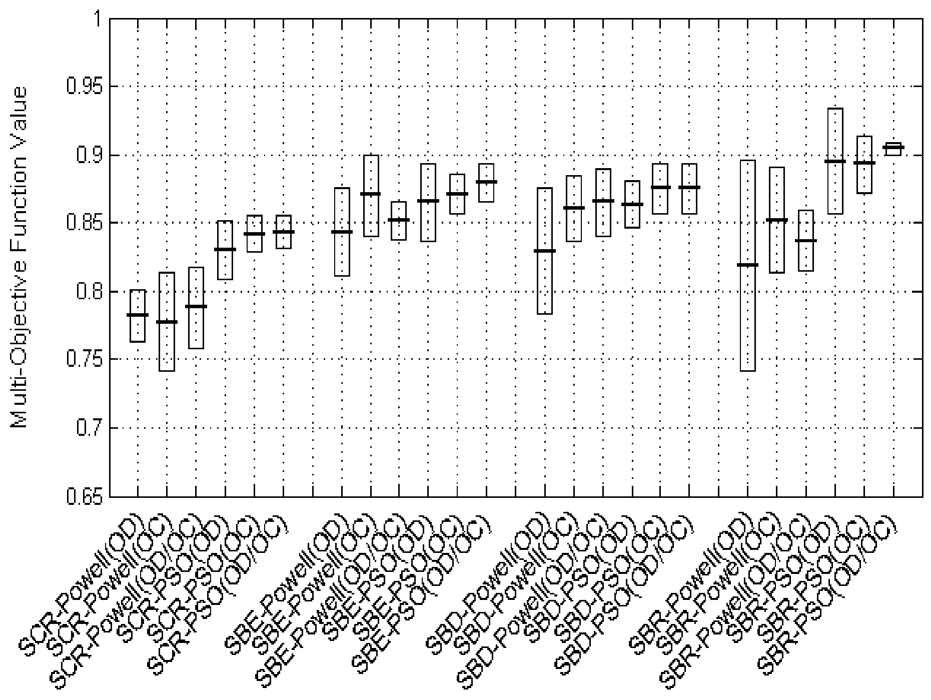


Figure 5.7 Boxplot of multi-objective function value (0.5 OD + 0.5 OC) of the testing data.

5.4 Conclusion

We propose a PSO-based adaptive optimization system to automatically determine the optimal number of bins and the corresponding optimal varied bin-widths for energy spectral transformation. Two PSOs are incorporated in the system with outer one being responsible for selecting the optimal number of bins while the inner one being in charge of searching for the optimal bin-widths. In our research, the overall detection accuracy and overall classification accuracy are the searching objectives. We can set them separately as the criterion function in the optimization process. To achieve high detection and classification accuracy, we propose a multi-objective PSO, which can well balance the detection and classification performance when both are of great concern in practical applications. From the optimization results, using variable bin-widths is better than fixed bin-width; PSO can provide better results than the Powell's method.

When analyzing an entire energy spectral curve, the data dimensionality is very high, which equals the number of energy channels. One of practical difficulties is the lack of enough training samples when implementing many traditional statistical methods, such as the maximum likelihood classifier. The proposed spectral partitioning approach can reduce the data dimensionality, thereby alleviating the requirement of a large training set. Our PSO-based searching system can be generalized to any detection or classification method.

However, due to the complexity of an objective function, the results are generally multimodal, exhibiting multiple local (and global) optima. Thus, multiple runs are required to find the best solution. Fortunately, with high performance computing facilities available, PSO can be easily implemented in parallel to reduce computing time.

CHAPTER VI

PARALLEL OPTIMIZATION

6.1 Introduction

PSO is a probabilistic searching algorithm based on a simplified social model. It has many attractive characters such as easy implementation. However, it is usually time-consuming when the problem space is high-dimensional. Fortunately, its basic structure is very suitable to parallel computation. Most parallel implementations of PSO algorithms are based on synchronous implementation [63],[63],[70] where all particles are evaluated within an iteration before the next design iteration is started. In such an implementation, all particles are sent to parallel computing nodes, and the algorithm waits for all the results from each computing node for analyses before entering to the next iteration. The problem is that it is nearly impossible to keep all processors working towards the end of each iteration. There are three reasons that would cause some of the processors being idle towards the end of iteration [65]: 1) having a swarm size that is not an integer multiple of the number of processors; 2) a heterogenous distributed computing environment including processors with varying computational speed; 3) using a numerical simulation to evaluate each design point, where the required simulation time depends on the design point being analyzed. The influence of having idle processors is reduction on speedup when more processors are needed.

Another type of parallel PSO is based on asynchronous parallel implementation where particles in the next design iteration are analyzed before the current design iteration is completed. In this way, no idle processors exist as the system evolves from current iteration to the next. The key to implementing an asynchronous parallel PSO algorithm is to divide the update of parameters associated with each point and those associated with the swarm as a whole. These update parameters include the inertia value, and the swarm and point histories. For the synchronous algorithm, all the update is performed at the end of each iteration. For the asynchronous algorithm, each particle is updated after its own iteration is completed. The swarm update action is done at the end of each iteration. The update actions in the algorithm, such as the velocity, the craziness operator, the dynamic reduction of the inertia value, need to be taken care of [66].

The fitness function in a PSO system should be as simple as possible to avoid large computational burden. Sometimes, we need to set the fitness function the one which provides the highest classification accuracy; this criterion function has to be selected based on small computation burden because classification needs to be conducted in each iteration. It is computationally prohibitive if the selected classifier, for example, support vector machine (SVM) [68], is very expensive with training and test. Here, we apply the k-NN which is a simple but effective classification method.

Although PSO algorithms present attractive global optima searching properties, they are plagued by high computational cost as measured by running time. It is natural to implement parallel computing for such an optimization system. Clusters are commonly used nowadays for high performance computing purpose. Venter *et al.* [65] developed an approach for parallel PSO to reduce the elapsed time, making use of coarse-grained parallelization to evaluate the design points. It utilized the interval time of receiving and sending by asynchronous parallel PSO algorithm that greatly improves the parallel efficiency. Lou *et al.* [69] proposed a multi-objective model of reactive power optimization (RPO). It takes account of power loss minimization, voltage stability margin maximization, and high service quality. Jin *et al.* [71] proposed a method that combines both the PSO and the finite-difference time-domain (FDTD) to achieve the optimum antenna satisfying a certain design criterion. The above parallel implementations of PSO result in great improvement in different applications.

Recently, graphics computing units (GPUs) are attracting more and more attention in engineering and science area due to its portability and low-cost. GPUs are first invented for graphics acceleration but it has been explored its computational power on general purpose computing [88]. GPU has already been successfully used in many computation fields, such as computer vision problems [89], Voronoi diagrams [90], neural network computation [91], and so on. It has also been applied to hyperspectral image analysis, e.g., detection, classification, and unmixing [92][93]. In this chapter, we will present GPU implementation for the PSO-based spectral optimization algorithm, and compare its performance with that of the cluster implementation.

6.2 Proposed Method

6.2.1 Parallel algorithm for one PSO on clusters

The synchronous parallel PSO algorithms are implemented. The parallel implementation used here is based on the Message Passing Interface (MPI) to provide a master-slave implementation , where one processor is used as the master processor, and all remaining processors are used as slave processors. The master processor is to collect data, determine the global update, and control the communication with the slave processors. The parallel implementation starts with the slave processors initiating themselves with random particle positions on their local temporary working memory. Within this working memory, the analysis is setup and conducted for the current particle.

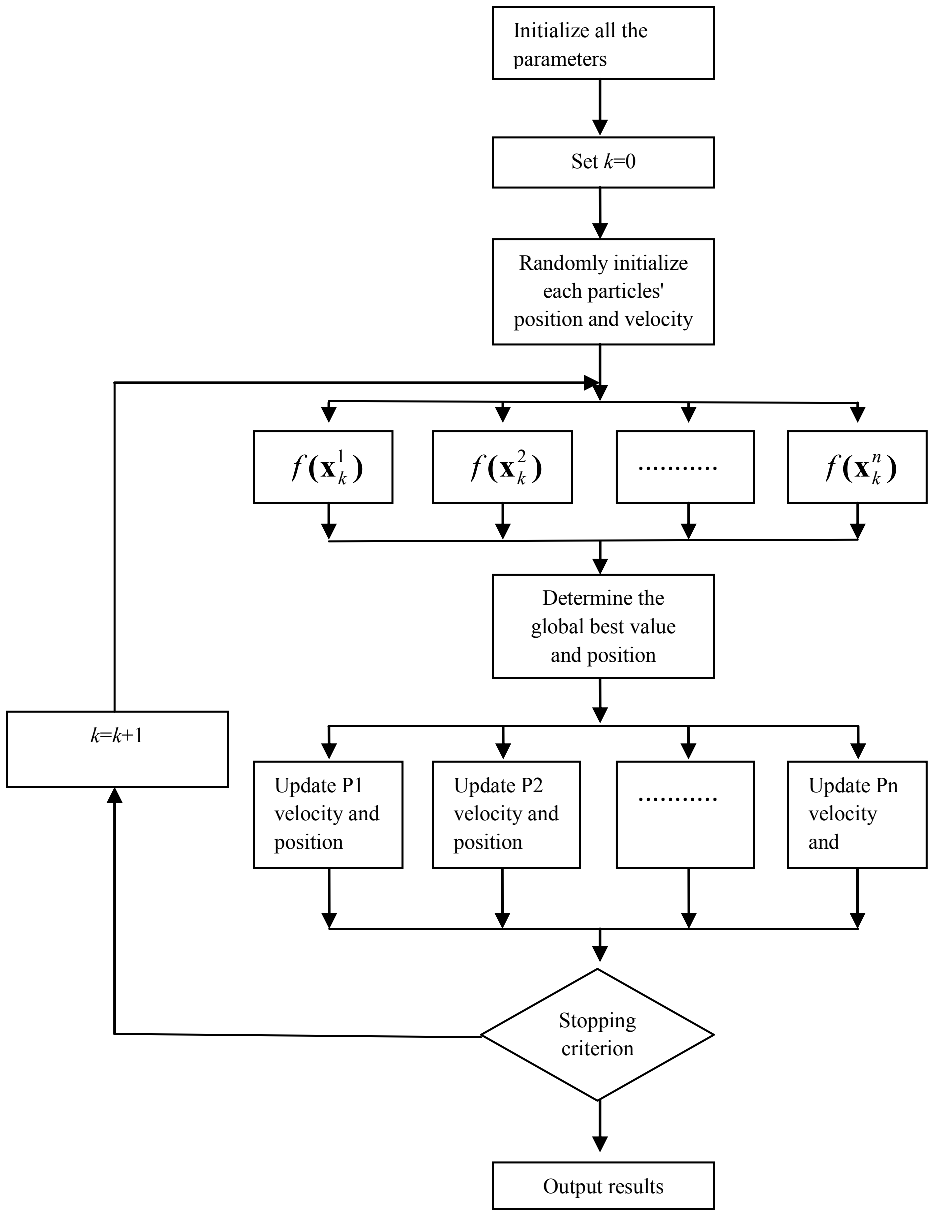

Initialize all the parameters
Set k=0
Randomly initialize each particles' position and velocity
$f(\mathbf{x}_k^1)$
$f(\mathbf{x}_k^2)$
....
$f(\mathbf{x}_k^n)$
Determine the global best value and position
k=k+1
Update P1 velocity and position
Update P2 velocity and position
....
Update Pn velocity and
Stopping criterion
Output results

Figure 6.1 The parallel PSO algorithm diagram

Once the analysis is completed, the objective function and locations are sent back to the master processor. The master processor checks the global optimal among the results from all other slaves and sends locations of current global optimum to the slave processors. This process is repeated until stopping criterion is met. The major characters of synchronous implementations are:

1. The synchronous algorithm performs all the particles and swarm updates at the end of each iteration.
2. The synchronous algorithm waits until all points are analyzed before updating and starting the next iteration.

The parallel PSO algorithm is illustrated by the flow chart in Fig. 6.1, where *k* represents the *k*-th iteration, $\mathbf{x}_k^n$ is the position vector of particle *n* in the *k*-th iteration, Pn represents the n-th particle , and *f*(●) *is a fitness function*

6.2.2 Parallel algorithm for one PSO on GPU

The GPU comes with the shared memory architecture. As all processors of the GPU can share data within a global address space, it fits the data parallelism very well. To achieve satisfied parallel performance, the data throughput is very critical in GPU parallel algorithm design, which means enough data and computation should be designed ahead to feed into the GPU to take advantage of its computing power. Many previous work shows that it can achieve excellent speedup performance only when the data size is increased to thousands. As it uses the share memory model, the major bottleneck is memory communication between the host and device; unnecessary data transfer between host and device should be avoided. After all, two key rules should be followed in the parallel algorithm design stage: 1) reduce the communication between host and device, and 2) parallel data size as large as possible.

In this chapter, our purpose is to accelerate the running speed of optimal bins using PSO searching method on GPU; meanwhile, optimization performance of the GPU-implemented PSO should not be deteriorated. By exploring the full power of the parallel computing ability of GPU, we expect the implementation can solve the global optimization problem for high-dimensional data with large swarm population.

6.2.2.1 Data organization

In PSO, the information of position and velocity for all the particles is stored in the global memory of GPU chips. One-dimensional arrays are used for storing parameters, including x_{id} (position), v_{id} (velocity), p_{id} (pbest) and p_{gd} (gbest) fitness values for all the particles. Here, we assume the dimension of the problem is D (equal to the number of bins), and the swarm population is N. So an array of length DN is used to represent each swarm by storing all the positions' velocity values. The pbest fitness values are stored on an array of length D.

6.2.2.2 Random number generation

In the process of optimization, PSO requires random numbers for velocity updating. Three random numbers are needed during each iteration. One is for the inertia weight and two are for the learning rates. As the absence of high precision integer arithmetic, generating random numbers in GPUs is not easy. Thus, we generate random numbers on CPU first and then transfer these numbers to the global memory of GPU. However, the data transportation between GPU and CPU is quite time consuming. If we generate random numbers on CPU for each iteration and then transfer them to GPU, the speedup performance will be degraded. In order to reduce the communication time between CPU and GPU as much as possible, we would transfer the random numbers in advance. First, we generate *T* random numbers on CPU before running PSO where *T* is large enough for need in the predefined iterations. Then they are transported to GPU global memory and stored in an array **R**. When it comes to the update of the velocity, we just pass three random numbers from **R** instead of transporting three times of Max Iterations random numbers from CPU to GPU. The running speed can be obviously improved by using this technique.

6.2.2.3 Overall algorithm of GPU-implemented PSO

The main steps illustrated in Fig. 6.2 for GPU-implemented PSO can be described as below. Here, we set the maximum number of iterations as the stopping criterion for the optimization process.

```
1. Initialize the positions and velocities of all particles.
2. Transfer these data from CPU to GPU's global memory.
3. for $i$ =1 to Max Iteration do
     Compute fitness values of all particles
     Update $p_{id}$ and $p_{gd}$ of each particle
     Update $p_{gd}$ and $p_{gd}$ position for all the particles
     Update $v_{id}$ and $x_{id}$ of each particle
   end for
4. Transfer results data back to CPU and output.
```

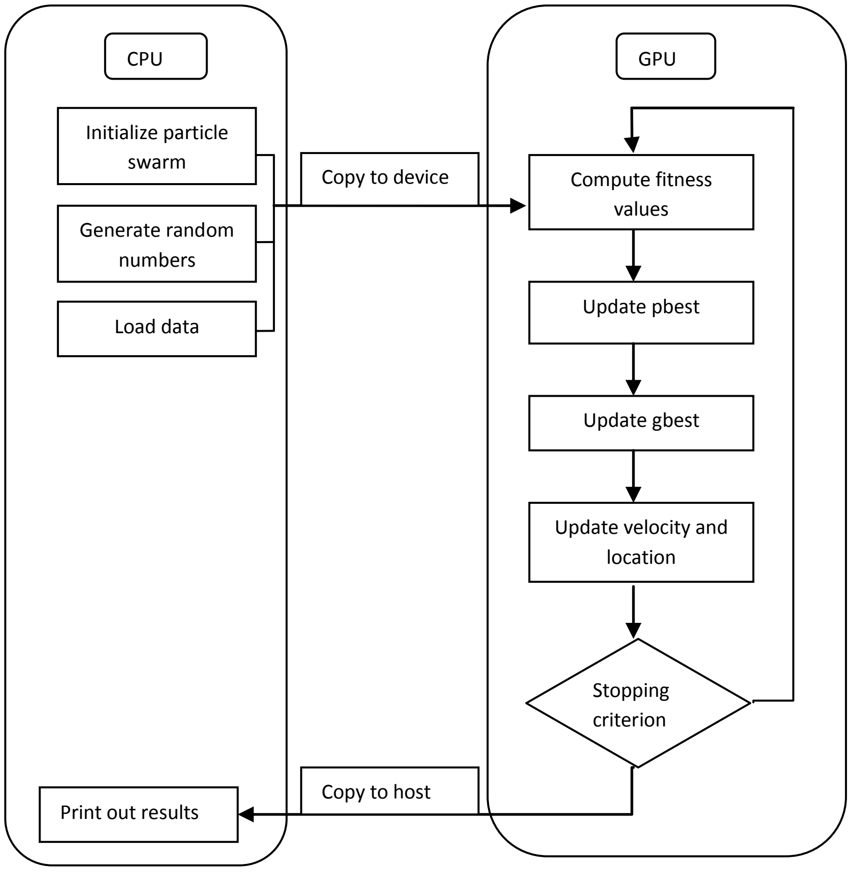


Figure 6.2 The diagram for the GPU-implemented PSO bins selection

6.2.2.4 Parallelization design on GPU

The difference between the implementation on CPU and a GPU kernel is that the kernel function of GPU is designed for single-instruction, multiple-data (SIMD) parallelized computing. So we design the parallelization methods for all the sub-processes in PSO.

1. Compute Fitness Values: Fitness values calculation is the most important task in the entire process, where the computation intensity is determined by the number of particles and the size of each particle. It should be carefully designed for parallelization so as to improve the overall

efficiency of the algorithm. The steps for fitness value calculation are shown as follows.

a. Set the block size and grid size with the number of threads equal to the number of particles *N*.
b. Load the position data of each particle from global memory to local memory of each thread.
c. Apply arithmetical operations to each thread for fitness function in parallel.
d. Store the final fitness values of all particles to an array.

2. Update p_{id} and p_{gd} : After the fitness values are computed, each particle may result in a better value than ever before in its history and new global best particles may be found. So p_{id} and p_{gd} must be updated according to the current information of the particles. The updating of p_{id} can be done as follows:

    a. Transfer p_{id} position, p_{id} fitness from global to shared memory of each block.
    b. Map each thread to each particle.
    c. If fitness value of any thread is better than its p_{id} fitness, then the new fitness value replaces the old one

```
  for each dimension D do
    Store the position to $p_{id}$
  end for
end if.
```

2. The update of gbest is different from that of pbest. Its parallel implementation is shown as below:
    a. Transfer p_{id} fitness data from global to shared memory.
    b. Apply the reduction on each block for minimum element; Store the minimum elements of each block to one array.
    c. Apply the reduction again for the array we got in step b.
    d. Update p_{gd} fitness and p_{gd} position by one thread.
3. Update Velocity and Position: After the p_{id} and p_{gd} position of all the particles have been updated, the velocities and positions should also be updated according to Eqs. (5.4) and (5.5), respectively. The update is critical in the whole algorithm which makes use of the new information provided by p_{id} and p_{gd}.
    a. Map each thread to each particle.
    b. for each dimension *D* do

```
  Transfer the $p_{id}$ and $p_{gd}$, particles' position and random number
to share memory of each block
  Update each particle on dimension $D$
end for.
```

6.2.3 Parallel hardware and software

In our study, we use a 2048 core cluster composed of 512 Sun Microsystems SunFire X2200 M2 servers, each with two dual-core AMD Opteron 2218 processors (2.6GHz) and 8 GB of memory. All of the computing nodes are diskless. The system uses gigabit ethernet to connect the 32 nodes in each rack together, and 10 GbE to connect the 16 racks to one another.

The CPU machine used in the experiments is an Intel Pentium4 3.40 GHz with Hyper thread and 2 GB of memory. The GPU is NVIDIA's GeForce GTX285 that has 240 cores with 1 GB memory.

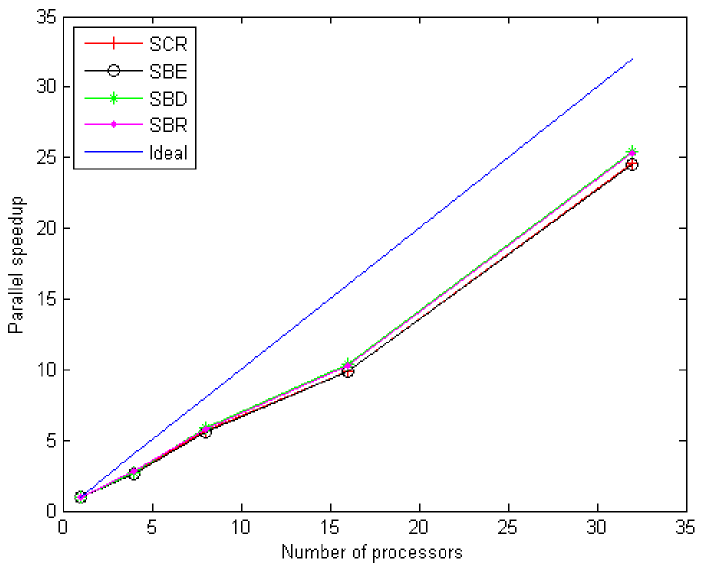
The parallel algorithms on the cluster are implemented in the C++ with the message passing interface (MPI) and EIGN library. The GPU versions are implemented in the CUDA.

6.3 Experiments

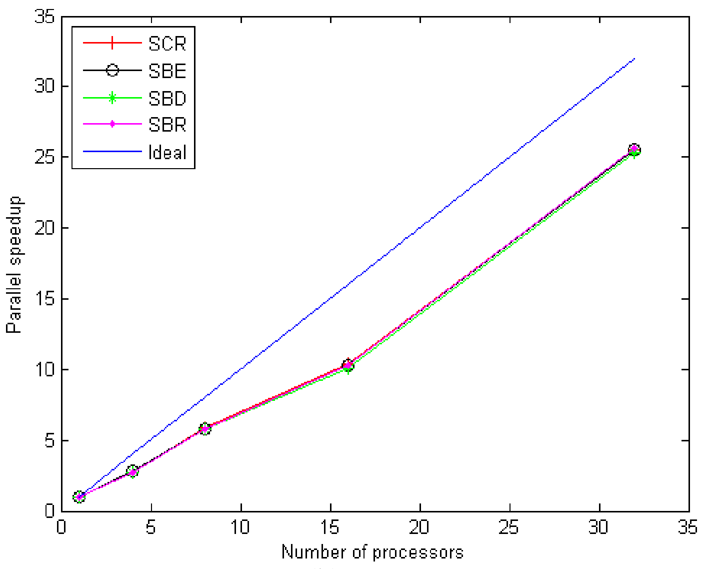
6.3.1 Cluster parallel implementation

Laboratory data was collected using a 10×10×40 cm sodium iodide (NaI) scintillation detector. The measured spectra covered the energy range from 0 keV to 2160.0 keV with 1011 channels. The target was buried at 15 cm, 23 cm, 30 cm, 45 cm, 60 cm, 75 cm, and 90 cm. Natural ore was buried at 45 cm and 75 cm depth. For each class, 24 samples were taken evenly by four different dwell times: 1 s, 0.5 s, 0.25 s, to 0.1 s. In the experiment, all the measurements were normalized into equivalent 1 s dwell time.

The lab data were divided into two parts with equal size. Each part contained 120 samples. Within the 120 samples, each class had 12 samples; among the 12 samples, each dwell time had 3 samples. We treated the first part of the data as training data and the second part were for testing. 1-NN with 3-fold cross validation was applied for detection and classification in both training and testing process.



(a) OD



(b) OC

Figure 6.3 The parallel speedup for two objective functions.

Table 6.1 Accuracy from the best uniform partitions

	Training Data				Testing Data			
	SCR	SBE	SBD	SBR	SCR	SBE	SBD	SBR
OD	89.2	94.2	94.2	95.8	83.3	85.8	87.5	91.2
OC	80.0	87.5	87.5	85.0	72.5	82.5	82.5	82.5

Table 6.2 Detection accuracy from parallel PSO-based methods

	Training Data				Testing Data			
Number of Processors	SCR	SBE	SBD	SBR	SCR	SBE	SBD	SBR
1	96.7	99.2	99.2	100.0	85.8	90.8	94.2	93.3
4	97.5	98.3	99.2	100.0	81.7	87.5	91.7	90.0
8	97.5	99.2	99.2	100.0	90.0	93.3	92.5	91.7
16	97.5	99.2	98.3	100.0	89.2	90.8	89.2	94.2
32	96.7	99.2	99.2	100.0	78.3	91.7	90.8	70.0
Average	97.2	99.0	99.0	100.0	85.0	90.8	91.7	87.8

The spectral transforms employed the PSO to automatically determine the varied bin-widths. Different numbers of bins were exhausted from 3 to 20. Fig. 6.3shows the parallel speedup of different number of processors. As the number increases, the computational burden was distributed and running time was decreased. For the four different spectral transforms, the speedups were almost the same. The objective function did not affect the speedup.

Table 6.3 Classification accuracy from parallel PSO-based methods

	Training Data				Testing Data			
Number of Processors	SCR	SBE	SBD	SBR	SCR	SBE	SBD	SBR
1	90.0	95.0	95.0	98.3	72.5	82.5	85.0	88.3
4	90.0	96.7	94.2	97.5	78.3	84.2	82.5	86.7
8	90.8	94.2	95.0	97.5	80.8	80.0	83.3	85.8
16	90.0	95.0	96.7	97.5	74.2	84.2	83.3	85.8
32	88.3	94.2	94.2	96.7	76.7	84.2	85.0	82.5
Average	89.8	95.0	95.0	97.5	76.5	83.0	83.8	85.8

The gross count (GC) yielded OD = 0.667 and OC = 0.550, while original data provided OD = 0.850 and OC = 0.767, better than GC. Table 6.1 lists the accuracy values from the best uniform partitions with fixed bin-widths. Tables 6.2 and 6.3 tabulate the detection and classification accuracy for the four transforms after parallel PSO-based bin partitions, which are better than those in Table 6.1. Note that when the number of processors is 1, it is the serial version. Fig. 6.4 illustrates the OD and OC values for the testing data, which clearly shows the improvement from the use of bin optimization.

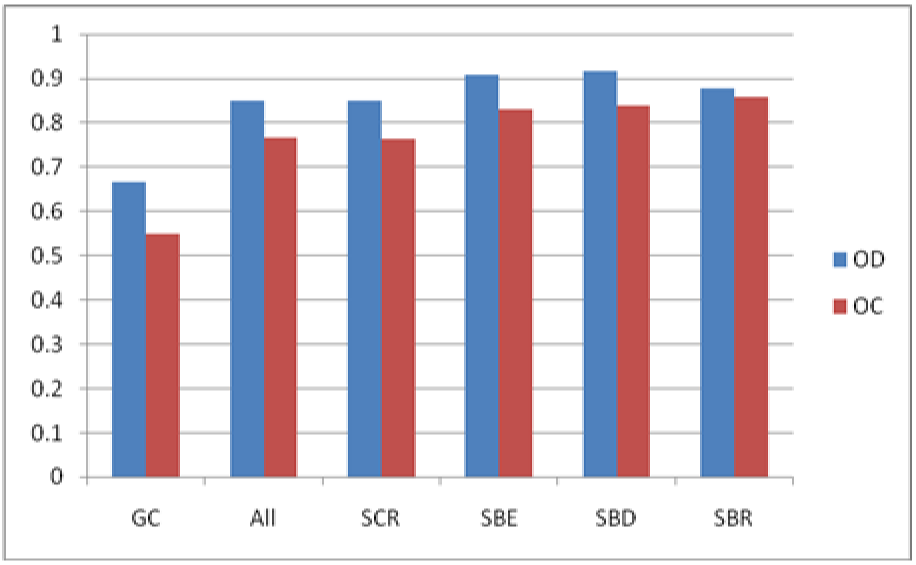


Figure 6.4 Comparision on OD and OC using the testing data.

6.3.2 GPU parallel implementation

As the four transformation methods are similar in the parallel implementation, we only implement the SCR with objective of OC. In this experiment, we set the number of particle swarm size of 300 to fully explore the computational power of GPU. The maximum iteration was set as 500 which was large enough in this case. The average running time in the cluster and GPU is shown in Table 6.4, and the speedup comparison of both parallel computing is shown in Figs. 6.5 and 6.6. From these results, we can see that the GPU achieved slightly better performance than the cluster with 32 cores.

6.3.2.1 Running time and speedup versus number of bins

Now we fixed the swarm population to a constant value but vary the number of bins to be selected. Analysis about the relationship between running time (as well as speedup) and the number of bins was conducted. The parallel PSO was run for five times, and the average results are shown in Table 6.4 (*N*=300, *Iter*=500).

As seen from Fig. 6.5, the speedup of the cluster system has not been affected much by the number of bins selected, or particle dimension. But from Fig. 6.6 of the speedup of GPU, the accelerations were decreased as the number of bins being increased. This is because when the number of bins is increased, it directly leads to the increase of particle dimension, which greatly degraded the speedup performance of GPU.

Table 6.4 further shows that the computation time all increased but that of GPU increased much in proportion compared with the serial algorithm. In other words, we could say the cluster has fast computation speed and speedy data transfer mechanism, thus the increase of particle dimension does not induce more computational burden.

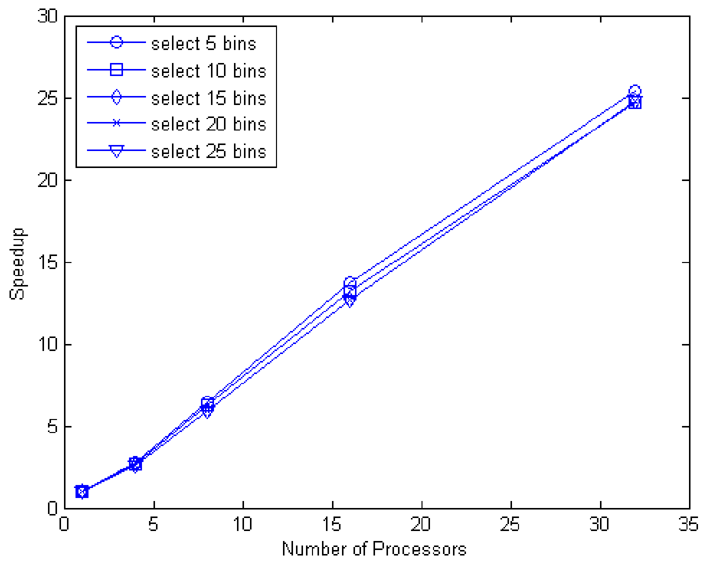


Figure 6.5 Speedup performance with different number of bins selected by Cluster

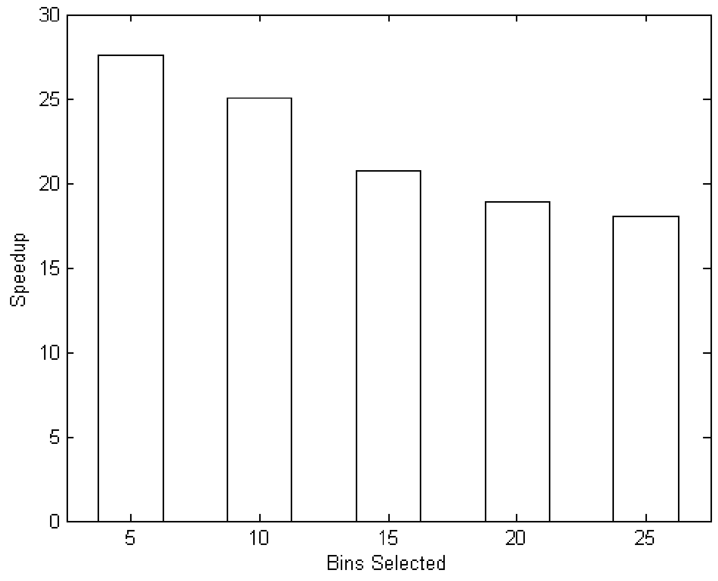


Figure 6.6 Speedup performance with different number of bins selected by GPU

Table 6.4 Results of parallel bins selection running time (in second)

Number of selected bins	5	10	15	20	25
1 processor	143.566	170.099	188.708	224.557	250.203
4 processor	51.380	62.589	71.810	82.737	92.248
8 processor	22.374	26.940	31.996	36.443	40.193
16 processor	10.433	12.905	14.917	16.762	18.812
32 processor	5.660	6.882	7.606	8.678	9.725
GPU	5.205	6.810	9.113	11.901	13.870

6.3.2.2 Running time and speedup versus swarm size

Now we fixed the number of bins and explored the effect of swarm size on the parallel performance. Corresponding analysis of running time on different swarm size was performed. The parallel PSO was run for five times, and the average results are shown in Table 6.5 (*Dimension=25*, *Iter*=500).

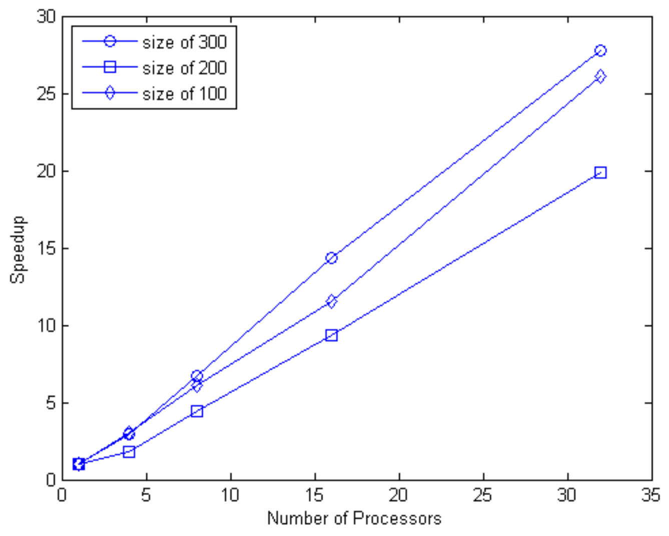


Figure 6.7 Speedup performance with different swarm size for Cluster

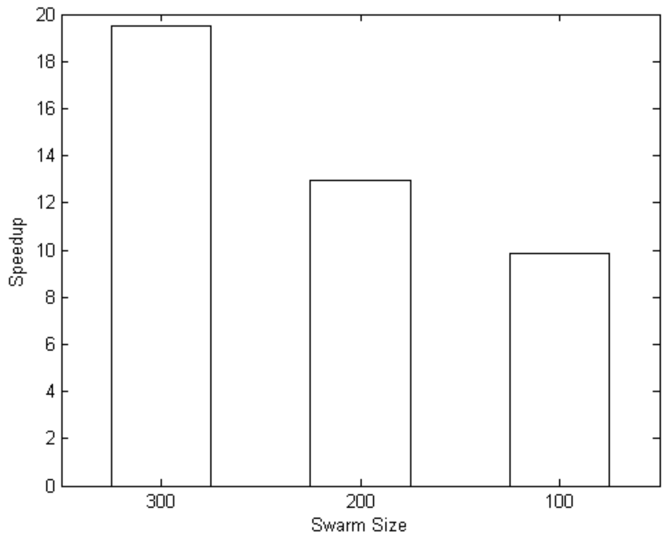


Figure 6.8 Speedup performance with different swarm size for GPU

As we can see from Fig. 6.7, the speedup of the cluster system was largely affected by the parameter of swarm size in PSO, especially when the number of processors was increased. This illustrates that as the swarm size allocated on each processor was decreased, the overhead between the processors became dominant and thus reduced the related speedup. From Fig. 6.8 about the speedup of GPU, the acceleration was also decreased as the swarm size being reduced, which is again due to the communication overhead between the host and device. This can be shown by the fact that the size of 100 has worse speedup than the size of 300.

The detailed running time of GPU is shown in Table 6.5. Although the decrease of swarm size brought down directly the computational intensity, the performance was also affected due to the fact that the proportion of the communication overhead was enlarged in the entire computation process.

Table 6.5 Results of parallel PSO running time (in second)

Swarm size	300	200	100
1 processor	270.203	115.235	77.866
4 processor	92.248	64.091	25.953
8 processor	40.193	26.037	12.801
16 processor	18.812	12.375	6.773
32 processor	9.725	5.796	2.988
GPU	13.870	8.912	7.903

6.4 Conclusion

We proposed a PSO-based optimization method for automatic bin partition to mitigate the impact from sparseness and randomness in an energy spectrum. The experiment shows that spectral transformation using PSO-selected bins can provide better results than the best uniform partition. In this chapter, a parallel PSO algorithm is developed, which can significantly reduce running time while maintaining the overall detection and classification accuracy. Since parallel computation is an appropriate approach to reduce the computation burden of the PSO-based searching process, the parallel implementation on cluster for optimal bin partition is effective in reducing the workload of the search algorithm. The speedup performance and resulting detection and classification performance are investigated. Corresponding results show that the speedup is almost linear with the number of processors involved in the PSO searching process.

We also proposed GPU parallel implementation. The GPU facility is currently popular in scientific computing. The experimental results show that GPU implementation has high scalability and is comparable to cluster implementation. In addition, we notice that the running time and swarm population size take an approximately linear relationship, which is also true for running time and dimension. However, the swarm size has a major impact on the speedup performance; to fully explore the power of GPU, a large size of swarm should be adopted.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this research, new methods have been developed to achieve detection and classification of buried radioactive materials. We have developed effective spectral transformation methods for this purpose. The transformation has been successfully suppressed the variation of noise introduced by the collecting equipments and enforce the feature in the new transformed spectral space. In this chapter, specific conclusions can be drawn in the following aspects:

1) We have introduced the anomaly detection methods to the detection of buried radioactive materials. Finding a way to detect the illicit sources is not an easy problem under the assumption that there is no predefined ground true sample. The classic RX has been employed to search the possible anomaly samples and they are used in conjunction with CEM for final decision. This unsupervised system can suppress the background noise by using the dimensionality-reduced data from energy windowing and PCA.

2) We improve the performance of buried target detection and classification by using NAPCA. A spectral transform was firstly used to alleviate the effects from spectral noisy variation and background clutters; then NAPCA, a better choice than PCA, can extract key target features from the spectrally-transformed data, thereby further improving the detection and classification performance.

3) In spectral transformation, uniform energy windowing is usually used. However, uniform partition often could not provide the best performance. In this

dissertation, we propose an adaptive optimization system with evolutionary algorithm like PSO to automatically determine the optimal number of bins and the corresponding optimal varied bin-widths for energy spectral transformation. In order to fulfill this purpose, we propose that two PSOs are incorporated in the system with the outer one being responsible for selecting the optimal number of bins and the inner one for optimal bin-widths. The experimental results demonstrate that using variable bin-widths is better than a fixed bin-width, and PSO can provide better results than the traditional Powell's method.

4) Due to the computational cost of evolutionary algorithm like PSO, we propose the parallel implementation scheme for the PSO-based bin partition algorithm. The master and slave model is used in this implementation. It can greatly reduce the time of training process. The graphics processing units (GPU) application in engineering is more popular in these years. Their portability and efficiency are being emphasized by more and more people. In this dissertation, the implementation for parallel PSO-based spectral transformation has been experimented. The computational burden of serial version has been greatly reduced. The experimental results show that the GPU algorithm has similar speedup as the cluster-based algorithm.

In this research, both detection and classification accuracy are our most concerns. We have done some work on the evaluation of multi-objective optimization by using the weighted method. However, a more sophisticated method should be developed to improve multi-objective optimization. Some researchers have investigated the ability of PSO to detect Pareto Optimal points and to capture the shape of the Pareto Front. The future work in our research is to develop Pareto-based multi-objective method for energy window optimization.

We will also concern with another existing problem in the future work. An energy spectrum is nonlinearly correlated with the mass and depth of buried materials. Although we explored using a traditional back-propagation neural network, the mass/depth prediction accuracy did not meet our expectation. The performance could be improved by more advanced neural networks.

REFERENCES

[1] E. R. Siciliano, J. J. Ely, R. T. Kouzes, B. D. Milbrath, J. E. Schweppe, and D. C. Stromswold, "Comparison of PVT and NaI(Tl) scintillators for vehicle portal monitor applications," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 550, pp. 647-674, 2005.

[2] D. C. Stromswold, E. R. Siciliano, J. E. Schweppe, J. H. Ely, B. D. Milbrath, R. T. Kouzes, B. D. Geelhood, "Comparison of plastic and NaI scintillators for vehicle portal monitor applications," *IEEE Nuclear Science Symposium Conference Record*, pp. 1065-1069, 2003.

[3] J. H. Ely, R. T. Kouzes, B. D. Geelhood, J. E. Schweppe, and R. A. Warner, "Discrimination of naturally occurring radioactive material in plastic scintillator material," *IEEE Transactions on Nuclear Science*, vol. 51, no. 4, pp. 1672-1676, 2004.

[4] R. T. Kouzes, J. H. Ely, B. D. Geelhood, R. T. Hansen, E. Lepel, J. Schweppe, E. R. Siciliano, D. T. Strom, and R. E. Warner, "Natureally occurring radioactive materials and medical isotopes at border crossings," *IEEE Nuclear Science Symposium Conference Record*, pp. 1448-1452, 2003.

[5] D. C. Stromswold, J. Darkoch, J. H. Ely, R. R. Hansen, R. T. Kouzes, B. D. Milbrath, R. C. Runkle, W. A. Sliger, J. E. Smart, D. L. Stephens, L. C. Todd, and M. L. Woodring, "Field tests of a NaI(Tl)-based vehicle portal monitor at border crossings," *IEEE Nuclear Science Symposium Conference Record*, vol. 1, pp. 196-200, 2004.

[6] J. H. Ely, R. T. Kouzes, J. E. Schweppe, E. R. Siciliano, D. M. Strachan, and D. R. Weier, "The use of energy windowing to discriminate SNM from NORM in radiation portal monitors," *Nuclear Instruments and Methods in Physics Research*, vol. 560, no. 2, pp. 373-387, 2006.

[7] R. T. Kouzes and J. H. Ely, "The role of spectroscopy versus detection for border security," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 276, no. 3, pp. 719-723, 2007.

[8] D. S. Haslip, T. Cousins, D. Estan, and T. Jones, "Field detection of depleted uranium final report of tasking W28476KR00Z (DSSPM)," *Defence Research Establishment*, *Ottawa (Ontario), Report No: DREO TM-2000-049*, 2000.

[9] K. K. Anderson, K. D. Jarman, M. L. Mann, D. M. Pfund, and R. C. Runkle, "Discriminating nuclear threats from benign sources in Gamma-ray spectra using a spectral comparison ratio method," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 276, pp. 713-718, 2008.

[10] D. M. Pfund, R. C. Runkle, K. K. Anderson, and K. D. Jarman, "Examination of count-starved Gamma spectra using the method of spectral comparison ratios," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 1232-1238, 2007.

[11] C.-I Chang and D. Heinz, "Constrained subpixel spectral detection for remotely sensed images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 3, pp. 1144-1159, 2000.

[12] I. S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 38, no. 10, pp. 1760-1770, 1990.

[13] W. H. Farrand and J.C. Harsanyi, "Mapping the distribution of mine tailing in the coeur d'Alene river valley, Idaho through the use of constrained energy minimization technique," *Remote Sensing of Environment*, vol. 59, no. 1, pp. 64-76, 1997.

[14] J. C. Harsanyi, W. Farrand, and C.-I Chang, "Detection of subpixel spectral signatures in hyperspectral image sequences," *Proceedings of American Society of Photogrammetry and Remote Sensing*, pp. 236–247, 1994.

[15] B. D. VanVeen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4-24, 1988.

[16] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[17] M. L. G. Althouse and C.-I Chang, "Chemical vapor detection with a multispectral thermal imager," *Optical Engineering*, vol. 30, no. 11, pp. 1725-1733, 1991.

[18] Q. Du, H. Ren, and C.-I. Chang, "A comparative study for orthogonal subspace projection and constrained energy minimization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 6, pp. 1525-1529, 2003.

[19] A. F. El-Rewainy, E. E. Farouk, A. E. Fouda, "A comparison study of dual window-based anomaly detection algorithms for hyperspectral imagery," *13th*

International Conference on Aerospace Sciences & Aviation Technology, pp. 22, 2009.

[20] R. C. Runkle, M. F. Tardiff, K. K. Anderson, D. K. Carlson, L. E. Smith, "Analysis of spectroscopic radiation portal monitor data using principal components analysis," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 1418-1423, 2006.

[21] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, pp. 559–572,1901.

[22] A. A. Green, M. Berman, P. Switzer, M. D. Craig, "A transformation for ordering multispectral data in terms of image quality with implications for noise removal," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 26, no. 1, pp. 65-74, 1988.

[23] J. B. Lee, A. S. Woodyatt, and M. Berman, "Enhancement of high spectral resolution remote sensing data by a noise-adjusted principal components transform," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 3, pp. 295-304, 1990.

[24] C.-I. Chang, and Q. Du, "Interference and noise-adjusted principal components analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 5, pp. 2387-2396, 1999.

[25] R. E. Roger and J. F. Arnold, "Reliably estimating the noise in AVIRIS hyperspectral imagers," *International Journal of Remote Sensing*, vol. 17, pp. 1951-1962, 1996.

[26] C.-I. Chang, "Hyperspectral imaging: techniques for spectral detection and classification," *Kluwer Academic/Plenum Publishers*, 2003.

[27] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swam theory," *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pp. 39-43, 1995.

[28] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications, resources," *Proceedings of the Congress on Evolutionary Computation*, pp. 81-86, 2001.

[29] X. Hu and R. C. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," *Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[30] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

[31] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[32] X. D. Li and A. P. Engelbrecht, “Particle swarm optimization: an introduction and its recent developments,” *Proceedings of the Congress on Evolutionary Computation*, pp. 3391–3414, 2007.

[33] R. A. Krohling and L. dos Santos Coelho, “Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems,” *IEEE Transactions on System, Man, Cybernetics,* vol. 36, no. 6, pp. 1407–1416, 2006.

[34] N. Franken and A. P. Engelbrecht, “Particle swarm optimization approaches to coevolve strategies for the iterated prisoner’s dilemma,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 562–579, 2005.

[35] S.-Y. Ho, H.-S. Lin, W.-H. Liauh, and S.-J. Ho, “OPSO: Orthogonal particle swarm optimization and its application to task assignment problems,” *IEEE Transactions on System, Man, Cybernetics Part A: System, Humans*, vol. 38, no. 2, pp. 288–298, 2008.

[36] B. Liu, L. Wang, and Y. H. Jin, “An effective PSO-based memetic algorithm for flow shop scheduling,” *IEEE Transactions on System, Man, Cybernetics Part B: Cybernetics*, vol. 37, no. 1, pp. 18–27, 2007.

[37] R. C. Eberhart and Y. Shi, “Guest editorial,” *IEEE Transactions on Evolutionary Computation—Special Issue Particle Swarm Optimization*, vol. 8, no. 3, pp. 201–203, 2004.

[38] R. Huang and X. Li, “Band selection based on evolution algorithm and sequential search for hyperspectral classification,” *IEEE International Conference on Audio, Language and Image Processing,* pp. 1270-1273, 2008.

[39] S. T. Monteiro and Y. Kosugi, “A particle swarm optimization-based approach for hyperspectral band selection,” *IEEE Congress on Evolutionary Computation,* pp. 3335-3340, 2007.

[40] H. Gao, M. Mandal, and J. Wan, "Classification of hyperspectral image with feature selection and parameter estimation," *International Measuring Technology and Mechatronics Automation, International Conference*, pp. 783-786, 2010.

[41] Y. Shi and R. C. Eberhart, “Fuzzy adaptive particle swarm optimization,” *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 101-106, 2001.

[42] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 94-100, 2001.

[43] P. J. Angeline, "Using selection to improve particle swarm optimization," *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 84-89, 1998.

[44] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on System, Man, Cybernetics Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, 2004.

[45] Y. P. Chen,W. C. Peng, and M. C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Transactions on System, Man, Cybernetics Part B: Cybernetics*, vol. 37, no. 6, pp. 1460–1470, 2007.

[46] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization," *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1044-1051, 2006.

[47] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 124-129, 2005.

[48] W. J. Zhang and X. F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," *Proceedings of IEEE Conference of System, Man, Cybernetics*, pp. 3816-3821, 2003.

[49] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211–224, 2004.

[50] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Applied Mathematics Computation*, vol. 189, no. 2, pp. 1859–1883, 2007.

[51] D. Parrott and X. D. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 440–458, 2006.

[52] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," *Proceedings of IEEE Congress on Evolutionary Computation*, 2002.

[53] J. Kennedy and R. Mendes, "Neighborhood topologies in fully informed and best-of-neighborhood particle swarms," *IEEE Transactions on System, Man, Cybernetics Part C: Applications*, vol. 36, no. 4, pp. 515–519, 2006.

[54] X. Hu and R. C. Eberhart, “Multiobjective optimization using dynamic neighborhood particle swarm optimization,” *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1671-1676, 2002.

[55] J. J. Liang and P. N. Suganthan, “Dynamic multi-swarm particle swarm optimizer,” *Proceedings of Swarm Intelligence Symposium*, pp. 124-129, 2005.

[56] R. Mendes, J. Kennedy, and J. Neves, “The fully informed particle swarm: Simpler, maybe better,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[57] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[58] A. Ratnaweera, S. Halgamuge, and H. Watson, “Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients,” *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[59] M. Clerc, “The swarm and the queen: towards a deterministic and adaptive particle swarm optimization,” *Proceedings of Congress on Evolutionary Computation*, 1999.

[60] S. Koziel and Z. Michalewicz, “Evolutionary algorithms, homoorphous mappings, and constrained parameter optimization,” *Evolutionary Computation*, vol. 7, no. 1, pp. 19-44, 1999.

[61] H. Kwon and N. M. Nasrabadi, ”Kernel RX-algorithm: a nonlinear anomaly detector for hyperspectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 2, pp. 388-397, 2005.

[62] C. I. Chang and S.-S. Chiang, “Anomaly detection and classification for hyperspectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 2, pp. 1314-1325, 2002.

[63] J. F. Schutte, B. J. Fregly, R. T. Haftka, and A., George, “A parallel particle swarm algorithm,” *Proceedings of the Fifth World Congress of Structural and Multidisciplinary Optimization*, 2003.

[64] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka and A. D. George, “Parallel global optimization with the particle swarm algorithm,” *International Journal of Numerical Methods in Engineering*, vol. 61, pp. 2296-2315, 2004.

[65] G. Venter and B. Watson, “Efficient optimization algorithms for parallel applications,” *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-2000-481, 2000.

[66] G. Venter and J. Sobieszczanski-Sobieski, “A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations,” *Proceedings of* the *6th World Congresses of Structural and Multidisciplinary Optimization*, pp. 123-137, 2005.

[67] http://www.hpc.msstate.edu/computing/hpc.php

[68] V. N. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer, 1995.

[69] S. Lou, Y. Wu, X. Xiong, G. Tu, “A parallel PSO approach to multi-objective reactive power optimization with static voltage stability consideration,” *IEEE conference on Transmission and Distribution Conference and Exhibition,* pp. 172-176, 2005.

[70] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George “Parallel global optimization with the particle swarm algorithm,” *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 2296-2315, 2004.

[71] N. Jin and Y. Rahmat-Samii, “Parallel particle swarm optimization and finite-difference time-domain (PSO/FDTD) algorithm for multiband and wide-band patch antenna designs,” *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3459-3468, 2005.

[72] N. Christiani and J. Shawe-Taylor, *An Introduction to Support Vector Machines*,Cambridge, MA: Cambridge Univ. Press, 2000.

[73] S. Gunn, “Support vector machines for classification and regression,” *Image Speech and Intelligence Syst. Group, University of Southampton, Southampton, U.K., Technical Report*, 1998.

[74] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[75] R. M. Rifkin, *Everything old is new again: A fresh look at historical approaches in machine learning*, Ph.D. dissertation, MIT Cambridge, Cambridge, 2002.

[76] C.-W. Hsu and C.-J. Lin, “A comparison of methods formulticlass support vector machines,” *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[77] T. F. Wu, C. J. Lin, and R. C. Weng, “Probability estimates for multiclass classification by pairwise coupling,” *Journal of Machine Learning Research.*, vol. 5, pp. 975–1005, 2004.

[78] F. Melgani and L. Bruzzone, “Classification of Hyperspectral Remote Sensing Images With Support Vector Machines,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 8, pp. 1778-1790, 2004.

[79] C. B. E. Boser, I.M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” *Proceedings of 5th Annual ACM Workshop Computational Learning Theory*, pp. 144-152, 1992.

[80] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[81] Set of tutorials on SVM’s and kernel methods [Online]. Available: http://www.kernel-machines.org/tutorial.html.

[82] I. El-Naqa, Y. Yongyi, M. N. Wernick, N. P. Galatsanos, and R. M. Nishikawa, “A support vector machine approach for detection of microcalcifications,” *IEEE Transactions on Medical Imaging*, vol. 21, pp. 1552–1563, 2002.

[83] J. Robinson and V. Kecman, “Combining support vector machine learning with the discrete cosine transform in image compression,” *IEEE Transactions on Neural Networks*, vol. 14, pp. 950–958, 2003.

[84] M. Pontil and A. Verri, “Support vector machines for 3D object recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 637–646, 1998.

[85] S. A. Dyer, *Survey of instrumentation and measurement*, Wiley, 2001.

[86] W. H. Press, B. P.Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C-The art of scientific computing*, Cambridge University Press, 1992.

[87] B. Thai and G. Healey, “Invariant subpixel detection in hyperspectral imagery,” *IEEE Transactions on Geoscience Remote Sensing*, vol. 40, pp. 599–608, 2002.

[88] R. Yang and G. Welch, “Fast image segmentation and smoothing using commodity graphics hardware,” *Journal of Graphics Tools, special issue on Hardware-Accelerated Rendering Techniques*, pp. 91-100, 2003.

[89] K. E. Hoff III, T. Culver, J. Keyser, M. Lin, and D. Manocha, “Fast computation of generalized voronoi diagrams using graphics hardware,” *Proceeding of SIGGRAPH*, pp. 277- 286, 1999.

[90] Z. W. Luo, H. Z. Liu, and X. C. Wu, “Artificial neural network computation on graphic process unit,” *Proceedings of International Joint Conference on Neural Networks*, pp. 622-626, 2005.

[91] J. Setoain, M. Prieto, C. Tenllado, and F. Tirado, “Real-time onboard hyperspectral image processing using programmable graphics hardware,” in *High Performance Computing in Remote Sensing*, A. Plaza and C.-I. Chang, Eds. London, U.K.: Chapman & Hall/CRC, 2008.

[92] J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado, “Parallel morphological endmember extraction using commodity graphics hardware,” *IEEE Geoscience Remote Sensing Letter*, vol. 4, pp. 441–445, 2007.

[93] A. Paz and A. Plaza, “Clusters versus GPUs for parallel target and anomaly detection in hyperspectral images,” *EURASIP Journal of Advanced Signal Processing*, 915639, 2010.