Sparse Representation Based Hyperspectral Image Compression and Classification

Hairong Wang

Supervised by Professor Turgay Celik

WITS
UNIVERSITY

A thesis submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy.

May 2018, Johannesburg

Declaration

I, Hairong Wang, declare that this Thesis is my own, unaided work. It is being submitted for the Degree of Doctor of Philosophy at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

[illegible]

____

(Signature of candidate)

Signed on the 7th day of May 2018 in Johannesburg

Abstract

This thesis presents a research work on applying sparse representation to lossy hyperspectral image compression and hyperspectral image classification. The proposed lossy hyperspectral image compression framework introduces two types of dictionaries distinguished by the terms sparse representation spectral dictionary (SRSD) and multi-scale spectral dictionary (MSSD), respectively. The former is learnt in the spectral domain to exploit the spectral correlations, and the latter in wavelet multi-scale spectral domain to exploit both spatial and spectral correlations in hyperspectral images. To alleviate the computational demand of dictionary learning, either a base dictionary trained offline or an update of the base dictionary is employed in the compression framework. The proposed compression method is evaluated in terms of different objective metrics, and compared to selected state-of-the-art hyperspectral image compression schemes, including JPEG 2000. The numerical results demonstrate the effectiveness and competitiveness of both SRSD and MSSD approaches.

For the proposed hyperspectral image classification method, we utilize the sparse coefficients for training support vector machine (SVM) and k-nearest neighbour (kNN) classifiers. In particular, the discriminative character of the sparse coefficients is enhanced by incorporating contextual information using local mean filters. The classification performance is evaluated and compared to a number of similar or representative methods. The results show that our approach could outperform other approaches based on SVM or sparse representation.

This thesis makes the following contributions. It provides a relatively thorough investigation of applying sparse representation to lossy hyperspectral image compression. Specifically, it reveals the effectiveness of sparse representation for the exploitation of spectral correlations in hyperspectral images. In addition, we have shown that the discriminative character of sparse coefficients can lead to superior performance in hyperspectral image classification.

Acknowledgements

I would like to extend my sincere gratitude to my supervisor, Professor Turgay Celik. Professor Celik has probably been the best supervisor I could ask for. He constantly encouraged me to explore new research ideas; he was always open to me to discuss my research progress, to give good advices in tackling my research problems; his positiveness and enthusiasm have been truly inspiring. I have learnt so many things from him, and he helped to make my PhD study a worthwhile experience.

I am deeply grateful to my examiners, professor Michael Sears and two others, for taking their time to examine my thesis and provide helpful and encouraging comments. In particular, professor Michael Sears has shown genuine interest in my research work throughout my studies. Professor Sears has not only examined my thesis, but also put effort to correct some of the language errors in my thesis. His advice and suggestions have been always invaluable.

I would also like to thank my three wonderful daughters, Nandi, Christine and Marilyn, for always trying to help out in many ways, and most importantly, trying to do their best.

Lastly, I would like to thank my husband, Sheng, for his love and support, who made my years of study possible in the first place.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
ADPCM	Adaptive DPCM
AVIRIS	Airborne visible/infrared imaging spectrometer
BP	Basis pursuit
BPE	Bit plane encoder
CALIC	Context-based adaptive lossless image codec
CCSDS	Consultative committee for space data systems
CFAR	Constant false-alarm rate
DCT	Discrete cosine transform
DPCM	Differential pulse code modulation
DWT	Discrete wavelet transform
EBCOT	Embedded block coding with optimised truncation
EZW	Embedded zerotree wavelet
GAP	Gradient-adjusted prediction
GLA	Generalised Lloyd algorithm
HSI	Hyperspectral image
ICT	Irreversible colour transform
IDC	Image data compression
JPEG	Joint photographic experts group
KLT	Karhunen-Loève transform
M-NVQ	Mean-normalized vector quantization
MP	Matching pursuit
MSB	Most significant bit
MSE	Mean squared error
MSSD	Multi-scale spectral dictionary
NN	Nearest neighbour
ODL	Online dictionary learning
OMP	Orthogonal matching pursuit
PCA	Principle component analysis
PDF	Probability density function
PPI	Pixel purity index
PSNR	Peak signal to noise ratio
RCT	Reversible colour transform
RMSE	Root mean squared error
SAM	Spectral angle mapper

SD	Spectral dictionary
SKA	Square kilometre array
SNR	Signal to noise ratio
SPECK	Set partitioned embedded block
SPIHT	Set partitioning in hierarchical trees
SRSD	Sparse representation spectral dictionary
SVD	Singular value decomposition
SVM	Support vector machine
VQ	Vector quantization

Chapter 1

Introduction

In signal and image processing, finding an appropriate model for the data is the heart of the problem. Sparsity based models, or simply sparsity models, have been widely adopted as the choice of models in this topic. Using sparsity models, many types of signals such as still image, video and acoustic can be sparsely represented through linear transforms. These models assume that the signal or image vectors, usually lie in a combination of subspaces, where each subspace is spanned by a few elements from a set of basis vectors. Sparsity models have received significant attention in image processing in a variety of applications, including image compression, denoising, classification, to name a few. The well known digital image coding standard joint photographic experts group (JPEG), and its successor JPEG 2000 are both based on transforms that lead to sparsity.

A commonly adopted transform coding strategy is to apply a linear transform to an image vector $\mathbf{x} \in \mathbb{R}^M$ by taking the multiplication of the image with a set of *basis vectors* $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_L\}$, where $\mathbf{w}_i \in \mathbb{R}^M$. We may express this transform in vector notation as

$$\mathbf{b} = \mathbf{W}^{\mathrm{T}}\mathbf{x}, \tag{1.1}$$

where the columns in matrix $\mathbf{W}$ correspond to the basis vectors. The goal in such a transform coding is to find a set of orthogonal basis vectors that span the vector space of input images. The coefficient vectors obtained under these basis vectors usually lead to some criterion of optimality, such as decorrelation and sparseness. Discrete cosine transform (DCT) is an example of such transform coding strategy.

An alternative way of transforming images within the linear framework is using a generative sparsity model introduced by Olshausen and Field [1997]. It is based on learning a dictionary $\mathbf{D}$ using a set of training data. The learnt dictionary is utilizedchecks for sparsely representing the input images or signals. This type of sparsity models are often referred to as *sparse representation* in the literature.

An image $\mathbf{x}$, using sparse representation, is modelled in terms of the linear superposition of a set of vectors $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K\}$, commonly termed as *atoms*.

$$\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}, \tag{1.2}$$

where the atoms in $\mathbf{D}$ correspond to a set of learned vectors, and $\boldsymbol{\alpha} \in \mathbb{R}^K$ is the coefficient vector. In contrast to the transform in (1.1), the vectors in dictionary $\mathbf{D}$ are not necessarily orthogonal to each other, and it requires custom algorithms for finding the coefficient vector $\boldsymbol{\alpha}$. Specifically, if the dictionary $\mathbf{D}$ is overcomplete, which is true when the number of atoms in $\mathbf{D}$ exceeds the dimensionality of the input vector, then the solution for $\boldsymbol{\alpha}$ is not unique.

An image is always associated with a set of data and visually interpretative entities. The data set associated with an image can be understood as an array with each entry being an intensity value and referred to as a pixel. A typical colour image has three channels (or bands) and each of them conveys information at different electromagnetic wavelengths. In fact, the human visual system needs only three wavelength channels, i.e., red, green, and blue (RGB), to see colours [Geladi et al. 2007]. However, there are many imaging techniques that are able to acquire more than three bands, and this number is constantly growing. Hyperspectral imaging is one among such techniques. It is a spectral imaging modality that obtains environmental and geographical information by imaging ground locations from airborne or spaceborne platforms. It is also frequently applied in laboratories and hospitals. Hyperspectral imaging typically uses hundreds of contiguous spectral bands that are regularly spaced from infrared to ultraviolet spectrum. For example, the airborne visible/infrared imaging spectrometer (AVIRIS) delivers calibrated images of the upwelling spectral radiance in 224 contiguous spectral bands with wavelengths from $0.4\mu m$ to $2.5\mu m$ [NASA JPL 2015]. The image data results from hyperspectral imaging is viewed as a three-dimensional (3D) data cube.

1.1 Problem Motivation

Modern hyperspectral sensors have high spectral resolutions covering up to several hundred bands. The transmission and storage of the resultant large volume of data collected by such sensors mean that data compression is an essential feature of the systems incorporating hyperspectral sensors. Hyperspectral data compression has received particular interest in hyperspectral image processing and analysis due to several reasons. Firstly, the volume of such data is commonly enormous. Secondly, there is a significant amount of redundancy in the spectral domain. Thus, by designing efficient compression schemes, high compression ratio can be achieved without loss of significant information. Thirdly, although a hyperspectral image can be viewed as a 3D image cube, it does not necessarily lead to an efficient compression technique by simply extending the existing two-dimensional (2D) image compression methods. For 2D image compression, exploiting the 2D spatial redundancy is the key. When it comes to 3D hyperspectral image cube, the spectral domain often demands distinctive approaches in reducing the redundancy than those for spatial domain. Lastly, the compression performance of hyperspectral data is evaluated not only in terms of compression ratio, but also information preserving. In fact, the latter is often more important than the former. Unlike the conventional images with high spatial resolutions, hyperspectral images have relatively lower spatial resolutions. A pixel in a hyperspectral image may have mixed spectra. It is crucial to preserve such information in data compression for further data analysis in various applications, such as target detection and anomaly detection. For this purpose, spectral compression and spatial compression should be decoupled [Chang 2013]. One might argue that lossless compression should be preferred over the lossy one for the purpose of information preserving. However, lossless compression only allows very limited compression ratios, and the abundant spectral and spatial redundancies in hyperspectral image can not be adequately reduced. As a result, storage and bandwidth could be wasted.

Compression of still images has been a very active research and application field. It offers many contributions in the literature, and a number of standard image compression schemes exist. A representative standard would be the JPEG 2000, which is a general purpose, wavelet transform based compression framework. By general purpose, we mean that the standard does not necessarily adapt to or perform optimally for some specific families of images, including hyperspectral images. It is often necessary for different compression algorithms to be developed for specific types of images. Previous investigations of hyperspectral data compression include differential pulse code modulation (DPCM) or predictive coding based methods [Roger and Cavenor 1996; Wu and Memon 1997], vector quantization (VQ) based methods [Kaukoranta et al. 1999; Qian 2004; Rizzo et al. 2005], and transform coding based methods [Goyal 2001; Penna et al. 2007].

Transform coding plays an important role in hyperspectral image compression. DCT and discrete wavelet transform (DWT) have been applied successfully in still image compression standards JPEG and JPEG 2000, respectively. Both DCT and DWT offer excellent energy compaction features, which are essential in data compression, in these standards. On the other hand, given a dictionary, an image can be approximated as a linear combination of very few dictionary atoms. That is, in sparse representation domain, the coefficient is sparse and hence, displays energy compaction feature. A natural question is if sparse representation can be applied in data compression. This leads to the primary research question of this thesis.

Recent work on various image processing applications show that sparse representation based methods could lead to state-of-the-art results. These include but not limited to vision tasks such as image classification [Bahrampour et al. 2016], image denoising, deblurring, inpainting [Elad and Aharon 2006; Elad et al. 2010], image restoration [Mairal et al. 2008a]. There are also recent works on applying sparse representation to process some specific types of images, such as facial image [Bryt and Elad 2008], video [Guha and Ward 2012] and medical imaging [Li et al. 2012]. In addition, the application of sparse representation has been widely extended to hyperspectral image processing tasks such as classification [Chen et al. 2011a; Soltani-Farani et al. 2013], target detection [Chen et al. 2011b], denoising [Zhao and Yang 2015; Lu et al. 2016], anomaly detection [Xu et al. 2016], and hyperspectral unmixing [Iordache et al. 2011 2014]. Compared with these, there is relatively little work on applying sparse representation to data compression, and in particular hyperspectral data compression.

Motivated by the recent advances in theoretical, methodological, application aspects of sparse representation, in this thesis, we present an application of sparse representation in the lossy hyperspectral image compression framework. The study is extended further to investigate hyperspectral image classification using sparse representation.

The study in this thesis is not only relevant to hyperspectral imagery, it certainly contributes to the broader communities of data compression, image and signal processing. For instance, they could be the ones in the exciting square kilometre array (SKA) project [SKA 2015]. The SKA project is an international effort to build the world's largest radio telescope. It aims at enabling astronomers to monitor the sky in unprecedented detail and survey the entire sky thousands of times faster than any systems currently in existence [SKA SA 2015]. Signal and image processing is an integral part in the SKA project to produce high resolution radio images and to analyse them. The central data processing elements in the SKA project will need to process and transmit the data received by thousands of telescopes. The input data rates are expected to be in the hundreds of gigabits per second and need to be processed in soft-realtime. One of the main challenges in this scenario will be the development and deployment of sophisticated radio frequency interference mitigation, calibration and imaging algorithms. The current sparsity based models, including the sparse representation, adapted to the high dimensional data such as hyperspectral imagery would provide substantial insights to the relevant tasks in SKA project.

1.2 Problem Statement and Research Objectives

The primary research questions in this thesis are:

1. How sparse representation can be effectively utilized in hyperspectral image compression framework?
2. If sparse representation is employed in the hyperspectral image compression, are the sparse coefficients of hyperpsectral image pixels applicable in the subsequent hyperspectral image processing tasks, such as classification?

Problem Statement: Based on the research questions above, the problem to be addressed in this thesis concerns *the applications of sparse representation to i) hyperspectral image compression, and ii) hyperspectral image classification.*

The aim of this thesis is to effectively apply sparse representation to hyperspectral image compression and hyperspectral image classification. We set the specific research objectives as follows.

1. Development of a hyperspectral image compression framework that deploys sparse representation.
2. Implementation of a prototype of the proposed compression framework.
3. Dictionary learning for effective exploitation of the spectral and spatial correlations in a hyperspectral image.
4. Evaluation and comparison of the proposed compression method to the existing state-of-the-art hyperspectral image compression approaches, which include JPEG 2000 standard.
5. Development and implementation of a hyperspectral image classification scheme that incorporates sparse representation.
6. Evaluation and comparison of the proposed classification approach to other representative hyperspectral image classification methods including sparse representation based ones.

A number of problems may be subsumed under the primary problems. These include:

- A dictionary is a key element in sparse representation. How do we alleviate the overhead of transmitting the dictionary in a data compression framework?
- When we pursue reducing the data size in lossy data compression, it is also important to preserve information. How do we achieve this goal in sparse representation based compression scheme?

1.3 Overview of Methodology

In this section, we give an overview of our methodology for applying the sparse representation to hyperspectral image compression and classification. The proposed lossy hyperspectral image compression framework consists of two integral parts: encoder and decoder. Sparse representation is employed in the encoder in two aspects, that is dictionary learning and sparse coding.

Dictionary Learning: Dictionaries are learned to exploit the redundancy of a hyperspectral image in spectral domain, and such a dictionary is termed as spectral dictionary (SD). Online dictionary learning algorithm [Mairal et al. 2008a] is applied for the training process. Online dictionary learning is a stochastic gradient descent based approach that picks a random training sample at each iteration and updates the objective function on the basis of this example only. It optimizes the expected cost of the objective function rather than the empirical cost. Airborne or spaceborne hyperspectral images are often obtained over the same or similar geological locations. This makes it possible to learn a base dictionary offline. By ‘offline’ we mean the dictionary learning process is not included as a part in the encoder. The base dictionary can be used in the sparse representation of hyperspectral images from the similar geological locations. To improve upon the representative capacity of the base dictionary, a dictionary update step is included in the encoder. This approach has two advantages. One is to improve the adaptivity of the dictionary with moderate computational cost. The other is reducing the cost of transmitting the dictionary, as we only need to transmit the difference between the base and updated dictionaries.

The spectral dictionaries discussed above can only be utilized for exploiting the spectral redundancy in hyperspectral images. In order to exploit the spatial redundancy as well, we take an approach where the dictionaries are trained in the multi-scale spectral domain, and such a dictionary is termed as MSSD. Specifically, the DWT is performed in the spatial domain to exploit the spatial redundancy in hyperspectral images. This is followed by training multi-scale spectral dictionaries in the resulting wavelet spectral domain.

Sparse coding: Using a learned dictionary and sparse coding, the hyperspectral image pixels are transformed into sparse coefficients. Being the main information to be transmitted to the decoder, these sparse coefficients are represented as a sparse matrix. Subsequently, an efficient representation of a sparse matrix, called index-value pair, is applied. The index-value pairs of the sparse coefficients have two types of information, one is position index, and the other is coefficient value. As such, these are treated in separate manners in the encoding stage.

Besides the sparse coefficients, the dictionary and the mean values of the hyperspectral image pixels, also need to be transmitted. The information with non-integer values are quantized using uniform scalar quantizers. Huffman coding is applied in the entropy coding to formulate the final bit stream.

The hyperspectral image classification in this thesis is a supervised classification approach that utilizes sparse representation. The proposed method starts with learning a dictionary that induces sparsity in the representation coefficients and minimizes the reconstruction error. In order to incorporate the contextual information into the spectral sparse coefficients, we enforce smooth variations in the neighbourhoods of pixels by applying a local filter on sparse coefficients. These contextual sparse coefficients are then used as feature vectors for two popular classifiers, namely SVMs and nearest neighbour (NN) classifier, for their robustness and simplicity.

A highlight of the methodology in this thesis is that the proposed compression and classification approaches are evaluated not only in terms of relevant performance criteria that are commonly used in the literature, but also are compared to some of the state-of-the-art hyperspectral image compression or classification methods.

1.4 Contribution of The Thesis

The successful applications and developments of sparse representation in various signal and image processing tasks, including hyperspectral image (or data) processing, show the effectiveness of this particular sparsity based model. In contrast, there is little work done in respect of the application of sparse representation to data compression. Theoretically, however, due to its sparsity inducing nature, sparse representation is suitable to be applied in data compression. This thesis is such an attempt to incorporate sparse representation with hyperspectral image compression. The experimental results show that the proposed approach is comparable to some of the state-of-the-art hyperspectral image compression methods. This shows that sparse representation is indeed a very effective method to exploit the spectral redundancy in hyperspectral images.

The sparse representation based hyperspectral image classification approach reveals the effectiveness of sparse coefficients in the discriminative tasks such as classification. While the work in this regard takes mostly standard approaches, it presents a much simpler way of applying sparse representation to the concerned task and achieves similar or better results. In particular, the dictionary learning and sparse coding may take the same approach in both the compression and classification of hyperspectral images. Under such scenario, it becomes feasible in the proposed hyperspectral image compression framework that the side information, including the dictionary, can be entirely excluded in the final bit stream. Thus, the major challenge of transmitting the dictionary can be alleviated.

So far, some of the results of this thesis are either published or accepted for publication. They are included in the following list.

1. H. Wang and T. Celik, "Sparse representation based lossy hyperspectral data compression," *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS),* pp. 2761-2764, July 2016, Beijing, China.
2. H. Wang and T. Celik, "Sparse representation based hyperspectral data processing: lossy compression," *IEEE Journal of Selected Topics in Applied Earth and Remote Sensing,* 10(5), pp. 2036-2045, May 2017.
3. H. Wang and T. Celik, "Sparse representation based hyperspectral image classification," *Signal, Image and Video Processing* (2018). https://doi.org/10.1007/s11760-018-1249-1.

1.5 Notation

In This Thesis , constant scalar values are denoted by upper-case letters, and variables by lower-case ones. Vectors are denoted by bold lower-case letters, and matrices by bold upper-case ones.

1.6 Organisation of The Thesis

The rest of this thesis is organised as follows. Chapter 2 gives an overview of some of the principles of hyperspectral imaging sensors, the methods of data acquisition, and the characteristics of the data acquired. It also gives a brief discussion of some common tasks and approaches in hyperspectral image processing.

Chapter 3 presents a review of sparse representation, where its theoretical background, relevant existing algorithms for dictionary learning and sparse coding are studied.

The general topics of data compression are discussed in Chapter 4, with the focus on hyperspectral image compression. These include entropy coding, predictive coding, quantization, and transform coding. Brief overviews of popular hyperspectral image compression methods, as well as the latest image compression standards from JPEG and consultative committee for space data systems (CCSDS) are also included in this chapter.

Chapter 5 provides the details of the proposed lossy hyperspectral image compression framework based on sparse representation. It also discusses some related work in data compression based on sparse representation. Further, it presents the evaluation of the proposed method including the experimental setup, results, and discussions.

Chapter 6 introduces the proposed sparse representation based hyperspectral image classification method. It includes the details of related work in this topic, the evaluation and comparison results of various hyperspectral image classification methods, and the related discussions.

Finally, Chapter 7 concludes this thesis and proposes some possible future work.

Note that efforts are taken to write each chapter independently, in the hope that they are self contained. Thus, minor repetitions of some contents are present in Chapters 3, 5, and 6.

Chapter 2
Hyperspectral Imaging

Remote sensing of the Earth provides a vital form of information not easily acquired by conventional surface observation. The advances in aviation and space exploration inevitably lead to the integration of these technologies with remote sensing. Imaging spectroscopy, a major technique for remote sensing, is capable of acquiring images in contiguous, dozens or hundreds of spectral bands throughout the visible and solar-infrared intervals of the electromagnetic spectrum. The resulting image data sets are referred to as *hyperspectral imagery* or simply *hyperspectral image*. Hyperspectral imagery characterizes the reflection of light against objects. It uses the fundamental principles of the interrelations among colour, frequency and wavelength of natural light [Borengasser et al. 2007]. *Multispectral imagery*, on the other hand, usually contains a small number of relatively broad wavelength spectral bands [Borengasser et al. 2007; Chang 2007b]. In addition, multispectral imaging systems often have their spectral bands widely irregularly spaced, while hyperspectral imaging systems have spectral bands that are adjacent to each other and regularly spaced.

Remote sensors allow us a vantage point to obtain not only vertical views of the Earth's land surface, but also information covering large areas quickly. This raises the question of dealing with large volumes of data, even with relatively low spatial resolutions of remote sensing data. Higher spatial resolution results in larger quantities of data, more sophisticated sensors and control systems, increased requirements for storage and downlink bandwidth. These motivated the consideration of ways to collect information via spaceborne or airborne platforms from image data with lower spatial resolutions. This resulted in the concept of using spectral measurements of a pixel to analyse the type of ground cover a pixel represents, instead of using spatial variations and the conventional image processing techniques that rely on higher spatial resolution [Landgrebe 1999].

Hyperspectral imaging spectrometers are capable of capturing images that contain both spatial and spectral information at relatively fine resolutions. The resulting data was shown to be useful in identifying different materials, and producing thematic maps that display the spatial distribution of these materials. This property led to broad applications of hyperspectral imaging in different problems, adapting the processing used to extract relevant information. In remote sensing, the applications of hyperspectral imaging cover a variety of fields such as land use monitoring [Bachmann et al. 2002], agriculture [Haboudane et al. 2004], urban mapping [Benediktsson et al. 2003 2005], coastal environment and water monitoring [Corson et al. 2008], geology [Kruse et al. 2003], mine detection [Zare et al. 2008], chemical agent detection [Kwan et al. 2006], and others [Meer and de Jong 2001; Eismann et al. 2009; Nasrabadi 2014, and references therein]. In general, these applications perform the tasks of detection, recognition, or classification of anomalies or targets, or characterization of background scenes. Beyond remote sensing, hyperspectral imaging has also found applications in various fields such as pharmaceuticals [Ravn et al. 2008], medical [Harvey et al. 2002], and food quality and safety [Elmasry et al. 2012].

Hyperspectral images contain a wealth of information. Processing and analysing this type of data, such as information extraction, requires an understanding of what properties of land surface materials are captured and how they are measured by hyperspectral sensors. For this purpose, in the following section, we investigate briefly some of the principles of hyperspectral imaging sensors, methods of data acquisition and the characteristics of the data acquired. We also discuss some common tasks and approaches in hyperspectral image processing before we summarize this chapter.

2.1 Imaging Spectroscopy

Spectroscopy is the study of light as a function of wavelength that has been emitted, reflected, or scattered from a solid, liquid, or gas [Clark 1999]. As photons enter a material, reflection, transmission and absorption all occur. Material surfaces are also capable of emitting photons under certain conditions. The varying characteristics of photon absorption, reflection or emission from different materials provide us with information about the materials. When applied to the optical remote sensing, spectroscopy deals with the spectrum of sunlight that is absorbed, reflected or scattered by materials on the Earth's surface. Based on spectroscopy, instruments called *spectrometers* are developed to measure the light energy reflected from a test material. An optical refraction device such as a prism in a spectrometer splits the light into many narrow, consecutive wavelength bands. The energy in each one of these bands is measured by a separate detector. By using hundreds or thousands of detectors, spectrometers can make spectral measurements of bands as narrow as several nanometres.

As a type of remote sensing instrument, imaging spectrometers are passive sensors. Passive sensors detect natural energy that is emitted or reflected by the objects they observe. Reflected sunlight is the most common source of radiation measured by passive sensors. On the other hand, active sensors provide their own source of energy to illuminate the objects. They emit radiation towards the targets, and then detect and measure the reflected or backscattered radiation from them [NASA 2018].

Imaging spectroscopy is a technique for obtaining a spectrum at each position of a large array of spatial positions, so that any one spectral wavelength can be used to make a recognizable image [Clark 1999]. In remote sensing literature, imaging spectroscopy is also known as *imaging spectrometry*, *hypespectral imaging*, or *ultraspectral imaging*. The concept of imaging spectroscopy in the case of satellite remote sensing is illustrated in Figure 2.1.

Hyperspectral images are produced by *imaging spectrometers*. Most imaging spectrometers used are aboard air- or space-based platforms. For example, the NASA's AVIRIS is flown on the ER-2 jet at 20km above the ground. These imaging spectrometers are developed to measure spectra as images in some or all of the $400 - 2500$ nm electromagnetic spectral region. As the analysis of hyperspectral data in this part of the spectrum mainly deals with the reflectance nature of solid and liquid materials. These hyperspectral images are typically arranged as a 3-dimensional data cube as shown at the centre of Figure 2.2. Hyperspectral data can be viewed as a set of *pixel vectors*, each consists of a group of spectral components for the same spatial location. Each component of a pixel vector corresponds to the received energy at a particular wavelength. The spectra of a hyperspectral pixel vector is shown at the left of Figure 2.2. Alternatively, we

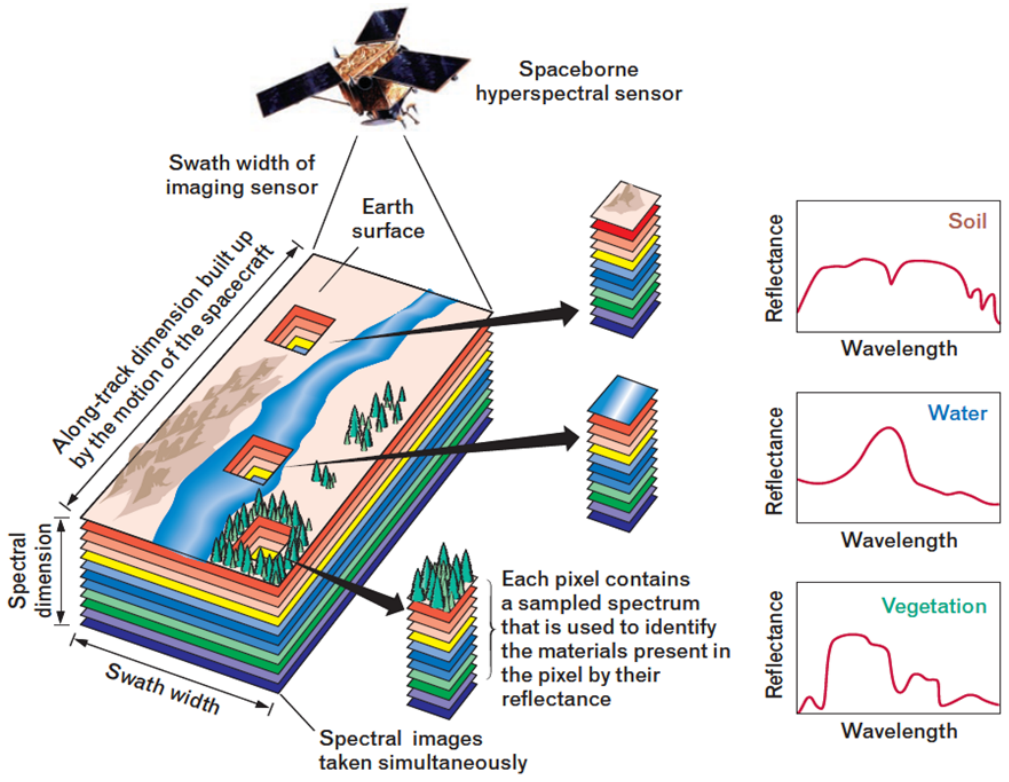

Spaceborne hyperspectral sensor
Swath width of imaging sensor
Earth surface
Along-track dimension built up by the motion of the spacecraft
Spectral dimension
Swath width
Spectral images taken simultaneously
Each pixel contains a sampled spectrum that is used to identify the materials present in the pixel by their reflectance
Reflectance
Wavelength
Soil
Water
Vegetation

Figure 2.1: The concept of imaging spectroscopy (Source: [Shaw and Burke 2003]). An airborne or spaceborne sensor samples multiple spectral wavelengths over a ground based scene. In the resulting image, each pixel contains a spectral measurement of reflectance. The spectral variance of three types of materials is also shown in the graph.

can view the hyperspectral data cube as being composed of a group of co-registered 2D images with each of them corresponding to a particular wavelength. These images are often referred to as *band images* or simply *bands*. An example band image is shown at the right of Figure 2.2.

2.1.1 Reflectance and Radiance

Through imaging spectroscopy, we want to obtain a measurement, called *spectral reflectance*, which reveals the property of a particular material. Reflectance is the ratio of the amount of energy leaving a target to the amount of energy striking the target. It is a fractional number between 0 and 1. Reflectance varies with wavelength for most materials because light at a certain wavelength is reflected, scattered or absorbed differently. Figures 2.3 and 2.4 show the spectra of different materials. The former figure shows three common Earth surface materials, namely green vegetation, dry vegetation, and soil, and the latter shows the spectra of three materials, namely, bricks, starch (corn), and limestone. Each spectral curve shows the differing levels of absorption of light energy of the corresponding material. For example, vegetation has higher reflectance (or lower absorption) in the near infrared range, and lower reflectance of red light than soil. In the case of green and dry vegetations, the spectral curves are generally similar due to the two materials are made of the same basic components. The shape of the spectral curve, the position and strength

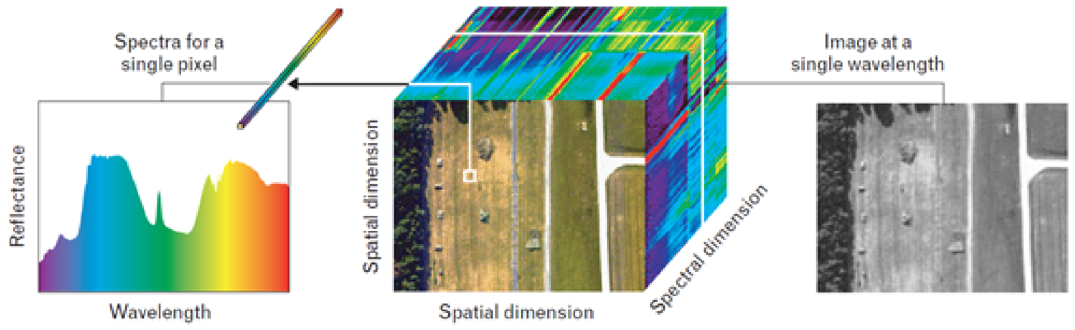


Figure 2.2: The 3D structure of hyperspectral cube (Source: [Manolakis et al. 2003]).

of absorption bands are often used in distinguishing among different surface materials.

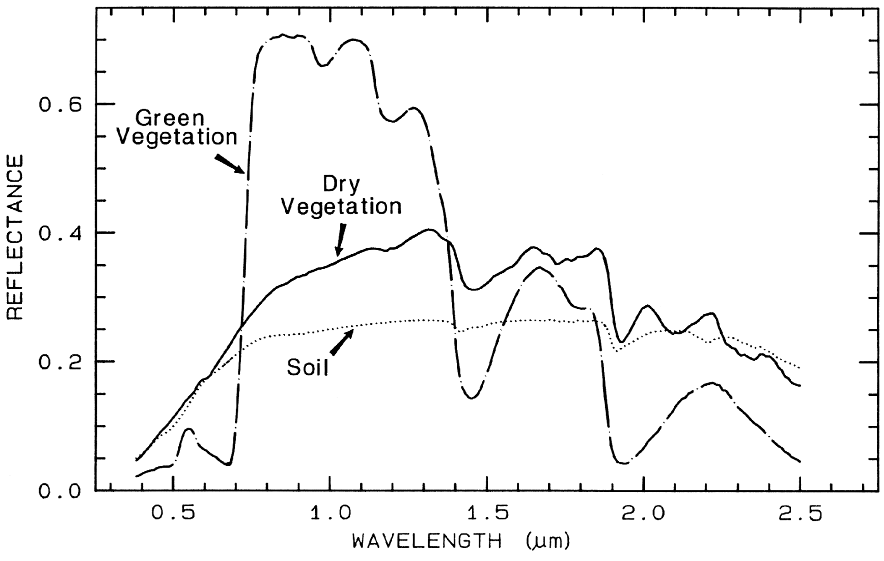


Figure 2.3: Reflectance spectra of Soil, Green and Dry vegetations (Source: [Clark 1999]).

Radiance typically describes the amount of light the imaging spectrometer detects from the object being observed. This is the quantity directly measured by a spectrometer. Radiance indicates how much of energy emitted or reflected from a surface is received by an imaging spectrometer looking at the target at a certain angle. While radiance reflects the total amount of emission or reflectance, *spectral radiance* indicates the amount of radiance at a single wavelength. Radiance data contains information relevant to the material composition of each pixel. However, this information is generally perturbed in the collected data by various aspects. These include environmental factors such as weather, solar illumination, downward and upward atmospheric passages, and adjacency effects, as well as sensor characteristic influences. These factors need to be compensated effectively through the combination of sensor calibration [Green et al. 1990] and atmosphere normalization [Green et al. 1993]. Several commonly used strategies for converting AVIRIS radiance data to reflectance data is discussed in [Farrand et al. 1994].

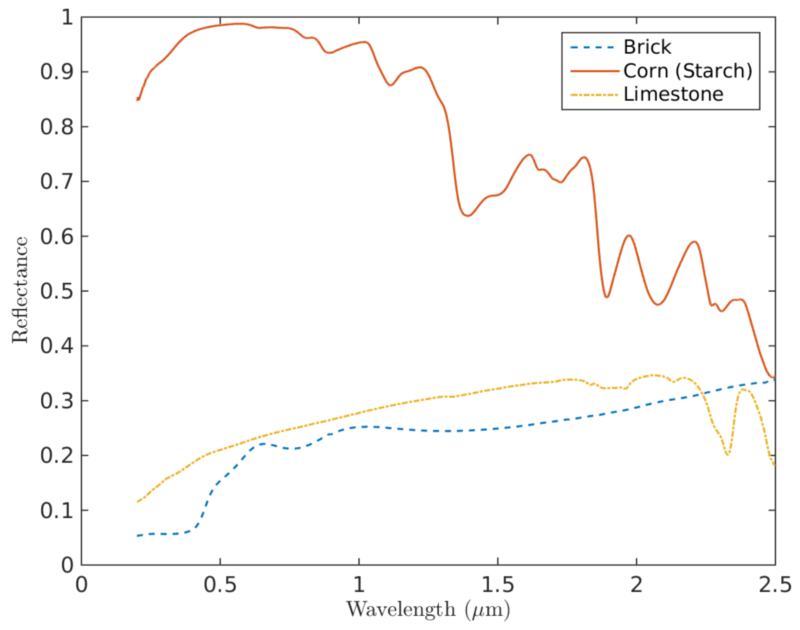


Figure 2.4: Reflectance spectra of brick, starch (corn) and limestone.

2.1.2 Sensors

The critical part of hyperspectral imaging is the sensor. The imaging process in hyperspectral sensing involves the focusing of light from a small region of the Earth's surface to a given detector unit that forms a pixel in the resulting image [Chang 2007a]. The spatial resolution of the image is determined by the sizes of relevant hardware elements such as the detector and lens. The sensors form an image by arranging the pixels in two-dimensional array. Each pixel has a number of spectral components. Thus, a hyperspectral image is often described as a data cube. The two spatial dimensions include along-track dimension (column) and its orthogonal dimension (row or scan line) which is sometimes referred to as *swath*. The along-track dimension is built up by the motion of the platform carrying the sensor, and the swath is the width of the field-of-view covered by the sensor. The spectral dimension consists of a number of spectra at varying wavelengths (see Figure 2.1).

Modern imaging spectrometers differ in the methods they collect data. These methods focus on two primary aspects, namely, spatial scanning and spectral selection. From the spatial point of view, there are a number of basic ways the sensors form (2D) images. For different approaches, the focal plane complexity and platform stability requirements differ for the sensors [Schaepman 2009; Chang 2007a]. In the following, we briefly describe the three approaches for spatial scanning. In general, whiskbroom imaging spectrometer collects series of pixels, pushbroom scanners series of lines, and filter wheel cameras series of monochromatic images.

Whiskbroom scanners cover the field-of-view by a mechanized angular movement of a scanning mirror sweeping from one edge of the swath to the other, or by the mechanical rotation of the sensor. The scanner view of the field at a moment is a single spatial pixel and its associated spectral components. The image lines are collected using an across-track scanning and the entire image is obtained using the forward movement of the carrier platform.

Pushbroom scanners acquire a series of one-dimensional samples orthogonal to the platform line of flight, with along-track spatial dimension constructed by the forward movement of the platform. The spectral component is obtained by dispersing the incoming radiation onto an area array. This type of scanners do not require the use of moving mirror because a linear array of detectors image a line of image across the field of view simultaneously.

Filter wheel cameras are opto-mechanical sensors that change the spectral sensitivity of various channels using a turnable filter wheel in the optical path. Such a camera relies on the use of two-dimensional detector array in the focal plane to acquire a monochromatic frame of the field-of-view at one time. The spectral components are obtained by rotating the filter wheel to different band pass filters.

On the other hand, the spectral measurements are accomplished by several techniques including *prism*, *grating*, *filter wheel*, *interferometer*, and *acousto-optical tunable filter* or *liquid crystal tunable filter*. Using prism, the incident light is diffracted in different directions, depending on the wavelength. The linear detectors are then positioned at appropriate locations to the prism to sample the varying spectra. Grating works similar to that of prisms by dispersing the light spatially, except that the light is reflected rather than diffracted through the spectral selection elements. For filter wheel approaches, interference filters are arranged around the edge of a wheel, which are rotated in front of the detector to collect data at various wavelengths. The interferometer systems collect interferograms which must be converted to optical spectrum. Finally, the acousto-optical and liquid crystal tunable filters work on the principle of selective transmission of light through materials in which acoustic waves are passed or a varying voltage is applied [Chang 2007a].

The recent developments in hyperspectral imaging spectroscopy mostly concentrated on airborne spectrometers covering visible to near-infrared, and shortwave-infrared range of spectra. For an example, the AVIRIS [Green et al. 1998], designed and operated by NASA's Jet Propulsion Laboratory in California, USA, has been used since late 1980's in a large number of scientific experiments and field campaigns. Deployed on NASA's ER-2 or WB-57, flying up to 20km above the ground, AVIRIS is able to collect data with 20m spatial resolution over an estimate of 11km swath. It can also be applied at lower altitude for finer ground resolution. Among the spaceborne spectromters, Hyperion, on board NASA's Earth Observing-1 (EO-1) spacecraft, has been one of the main providers of space based hyperspectral data. To improve the amount and quality of space based hyperspectral data, the number of space borne projects are gradually increasing. The environmental mapping and analysis program (EnMAP) German imaging spectroscopy mission is one such program. EnMAP is expected to become a key system in the field of spaceborne imaging spectroscopy in potential co-existence with other comparable missions around the world [Guanter et al. 2015].

2.2 Hyperspectral Image Processing

Extracting information in hyperspectral data is an important subject. Various techniques are developed recently to classify, detect, identify, quantify, and characterize the features and objects using the hyperspectral images [Chang 2007a]. Although the applications of hyperspectral imaging cover numerous fields, some application specific tasks are dealt with commonly. One such task is *dimensionality reduction*. Hyperspectral sensors are designed to capture spectral information in fine resolutions so that any narrow spectral features are adequately represented. Some sensors can achieve high resolutions in spatial feature as well. As an important task, dimensionality reduction tries to reduce the redundancies in both spectral and spatial domains without loss of significant information. It is implemented either explicitly or implicitly in a specific application. The most widely used approaches for dimensionality reduction is the principle component analysis (PCA) or Karhunen-Loève transform (KLT). Other common hyperspectral image processing tasks include the following:

- Endmember extraction and spectral unmixing involves identifying the endmembers in a hyperspectral scene, and estimating the fractional abundance of each endmember in each pixel spectrum;
- Target and anomaly detection;
- Classification is a task of assigning a class label to each pixel vector in a hyperspectral image.

A comprehensive treatment of hyperspectral image processing is beyond the scope of this thesis. Interested readers are referred to the topics, including hyperspectral image endmember extraction, unmixing, detection, classification and compression, covered in [Chang 2013; Camps-Valls et al. 2012], and some recent publications on other topics such as hyperspectral image denoising [Chen and Qian 2011; Zhao and Yang 2015], the estimation of noise in hyperspectral images [Gao et al. 2013; Mahmood et al. 2017], and hyperspectral image fusion [Yokoya et al. 2012; Akhtar et al. 2014; Wei et al. 2015]. The main focus of this thesis is hyperspectral image compression and classification, which we discuss in detail in Chapters 4, 5, and 6. In the following sub-sections, we briefly describe a few selected common tasks in hyperspectral imaging applications. These include *endmember extraction*, *spectral unmixing*, *target detection*. The main reason these tasks are selected is that they share closely related techniques.

2.2.1 Endmember Extraction and Hyperspectral Unmixing

A basic approach to analysing hyperspectral image is to match each pixel spectrum to one of the reference spectra in a spectral library. The approach is effective in the case where the scene of interest includes areas of pure materials that have corresponding reflectance spectra in the reference library. However, most hyperspectral scenes include pixels of mixed materials. This is possibly due to the low spatial resolutions of most hyperspectral sensors, or the mixture of distinct materials that are formed naturally. The resulting image spectrum may resemble multiple reference spectra. In general, the *endmembers* represent the pure materials present in the image, and the set of abundances at each pixel to indicate the percentage of each endmember that is present [Bioucas-Dias et al. 2012]. In hyperspectral imaging, endmember extraction is the process of selecting a group of pure signature spectra of materials that are present in the hyperspectral scene.

Many techniques are available for endmember extraction with or without the assumption of existence of pure pixels in a scene [Plaza et al. 2012, and references therein]. A popular algorithm based on the geometrical approach is the pixel purity index (PPI) [Boardman et al. 1995]. PPI projects every spectral vector to *skewers*, defined as a set of random vectors. The vectors that are extremes for each skewer are counted. After a certain number of repeated projections onto different skewers, the pixels with highest count scores are the purest ones. N-FINDER [Winter

1999] is another widely used approach for endmember extraction. This method is based on the fact that in the spectral domain, the simplex formed by the purest pixels has the largest volume. The algorithm finds the set of pixels that define a simplex with the largest volume.

The endmembers are used to decompose each pixel's spectrum to identify the relative fractional abundance of each endmember material through the process called *spectral unmixing*. Spectral unmixing involves decomposing a pixel spectrum into its pure component endmember spectra, and estimating the abundance of each endmember in the pixel. Many unmixing techniques [Keshava and Mustard 2002; Bioucas-Dias et al. 2012; Ma et al. 2014] have been developed using approaches such as geometrical [Craig 1994; Boardman et al. 1995; Winter 1999], statistical [Moussaoui et al. 2008; Dobigeon et al. 2009; Bioucas-Dias and Nascimento 2012], and sparse regression based [Iordache et al. 2011 2014]. Unmixing approaches usually start with modelling the underlying physics foundation of hyperspectral imaging, and then use these models to perform the inverse operation. Two common such models are linear mixing model and nonlinear mixing model. Compared to the linear mixing model, nonlinear unmixing approaches are more complex, relying on the scene parameters that are hard to obtain. Notable nonlinear unmixing approaches include kernel-based methods [Broadwater et al. 2011], and machine learning methods [Plaza et al. 2009b]. On the other hand, the linear mixing model is well studied and has become a standard technique. It assumes that each spectral vector can be approximated by a linear combination of the endmembers weighted by their corresponding fractional abundances. That is, given a set of endmembers as the columns in matrix $\mathbf{\Phi} \in \mathbb{R}^{M\times P}$, a spectral vector $\mathbf{x} \in \mathbb{R}^M$ is modelled by

$$\mathbf{x} = \sum_{j=1}^{P} \mathbf{\Phi}_j \beta_j + \boldsymbol{\epsilon}, \tag{2.1}$$

where $\mathbf{\Phi_j} \in \mathbb{R}^M, j = 1, 2, \cdots, P$, is the jth column (or endmember) in $\mathbf{\Phi}$, $\boldsymbol{\beta} = [\beta_1\, \beta_2 \cdots \beta_P]^{\mathrm{T}} \in \mathbb{R}^P$ where β_j represents the fractional abundance of jth endmember, and $\boldsymbol{\epsilon} \in \mathbb{R}^M$ is the model error. The fractional abundances are subject to the constraints

$$\begin{aligned} &\beta_j \geq 0 \quad j = 1, 2, \cdots, P, \\ &\sum_{j=1}^{P} \beta_j = 1. \end{aligned} \tag{2.2}$$

Under the linear mixing model, hyperspectral unmixing process generally consists of a number of stages, namely, atmospheric correction, data reduction, unmixing, and inverse operation. Atmospheric correction converts radiance data to reflectance. The second stage performs dimensionality reduction to improve the algorithm performance, reduce the computational complexity and data storage. During the unmixing step, the endmembers in the scene are identified, as well as their abundances at each pixel. Given a spectral vector and the identified endmembers, the inversion operation involves solving a constrained optimization problem which minimizes the residual between the given spectral vector and a linear combination of the endmembers.

2.2.2 Target Detection in Hyperspectral Images

Target detection aims at labelling every pixel in a hyperspectral image as target or background. Typically, such applications seek to identify a relatively small number of targets with fixed shape or spectrum in a scene. Background refers to all the non-target pixels. Targets are usually artificial objects with spectra differing from the spectra of natural background pixels. The target detection problem is often formulated as binary hypothesis test with two possible outcomes: H_0 – *background only* or *target absent* and H_1 – *target and background* or *target present*. Given an observed spectrum $\mathbf{x} \in \mathbb{R}^M$ of a test pixel, target detection approaches are frequently based on likelihood ratio (LR) test, which is given by

$$y = d(\mathbf{x}) = \frac{p(\mathbf{x}|\mathrm{H}_1)}{p(\mathbf{x}|\mathrm{H}_0)}, \tag{2.3}$$

where scalar y is the ratio of the conditional probability density functions (PDFs), $p(\mathbf{x}|\mathrm{H}_1)$ and $p(\mathbf{x}|\mathrm{H}_0)$, of $\mathbf{x}$ under the two possible outcomes. If y exceeds a certain threshold, then the target is present. When there are unknown PDFs in (2.3), a widely used approach is to replace them with their maximum-likelihood estimates. The resulting LR is called generalized likelihood ratio (GLR).

Although target detection is essentially a binary classifier, the methods for classification are not applicable to target detection. In a classification task, we expect there are many instances for each class. The result of a classification is usually clusters of the majority of pixels from the same class with occasional misclassifications of a negligible number of pixels. In a target detection task, on the other hand, the number of target pixels in a scene is usually too small to support the statistical property of the target class. Moreover, depending on the spatial resolution of a sensor, the target may appear in just a single or a few pixels. Thus, different approaches are employed in the classification and detection applications of hyperspectral images.

Full-pixel target detection When the target of interest occupies the full pixel, i.e., full-pixel target, the detection algorithms are designed according to the variability of target and background spectra. The simplest case is where the statistics of the target and background are known. Assume the normal distribution probability model and the same covariance matrix, $\mathbf{\Gamma} \in \mathbb{R}^{M\times M}$, for both the target and background, the LR detector has the form [Manolakis et al. 2003]

$$y = d(\mathbf{x}) = \mathbf{C}_{\mathrm{MF}}^{\mathrm{T}}\mathbf{x}, \tag{2.4}$$

where

$$\mathbf{C}_{\mathrm{MF}} = k\mathbf{\Gamma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0), \tag{2.5}$$

$\boldsymbol{\mu}_1, \boldsymbol{\mu}_0$ are respectively the mean vectors for target and background, and k is a normalization constant. If the value of y exceeds a certain threshold η, then the target is present. The threshold is determined in a way so that the false-alarm rate is kept constant at a desirable level. This can be achieved by using a constant false-alarm rate (CFAR) [Kelly 1986] threshold-selection processor, which depends on the background statistical model.

In practice, it is often the case that we do not have sufficient statistical information on the target. That is, we have information only on the background. Using the assumption of normal distribution and the same covariance matrix for both the target and background, the detector takes the form

$$y = d(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_0)^{\mathrm{T}}\mathbf{\Gamma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_0), \tag{2.6}$$

which is the Mahalanobis distance of the test pixel spectrum and the mean of the background distribution. The detector in (2.6) is often referred to as *anomaly detector*.

If both the background and target statistics are not available, the mean vector and covariance matrix of the background are approximated using all the pixels in a selected region. The selected region can be chosen globally as the whole hyperspectral image, or locally using a sliding window approach. The performance of the detector is then dependent on the quality of the approximated covariance matrix. The widely used Reed-Xiaoli (RX) anomaly detector [Reed and Yu 1990] takes the adaptive detector approach. It adopts the CFAR property that allows the detector to use a single threshold to maintain a false alarm rate regardless of the background variations at the different locations of the scene [Nasrabadi 2014].

Subpixel target detection For subpixel targets, the spectrum is often a linear mixing of target and background spectra, given that the spectra of the subpixel size object is known. Such subpixel target detection algorithms model the variabilities of target and background spectra using either statistical (unstructured background) or subspace (structured background) approaches.

In unstructured background models, the background, $\mathbf{b} \in \mathbb{R}^M$, is modelled by a multivariate normal distribution with mean zero and covariance matrix $\mathbf{\Gamma} \in \mathbb{R}^{M\times M}$, where we assume the additive noise has been included in the background. The two competing outcomes under this model become

$$\mathrm{H}_0: \quad \mathbf{x} = \mathbf{b} \qquad \text{target absent,} \tag{2.7}$$
$$\mathrm{H}_1: \quad \mathbf{x} = \mathbf{S}\mathbf{a} + \mathbf{b} \qquad \text{target present,} \tag{2.8}$$

where $\mathbf{S} \in \mathbb{R}^{M\times P}$ consists of a set of user specified endmembers that characterizes the target subspace, and $\mathbf{a} \in \mathbb{R}^P$ is the abundance vector. Assuming the same covariance matrix for the two possible outcomes, i.e., the same background, the GLR based detection statistics is obtained as [Manolakis et al. 2003, and references therein]

$$y = d(\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}}\hat{\mathbf{\Gamma}}^{-1}\mathbf{S}(\mathbf{S}^{\mathrm{T}}\hat{\mathbf{\Gamma}}^{-1}\mathbf{S})^{-1}\mathbf{S}^{\mathrm{T}}\hat{\mathbf{\Gamma}}^{-1}\mathbf{x}}{N + \mathbf{x}^{\mathrm{T}}\hat{\mathbf{\Gamma}}^{-1}\mathbf{x}}. \tag{2.9}$$

In (2.9), $\hat{\mathbf{\Gamma}}$ is the maximum-likelihood estimate of $\mathbf{\Gamma}$, and N is the number of available samples. When we are looking for deterministic targets that lie in the data subspace, the (2.9) becomes

$$y = d(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\hat{\mathbf{\Gamma}}^{-1}\mathbf{x}, \tag{2.10}$$

which is essentially the adaptive version of (2.6).

When the background statistics is modelled as a subspace model, a target detector chooses between the following two possible outcomes:

$$\mathrm{H}_0: \quad \mathbf{x} = \mathbf{B}\mathbf{a}_{b,0} + \mathbf{w} \qquad \text{target absent,} \tag{2.11}$$
$$\mathrm{H}_1: \quad \mathbf{x} = \mathbf{S}\mathbf{a} + \mathbf{B}\mathbf{a}_{b,1} + \mathbf{w} \qquad \text{target present,} \tag{2.12}$$

where $\mathbf{B} \in \mathbb{R}^{M\times Q}$ is determined from the data, $\mathbf{a}, \mathbf{a}_{b,0} \in \mathbb{R}^Q$, and $\mathbf{a}_{b,1} \in \mathbb{R}^Q$ are the abundance vectors, and $\mathbf{w} \in \mathbb{R}^M$ is the additive noise. Use of GLR approach results in the detection statistic

$$y = d(\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}}\mathcal{P}^{\perp}_{\mathbf{B}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathcal{P}^{\perp}_{\mathbf{SB}}\mathbf{x}}, \tag{2.13}$$

where $\mathcal{P}^{\perp}_{\mathbf{A}}$ is an operator for any matrix $\mathbf{A}$ defined as $P^{\perp}_{\mathbf{A}} = \mathbf{I} - \mathcal{P}_{\mathbf{A}}$, $\mathbf{I}$ is the identity matrix, $\mathcal{P}_{\mathbf{A}} = \mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$ is the projection matrix onto the column space of matrix $\mathbf{A}$, and $\mathbf{SB} = [\mathbf{S}\ \mathbf{B}]$ is the matrix obtained by concatenating matrices $\mathbf{S}$ and $\mathbf{B}$.

2.3 Summary

From its acquisition, hyperspectral imaging evolves in terms of the quality of the acquired images. This ever growing evolution demands improved techniques for solving the variety of hyperspectral image processing problems. The analysis of hyperspectral images, or any multi-dimensional images from various remote sensing equipments, leads to many applications. These include land use monitoring, environmental monitoring, mining and mineralogy, target and change detection, urban growth monitoring, and crop field identification. In this chapter, we have briefly described hyperspectral imaging spectroscopy in terms of its principle of acquiring images, the characteristics of hyperspectral image data, and some basic hyperspectral image processing techniques.

Chapter 3
Sparse Representation

The problem in the generative model (1.2) is to find the dictionary $\mathbf{D}$ that best accounts for the structure in images. In the probabilistic framework, we want to match as closely as possible the distribution of images arising from our image model $P(\mathbf{x}|\mathbf{D})$ to the actual distribution of images observed in nature. In order to calculate $P(\mathbf{x}|\mathbf{D})$, we need to specify the á priori probability distribution over the coefficients, $P(\boldsymbol{\alpha})$. If we incorporate *sparsity* into $P(\boldsymbol{\alpha})$, we obtain a linear generative model which is referred to as *sparse representation*.

If we compare the two models formulated in (1.1) and (1.2), we can distinguish them as one being *analysis* and the other *synthesis*. Analysis involves the operation of associating with each signal a coefficient vector attached to atoms as in (1.1). Synthesis is the operation of building up a signal by superposing atoms as in Problem (1.2). These two operations can not be viewed simply as being inverse to each other. Note that in (1.2), if the dictionary $\mathbf{D}$ consists of a basis, then $\mathbf{D}$ is a non-singular matrix and invertible. Additionally, if the atoms in $\mathbf{D}$ forms an orthogonal bases, then the inverse of $\mathbf{D}^{-\mathbf{1}} = \mathbf{D}^{\mathrm{T}}$. In sparse representation, we assume that images are composed of features with certain continuum among them instead of being independent of each other. In order to match the rich structure present in an input image, the dictionary $\mathbf{D}$ is usually *overcomplete*. Such a dictionary not only possesses greater robustness in the case of noise, but also allows greater flexibility to match with a particular input structure. On the other hand, the assumption we make in the transform coding strategy, i.e., the model in (1.1), is that images are composed of a certain number (the dimensionality of the input) of linearly independent basis vectors, or features.

The sparse representation modelling of signal has been shown to be a powerful tool in image processing. Such image processing tasks include but are not limited to image compression, denoising, deblurring, inpainting and more. In this chapter, we present a review of sparse representation. Relevant known algorithms for learning a dictionary and sparse coding will be studied.

3.1 Problem Formulation

In signal and image processing, data modelling is a crucial step for performing various operations such as compression and restoration. Sparsity based models are considered viable in this field, and beyond, such as statistics. These models adhere to the rule of sparsity, which implies representing a phenomenon with as few variables as possible [Mairal et al. 2014]. Similarly, in terms of sparse representation, a signal is approximated by the linear combination of a few dictionary elements. Many researchers have contributed towards designing dictionaries that could achieve the goal of sparsity. This leads to various sorts of wavelets [Daubechies 1988; Donoho 1999; Do and Vetterli 2005; Mallat 2008]. Different from this line of traditional approach, Olshausen and Field [1997] proposed learning a dictionary from the training data, instead of using fixed pre-defined dictionaries. Interestingly, the work of [Olshausen and Field 1997] is concerned with the coding strategy of area V1 of the mammalian visual cortex. It is argued in [Olshausen and Field 1997] that there are comparable similarities between the functions of spatial receptive fields of simple cells and the wavelet transform. In particular, the investigation of overcomplete code leads to a deviation of the input-output function being purely linear. This deviation from linearity provides potential explanation of the observed weak form of non-linearity in the cortical simple cells and meaningful insights on their response to natural images.

Using sparse representation, we assume that an image $\mathbf{x} \in \mathbf{R}^M$ is generated by a linear combination of just a few elements from an overcomplete dictionary $\mathbf{D} \in \mathbf{R}^{M\times K}(M < K)$ that contains K columns or prototype atoms. That is, the representation of $\mathbf{x}$ is $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$, or $\mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha}$, satisfying $\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2 \leq \epsilon$, where $\|\cdot\|_2$ is the ℓ_2-norm. The vector $\boldsymbol{\alpha} \in \mathbf{R}^K$ is the sparse representation coefficient of the vector $\mathbf{x}$ [Aharon et al. 2006]. Clearly, $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$ defines a under-determined linear systems with more unknowns than the equations. Therefore, if it has a solution, there are infinitely many $\boldsymbol{\alpha}$s that may match $\mathbf{x}$. In order to narrow the choices to one well-defined solution, the problem is turned into an optimisation problem [Elad 2010, chap. 1]

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\ J(\boldsymbol{\alpha}) \quad \text{subject to} \quad \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}, \tag{3.1}$$

where $J(\boldsymbol{\alpha})$ controls the coefficient activities in $\boldsymbol{\alpha}$. A number of options for $J(\boldsymbol{\alpha})$ has been studied in the literature. Using ℓ_p-norm of $\boldsymbol{\alpha}$ as $J(\boldsymbol{\alpha})$ to drive the results to become sparse is one of them. This type of methods can be put in a general form as

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\ \|\boldsymbol{\alpha}\|_p^p \quad \text{subject to} \quad \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}. \tag{3.2}$$

3.1.1 ℓ_1-Norm Based Formulation

The problem (3.2) for $p = 2$ has a closed form unique solution, and it is widely used in various fields of engineering, image and signal processing. However, it is by no means the best choice for various problems, especially in image and signal processing. The fact that ℓ_2-norm makes the problem (3.2) to have a unique solution is that selecting any strictly convex function $J(\cdot)$ guarantees such uniqueness. Besides ℓ_2-norm, there are other choices of $J(\cdot)$ that are convex or strictly convex. (Strictly convex implies a unique solution, however, not necessarily a closed form.) The ℓ_p-norms for $p \geq 1$ are a family of such functions. These are defined as

$$\|\mathbf{x}\|_p^p = \sum_i |x_i|^p, \quad 1 \leq p < \infty, \tag{3.3}$$

and

$$\|\mathbf{x}\|_\infty = \max\{x_1, x_2, \cdots, x_M\}. \tag{3.4}$$

In particular, ℓ_1-norm is a special interest due to its tendency to give sparse solutions. In general, as we move from ℓ_2-norm towards ℓ_1-norm, or $p < 1$, we promote sparser solutions.

The exact constraint in both (3.1) and (3.2) is often relaxed to the approximated one $\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2 \leq \epsilon$, where $\epsilon > 0$ is the error tolerance. Specifically, the problem (3.2) for $p = 1$ becomes

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\ \|\boldsymbol{\alpha}\|_1 \quad \text{subject to} \quad \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2 \leq \epsilon. \tag{3.5}$$

Such relaxation is desirable in terms of i) handling the cases where no exact solutions exist; and ii) utilizing the ideas and methods in optimization and statistics.

Alternatively, the problem (3.5) can be formulated as

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2 \quad \text{subject to} \quad \|\boldsymbol{\alpha}\|_1 \leq \mu, \tag{3.6}$$

where $\mu > 0$ is the regularization parameter that controls the trade-off between the approximation error and the sparsity of $\boldsymbol{\alpha}$. The problem (3.6) is the *Lasso*, for 'least absolute shrinkage and selection operator', by [Tibshirani 1996].

The *basis pursuit denoising* formulation of [Chen et al. 2001] is similar to (3.6) with the constraint being replaced by a penalty,

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2 + \lambda\|\boldsymbol{\alpha}\|_1, \tag{3.7}$$

where λ is the parameter that controls the size of residual, i.e. $\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}$. When λ converges to 0^+, the residual converges to 0, and the data is considered essentially noise free. Then the problem (3.7) becomes the main *basis pursuit* formulation of [Chen et al. 2001]

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\boldsymbol{\alpha}\|_1 \quad \text{subject to} \quad \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}, \tag{3.8}$$

which is the problem (3.2) for $p = 1$.

Problems (3.6) and (3.7) are essentially equivalent from a convex optimization point of view [Boyd and Vandenberghe 2004; Mairal et al. 2014]. That is, given $\mathbf{x}$, $\mathbf{D}$, and $\mu > 0$, for any solution $\boldsymbol{\alpha}^*$ of (3.6), there exists $\lambda \geq 0$ such that $\boldsymbol{\alpha}^*$ is also a solution of (3.7). For this reason, (3.7) is well known as 'Lasso' in the literature.

3.1.2 ℓ_0-Norm Based Formulation

The extreme among all the sparsity inducing norms is the case of $p \to 0$. The ℓ_0-norm is defined as $\|\boldsymbol{\alpha}\|_0 = \#\,|\{i : \alpha_i \neq 0\}|$, which is the number of non-zero entries in $\boldsymbol{\alpha}$. Figure 3.1 illustrates the function $|\alpha|^p$, which is the core of the norm computation, for various values of p. It shows that as p tends to zero, the graph of $|\alpha|^p$ becomes similar to that of the indicator function. Thus, the summation of such measures for all the entries of $\boldsymbol{\alpha}$ can be approximated by the count of the nonzeros in vector $\boldsymbol{\alpha}$. When $p = 2$ or 1, the function weighs various values of α differently. For example, in the case of $p = 2$, the function value increases exponentially when $|\alpha|$ increases. For the purpose of promoting sparseness, we expect that function $\|\boldsymbol{\alpha}\|_p^p$ should be able to distinguish the entries of $\boldsymbol{\alpha}$ being zero or not. When $p \to 0$, the function approaches the state where its value becomes 0 for $\alpha = 0$ and 1 for other values.

Setting $J(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|_0$, the problem (3.1) becomes

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\boldsymbol{\alpha}\|_0 \quad \text{subject to} \quad \mathbf{x} = \mathbf{D}\boldsymbol{\alpha}. \tag{3.9}$$

Solving (3.9) is a combinatorial search problem which is proven to be NP-Hard [Davis et al. 1997]. Similar to the ℓ_1-norm based formulation, (3.9) can also be extended to the approximate solution problem

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \|\boldsymbol{\alpha}\|_0 \quad \text{subject to} \quad \|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2 \leq \epsilon, \tag{3.10}$$

and

$$\underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{D}\boldsymbol{\alpha}\|_2 \quad \text{subject to} \quad \|\boldsymbol{\alpha}\|_0 \leq \mu. \tag{3.11}$$

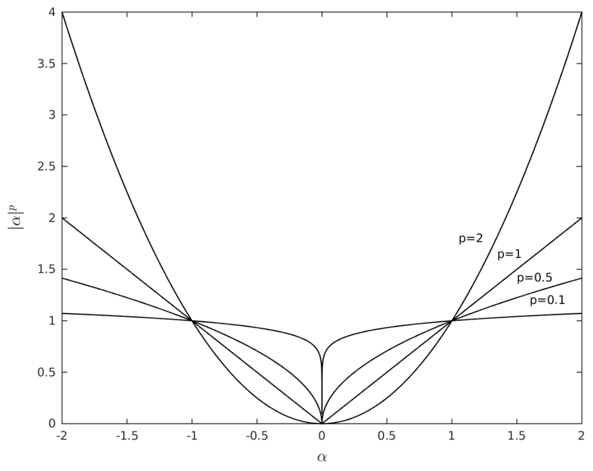


Figure 3.1: The behaviour of $|\alpha|^p$ for $p = \{0.1, 0.5, 1, 2\}$. As p tends to zero, $|\alpha|^p$ approaches the indicator function, which is 0 for $\alpha = 0$ and 1 for $\alpha \neq 0$.

3.1.3 Sparsity and Geometric View of Norms

The sparse representation problem formulations in Sections 3.1.1 and 3.1.2 assume the á priori of sparsity in the solutions. Why sparseness is an appropriate á priori for $\boldsymbol{\alpha}$? From the image and signal processing perspective, two reasons are given in [Olshausen and Field 1997] for the desirability of sparseness. One is based on intuition that natural images may generally be described in terms of a small number of structural primitives, such as edges, lines, or other elementary features. The second form of reasoning is to consider an alternative distribution model, where the distribution of code elements is not unimodal. However, this is contrary to the typical case where an event occurs rarely, and when it does occur it does so along a continuum.

From the geometric point of view, the sparsity inducing effect of the ℓ_p-norm for $p > 0$ can be illustrated using the normal cone of $\mathcal{B} = \boldsymbol{\alpha} \in \mathbb{R}^K$, $\|\boldsymbol{\alpha}\|_p^p \leq \mu$ for some $\mu \in \mathbb{R}_+$. The geometry of ball $\mathcal{B}$ is related to the properties of solution to (3.2) [Bach et al. 2012; Mairal et al. 2014]. Figure 3.2 shows the 2-dimensional ℓ_p-norm balls for $p = \{0.25, 0.5, 1, 2, 4, \infty\}$. The ℓ_p-norm balls for $0 < p \leq 1$ encourage solutions to be on its corners, and thus promote sparser solutions.

Suppose $\mathbf{D} \in \mathbb{R}^{1\times 2}$, and $\mathbf{D}\boldsymbol{\alpha} = \mathbf{x}$. The solutions of $\boldsymbol{\alpha}$ form a line $\mathcal{L}$. The solution that minimizes the ℓ_p-norm of $\boldsymbol{\alpha}$ is obtained at the intersection of ball $\mathcal{B}$ and the line $\mathcal{L}$. Figure 3.3 compares the solutions using ℓ_1- and ℓ_2-norms. In the case of ℓ_1-norm, the intersection is at the horizontal axis, and thus one of the two entries in the solution is zero. On the other hand, the solution using ℓ_2-norm has both of the two entries non-zero. Similarly, the ball $\mathcal{B}$ for ℓ_1-norm in dimension greater than 2, the singular points are located along the axes. Therefore, sparse solutions are obtained if they exist. This simple example can be generalized to problems in higher dimensions.

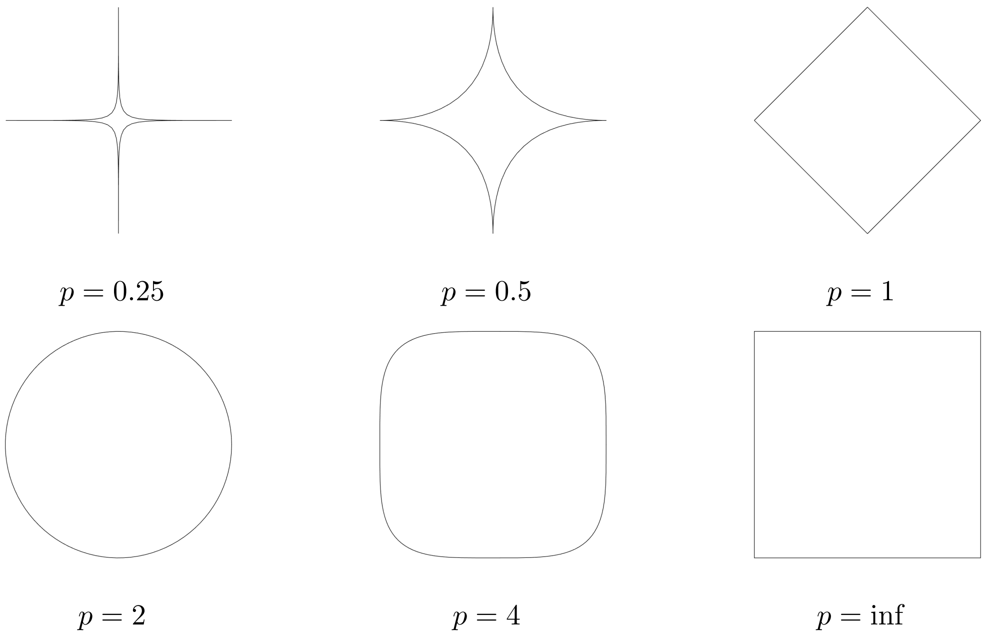


Figure 3.2: Comparison among different balls of sparsity inducing ℓ_p-norms in two dimensions. The singular points appearing along the axis on these balls illustrate the sparsity inducing behaviour of these norms.

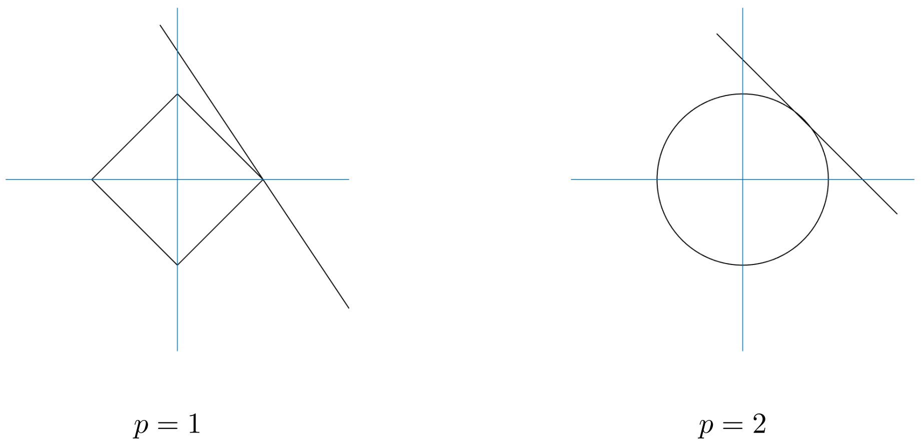


Figure 3.3: The intersection between ℓ_p-ball and the set $\mathbf{D}\boldsymbol{\alpha} = \mathbf{x}$. It is demonstrated in 2D here for $p = 1$ and $p = 2$, When $p \leq 1$, the intersection takes place at a corner of the ball, leading to a sparse solution.

3.1.4 Summary

The problems formulated in this section require the solutions of two main unknowns. That is, the dictionary $\mathbf{D}$ and the sparse coefficient vector $\boldsymbol{\alpha}$. The dictionary $\mathbf{D}$ is usually found by learning a set of atoms (or feature vectors) from a training data set. We refer to this process as dictionary learning. Once the dictionary is known and given a signal vector $\mathbf{x}$, we find the corresponding $\boldsymbol{\alpha}$ through sparse coding. Sparse coding is also referred to as sparse reconstruction, sparse decomposition, or sparse estimation. Both dictionary learning and sparse coding algorithms take different approaches depending on the varying objective optimization functions. In the following two sections, we investigate a number of methods for sparse coding and dictionary learning.

3.2 Sparse Coding

Sparse coding methods aim at obtaining sparse representations of data. Such methods try to solve optimization problems by regularizing the empirical cost with appropriate non-smooth norms. While the literature is vast in this field [Bach et al. 2012], we investigate a few pursuit algorithms, namely, matching pursuit, orthogonal matching pursuit, homotopy and LARS methods, and basis pursuit. These algorithms are commonly used to solve the ℓ_0- and ℓ_1-norm based sparse coding problems, including (3.6) and (3.11), which we applied in our proposed hyperspectral image compression and classification framework.

Problem (3.2) for $p = 2$ has a unique, closed form solution — called minimum-norm solution, and computed as follows. Using the Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^M$ for the constraint set, we define Lagrangian

$$\|\boldsymbol{\alpha}\|_2^2 + \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}), \tag{3.12}$$

where $\|\boldsymbol{\alpha}\|_2^2 = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\alpha}$. Taking the derivative of (3.12) with respect to $\boldsymbol{\alpha}$ results in

$$2\boldsymbol{\alpha} - \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{D}. \tag{3.13}$$

The solution is then

$$\boldsymbol{\alpha}^* = \frac{1}{2}\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{D}. \tag{3.14}$$

Substituting (3.14) into the constraint of Problem (3.2) for $p = 2$, that is

$$\mathbf{x} = \frac{1}{2}\mathbf{D}\boldsymbol{\lambda}^{\mathrm{T}}\mathbf{D} = \frac{1}{2}\mathbf{D}\mathbf{D}^{\mathrm{T}}\boldsymbol{\lambda}, \tag{3.15}$$

then $\boldsymbol{\lambda}$ is solved as

$$\boldsymbol{\lambda} = 2(\mathbf{D}\mathbf{D}^{\mathrm{T}})^{-1}\mathbf{x}. \tag{3.16}$$

From (3.14) and (3.16), we obtain

$$\boldsymbol{\alpha}^* = \mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{D}^{\mathrm{T}})^{-1}\mathbf{x} = \mathbf{D}^{+}\mathbf{x}, \tag{3.17}$$

where $\mathbf{D}^{+} = \mathbf{D}^{\mathrm{T}}(\mathbf{D}\mathbf{D}^{\mathrm{T}})^{-1}$ is the pseudo inverse. This solution has computational advantages such as closed-form and uniqueness as manifested above. However, it does not provide sparse solutions.

In the following subsections, we consider a number of sparse coding approaches that promote sparse solutions for the ℓ_1- and ℓ_0-norm based problems formulated in Section 3.1.

3.2.1 Matching Pursuit

Matching pursuit (MP) is a greedy algorithm that progressively refines the signal approximation with an iterative procedure. It is a common choice of algorithms for dealing with NP-hard problems such as (3.10) and (3.11). Given a dictionary $\mathbf{D}$ and signal $\mathbf{x}$, the first step of a matching pursuit is to approximate $\mathbf{x}$ by projecting it on a vector $\mathbf{d}^{(0)} \in \mathbf{D}$. The vector $\mathbf{d}^{(0)}$ is chosen in

Algorithm 1: Matching Pursuit Algorithm [Mallat and Zhang 1993; Mairal et al. 2014]

```
  Input: Signal $\mathbf{x} \in \mathbb{R}^M$; dictionary $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K\}$ for $\mathbf{d}_i \in \mathbb{R}^M$, and $\|\mathbf{d}_i\|_2 = 1$;
         and stopping criteria $\mu$ (or $\epsilon$).
  Output: Sparse coefficient $\boldsymbol{\alpha}$.
1 begin
2     Initialize $\boldsymbol{\alpha} = \mathbf{0}$
3     while $\|\boldsymbol{\alpha}\|_0 < k$ (or $\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 > \epsilon$) do
4         Select the coordinate with maximum partial derivative
              $\hat{\jmath} \in \underset{j}{\operatorname{argmax}} |\mathbf{d}_j^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})|$        (3.18)
          Update the entry
              $\boldsymbol{\alpha}[\hat{\jmath}] = \boldsymbol{\alpha}[\hat{\jmath}] + \mathbf{d}_{\hat{\jmath}}^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})$        (3.19)
```

such a way that the norm of the residue $\mathbf{R}^{(0)}$ can be minimised. Residue $\mathbf{R}^{(0)}$ here is the approximation error after the first step, i.e., the difference between the original signal and the current approximation using the vector $\mathbf{d}^{(0)}$. The algorithm iterates this procedure by sub-decomposing the current residue $\mathbf{R}^{(0)}$ in the next step. This procedure is also referred to as *coordinate descent* algorithm, where it iteratively selects one entry in the current sparse coefficient $\boldsymbol{\alpha}$ at a time and updates it. This iterative process is shown in Algorithm 1.

The operation (3.18) in Algorithm 1 essentially selects a coordinate by projecting the current residue on a $\mathbf{d} \in \mathbf{D}$ that maximizes $|\mathbf{d}_j^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})|$. This in turn minimizes next residue. The convergence of the approximation error to zero in the case of infinite dimensions is shown in [Mallat and Zhang 1993]. In infinite dimensions, the convergence of this residue can be extremely slow. However, in finite dimensions, the energy of the residue decreases exponentially with a minimum decay rate [Davis et al. 1997; Mallat and Zhang 1993]. That is, the convergence is exponential in finite dimensions. It is clear from (3.18) and (3.19) that the residue indeed monotonically decreases when the density of $\boldsymbol{\alpha}$ increases. In particular, the (3.19) can be interpreted as minimizing the residue with respect to one of the coordinates in $\boldsymbol{\alpha}$ while the others remain fixed.

The *coordinate descent* algorithm for Lasso [Mairal et al. 2014] is comparable to the matching pursuit algorithm. However, the former is for the ℓ_1-regularized problems, while the latter is for the ℓ_0-constraint ones. The coordinate descent algorithm for Lasso, i.e., Problem (3.6), is first proposed by [Fu 1998]. Similar to Algorithm 1, at each iteration, the coordinate descent approach selects one coordinate $\hat{\jmath}$; the selected coordinate is then updated by

$$\boldsymbol{\alpha}[\hat{\jmath}] = S_\lambda\left(\boldsymbol{\alpha}[\hat{\jmath}] + \frac{\mathbf{d}_{\hat{\jmath}}^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha})}{\|\mathbf{d}_{\hat{\jmath}}\|_2^2}\right), \tag{3.20}$$

where S_λ denotes the soft-threshold operator [Donoho and Johnstone 1994]

$$S_\lambda : \theta \mapsto \operatorname{sign}(\theta) \max\{|\theta| - \lambda, 0\}, \tag{3.21}$$

where λ is a chosen threshold. The operator S_λ not only sets small coefficients of a vector $\boldsymbol{\theta}$ to zero, but also reduces the magnitude of the nonzero ones. The selection of $\hat{\jmath}$ at each iteration can be done by cycling through all the entries in $\boldsymbol{\alpha}$, or choosing $\hat{\jmath}$ uniformly at random, or using the same strategy as in Algorithm 1.

Algorithm 2: Orthogonal matching pursuit algorithm [Elad 2010, chap. 3]

```
  Input: The dictionary matrix $\mathbf{D}$, the vector $\mathbf{x}$, and the error threshold $\epsilon_0$
  Output: Approximate solution $\boldsymbol{\alpha}$ to (3.10) and (3.11)
1 begin
2     $\boldsymbol{\alpha} = \mathbf{0}$, $\mathbf{r} = \mathbf{x}$, $S = \emptyset$
3     while (stopping criterion not met) do
4         Compute the error $\epsilon(j) = \min_{z_j} ||\mathbf{d}_j z_j - \mathbf{r}||_2^2$ for $j = 1, 2, \cdots, K$, using $z_j$ as the
           optimal choice $z_j^* = \mathbf{d}_j^T \mathbf{r} / ||\mathbf{d}_j||_2^2$
5         Find a minimiser, $j_0$ of $\epsilon(j) : \forall j \notin S, \epsilon(j_0) \leq \epsilon(j)$, and update $S \leftarrow S \cup j_0$
6         Compute $\boldsymbol{\alpha}$, the minimiser of $||\mathbf{D}\boldsymbol{\alpha} - \mathbf{x}||_2^2$ subject to $S$
7         Update the residue: $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{D}\boldsymbol{\alpha}$
```

3.2.2 Orthogonal Matching Pursuit

It is shown in [Mallat and Zhang 1993] that the convergence of the matching pursuit algorithm is guaranteed in both infinite and finite dimensions. However, the rate of convergence is low in the infinite dimension and exponential in the finite one. The vector selected at each iteration is orthogonal to the updated residue, but in general it is not orthogonal to the previously selected vectors. By orthogonalizing the directions of the projection, the approximation derived from the matching pursuit can be refined. The resulting orthogonal matching pursuit (OMP) converges with a number of iterations in finite dimensional spaces [Davis et al. 1997; Pati et al. 1993].

The OMP selects at each iteration a vector which is orthogonal to all the other previously selected vectors. Davis et al. [1997] shows that the residue of an OMP converges strongly to zero and the number of iterations required for convergence does not exceed the dimension of the input space. Thus, it is guaranteed that the OMP converges in a finite number of iterations. Algorithm 2 shows the OMP algorithm. In Algorithm 2:

- Line 2: The coefficient vector $\boldsymbol{\alpha}$ is initialised to zero vector; the initial residue $\mathbf{r}$ is equal to the input signal $\mathbf{x}$; the set S consists of the indexes of the selected atoms in $\mathbf{D}$, which is initially empty.
- Line 4: The error function is $\epsilon(j) = ||\mathbf{d}_j z_j - \mathbf{r}||_2^2$. That is, the current residue is approximated by a scalar multiple of some column of the dictionary $\mathbf{D}$. We are going to identify this column by computing the error function for each column of $\mathbf{D}$. The scalar value for each test is chosen as the optimal value by setting the $\epsilon(j)$ to zero. This leads to $z_j^* = \mathbf{d}_j^T \mathbf{r}/||\mathbf{d}_j||_2^2$. Substitute z_j^* back into the error expression $\epsilon(j)$, we get

$$\begin{aligned}
\epsilon(j) &= \left\| \frac{\mathbf{d}_j^T \mathbf{r}}{||\mathbf{d}_j||_2^2} \mathbf{d}_j - \mathbf{r} \right\|_2^2 \\
&= ||\mathbf{r}||_2^2 - 2\frac{(\mathbf{d}_j^T \mathbf{r})^2}{||\mathbf{d}_j||_2^2} + \frac{(\mathbf{d}_j^T \mathbf{r})^2}{||\mathbf{d}_j||_2^2} \\
&= ||\mathbf{r}||_2^2 - \frac{(\mathbf{d}_j^T \mathbf{r})^2}{||\mathbf{d}_j||_2^2}.
\end{aligned}$$

Thus, finding the smallest error $\epsilon(j)$ is equivalent to finding the largest inner product of $\mathbf{r}$ and the normalised vectors of matrix $\mathbf{D}$, i.e., solving $\text{argmax}_j\, |\mathbf{d}_j^T \mathbf{r}|$.

- Line 5: Add the index of the newly selected atom to the set S.
- Line 6: Once an atom is selected, we minimise the term $\|\mathbf{D}\boldsymbol{\alpha} - \mathbf{x}\|_2^2$ with respect to $\boldsymbol{\alpha}$, such that its support is S. The problem to be solved is a minimisation of $\|\mathbf{D}_S\boldsymbol{\alpha}_S - \mathbf{x}\|_2^2$, where $\mathbf{D}_S$ contains the selected atoms, and $\boldsymbol{\alpha}_S$ is the non-zero portion of the vector $\boldsymbol{\alpha}$. The solution is given by
$$\boldsymbol{\alpha}_S = (\mathbf{D}_S^{\mathrm{T}}\mathbf{D}_S)^{-1}\mathbf{D}_S^{\mathrm{T}}\mathbf{x}. \tag{3.22}$$

In order to reduce the computational cost involved in Line 6, Rubinstein et al. [2008] introduced an alternative implementation of OMP algorithm, called batch-OMP. The batch-OMP is specifically optimised for sparse coding of large number of signals.

3.2.3 Homotopy Method

Homotopy method relates to orthogonal matching pursuit, but for solving the ℓ_1-norm based problems such as (3.5), (3.6), and (3.7). Similar to OMP, homotopy method builds an active set which is initially empty, and iteratively adds one variable at a time. While the method works similarly for the three problems mentioned above, we attempt to explain the method using Problem (3.6) in particular. The solution to (3.6) is unique, and we denote the set of such solutions for different values of λ as $\mathcal{P} = \{\boldsymbol{\alpha}^*(\lambda) : \lambda > 0\}$ [Bach et al. 2012]. We also name the function $\lambda \mapsto \boldsymbol{\alpha}^*(\lambda)$ as *regularization path*. In fact, the homotopy method finds the set of all solutions of (3.6), i.e., $\mathcal{P}$. An appealing property of the regularization path is that it is piecewise linear, see Figure 3.4 for an example. This property leads to the design of homotopy method for Lasso in [Osborne et al. 2000], and *least angle regression* (LARS) of [Efron et al. 2004].

The homotopy method finds a point on the piecewise linear path, computes the direction of the current piece (see (3.23) and (3.24)), and follows it until the direction changes (the points where the direction change are called kinks). The algorithm either returns the full regularization path, or stops if it reaches a solution for a target λ, a target residue ϵ, or ℓ_1-norm μ. Thus, it may be used for solving (3.5), (3.6), and (3.7) [Mairal et al. 2014]. The homotopy algorithm for Lasso, i.e., (3.6), is described below following [Bach et al. 2012; Mairal and Yu 2012].

Step 1. Set $\lambda = \|\mathbf{D}^{\mathrm{T}}\mathbf{x}\|_\infty$ so that $\boldsymbol{\alpha}^*(\lambda) = 0$.
Step 2. Set $J = \{j \in \{1, \cdots, K\} : |\mathbf{d}_j^{\mathrm{T}}\mathbf{x} = \lambda|\}$ as the set of active atoms.
Step 3. Follow the regularization path by decreasing the value of λ, by computing
$$\boldsymbol{\alpha}_J^*(\lambda) = (\mathbf{D}_J^{\mathrm{T}}\mathbf{D}_J)^{-1}(\mathbf{D}_J^{\mathrm{T}}\mathbf{x} - \lambda\mathbf{t}_J), \tag{3.23}$$
where $\mathbf{t} \in \{-1, 0, 1\}^K$, $\mathbf{t} = \text{sign}(\mathbf{D}^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^*))$, and
$$\boldsymbol{\alpha}_{J^c}^*(\lambda) = 0, \tag{3.24}$$
where J^c is the complement of J, until one of the following events occurs:

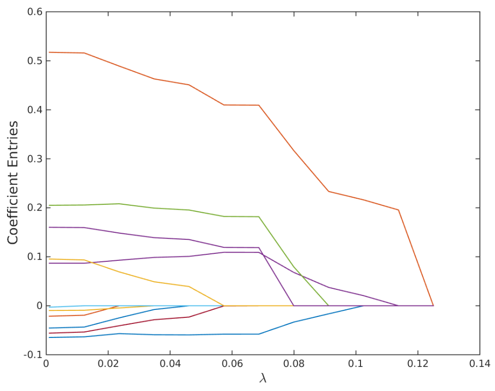
(a) There exists $j \in J^c$ such that $|\mathbf{d}_j^{\mathrm{T}}(\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}^*(\lambda))| = \lambda$, then add j to J;
(b) There exists $j \in J$ such that a non-zero coordinate of $\boldsymbol{\alpha}^*$ hits zero, then remove j from J.

We assume only one of the event occurs at any moment. The value of λ corresponding to the next event can be obtained in closed form such that one can "jump" from a kink to another.

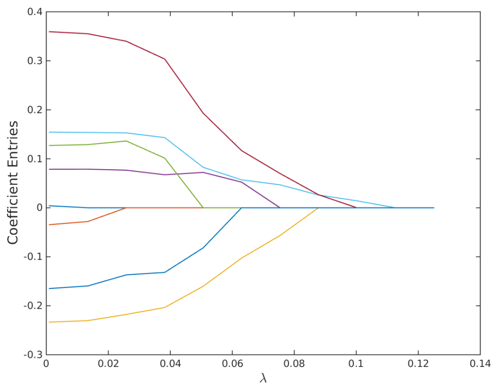
Step 4. Go back to Step Step 3..

(a) Image ‘girl’

(b) Image ‘shutteredwindows’



(c) Path for image ‘girl’



(d) Path for image ‘shutteredwindows’

Figure 3.4: Two examples of regularization path for Lasso using homotopy method. The dictionary used here is the one displayed in Figure 3.8, (a), without sorting. The sparse coding was done for two different images. The initial $\lambda = 0.125$. The support, i.e., the number of non-zero components in a coefficient vector, $\mu = 11$ for the image ‘girl’, and $\mu = 8$ for the image ‘shutteredwindow’. Each path corresponds to one coefficient entry in the solutions. Note that the paths are piecewise linear. Only the ‘Red’ channel of the colour image is used here.

3.2.4 Basis Pursuit

The principle of *basis pursuit (BP)* is to find a sparse coding of the signal whose coefficients have minimal ℓ_1 norm. Thus, it can be used to solve Problem (3.8). The solution involves a convex, non-quadratic optimisation approach. BP adopts algorithms for *linear programming* in solving its optimisation problem [Chen et al. 2001]. The main idea of BP is to identify an optimal basis in $\mathbf{D}$. The atoms in this basis are linearly independent but not necessarily orthogonal. Once such a basis is found, the solution is uniquely determined by the basis. Due to the resemblance to linear programming problem, BP can apply any algorithm for linear programming, such as *simplex* or *interior point* methods to solve its optimisation problem.

Using simplex algorithm, one starts from a linearly independent collection of dictionary atoms, named as current decomposition. Then the current decomposition is updated iteratively by replacing some atoms by new ones, in order to optimize the objective function. The iterative update of the basis stops when an optimal solution is reached.

Interior point algorithm starts from a solution to $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}^{(0)}$ with $\boldsymbol{\alpha}^{(0)}$. It iteratively modifies the coefficients, maintaining $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}^{(l)}$ and applying a transform that sparsifies $\boldsymbol{\alpha}^{(l)}$. The iteration stops when the vector $\boldsymbol{\alpha}$ contains just a few significantly nonzero coefficients. Then, all other coefficients can be forced to zero, leaving those few selected significant coefficients as the sparse coding.

3.2.5 Summary

A few sparse coding algorithms are discussed in this sub-section. The MP and OMP algorithms are commonly used to solve the ℓ_0-norm based sparse coding problems formulated in (3.10) and (3.11). They are both greedy algorithms that find a series of locally optimized terms one by one for a solution, instead of finding a globally optimized one by exhaustive search. As the ℓ_0-norm in (3.9), (3.10), and (3.11) is highly discontinuous, basis pursuit algorithms replace it with continuous or smooth approximations such as ℓ_1-norm. The homotopy and LARS methods are applied to solve ℓ_1-norm based sparse coding problem in (3.6). The comparisons of these methods [Elad 2010, Chapters 3 and 5] show that the LARS and basis pursuit methods perform better than greedy algorithms in reducing the approximation error, while the OMP is more accurate in finding the support (number of nonzero components) in the solutions.

3.3 Dictionary Learning

We now turn to the problem of learning a dictionary. The goal of dictionary learning is to discover a set of base elements, i.e., dictionary atoms, that best describes the latent structure of the collection of signal instances. A set of base elements that attempts to meet such a goal can be constructed in a number of different ways. Recall the Fourier series expansion. We can decompose a periodic signal into the superposition of the basis set $\phi_k(t)$. In general, if $\phi_k(t)$ is chosen appropriately, there exists a dual basis set $\tilde{\phi}_k(t)$ such that $\phi_k(t)$ and $\tilde{\phi}_k(t)$ are orthonormal. Further, the coefficients resulting from such transforms usually concentrate on a number of crucial values, such as the low frequency terms in Fourier transforms. Similar approaches include the widely applied *discrete cosine transform* (DCT) and *discrete wavelet transform* (DWT). These approaches are theoretically well founded and have many successful applications in signal and image processing. However, one limitation of pre-constructed dictionaries is that the basis functions are usually limited to certain types of signal or image [Elad 2010]. This limitation can be overcome by learning a dictionary based on a set of signal instances. The dictionary atoms in this type of approach are built from the underlying empirical data, hence, they can be adapted to any type of signal or image as far as the assumption of á priori sparsity is satisfied.

Given a finite set of training instances $\mathbf{X} = (\mathbf{x_1}, \cdots, \mathbf{x_N})$ in $\mathbb{R}^{M\times N}$, in order to find a dictionary $\mathbf{D} = (\mathbf{d}_1, \cdots, \mathbf{d}_K) \in \mathcal{C}$, where

$$\mathcal{C} = \{\mathbf{D} \in \mathbb{R}^{M\times K}, \text{ s. t.} \forall j = 1, 2, \cdots, K, \mathbf{d}_j^{\mathrm{T}}\mathbf{d} \le 1\}, \tag{3.25}$$

we can set the dictionary learning problem similar to the ℓ_0- and ℓ_1-norm based sparse coding problems in Section 3.1, namely Problems (3.5) ~(3.11), with the quest for $\mathbf{D}$. While the approaches in solving dictionary learning problems differ, they are common in the sense that they are usually set as joint optimization problems of the dictionary and coefficient vector, which are not jointly convex, but convex with respect to each of the two variables ($\mathbf{D}$ and $\boldsymbol{\alpha}$) when the other is fixed [Mairal et al. 2010]. To prevent the entries in $\mathbf{D}$ from being arbitrarily large, which would result in arbitrarily small values in $\boldsymbol{\alpha}$, its columns are constrained to have ℓ_2-norms less than or equal to 1 as defined in $\mathcal{C}$. In the rest of this section, we put our focus on a number of dictionary learning algorithms, which include method of optimal directions, K-SVD, and online dictionary learning. These methods are the most basic ones and they are frequently applied in the quest of a dictionary.

3.3.1 The Method of Optimal Directions

Let $\mathbf{X} \in \mathbb{R}^{M \times N}$ be the set of training samples, $\mathbf{D} \in \mathcal{C}$ the dictionary, and the coefficient vector $\boldsymbol{\alpha}$s constitute the columns in a matrix $\mathbf{A} \in \mathbb{R}^{K \times N}$. The method of optimal directions (MOD) proposed by Engan et al. [Engan et al. 1999] solves the dictionary learning problems, which are expressed in the matrix forms

$$\underset{\mathbf{A},\mathbf{D}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{DA}\|_{\mathrm{F}}^2, \quad \text{subject to} \quad \|\boldsymbol{\alpha}_i\|_0 \leq \mu \quad \text{for} \quad \forall i, \tag{3.26}$$

and

$$\underset{\mathbf{A},\mathbf{D}}{\operatorname{argmin}} \|\boldsymbol{\alpha}_i\|_0 \quad \text{subject to} \quad \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 \leq \epsilon_i, \quad \forall i, \tag{3.27}$$

where the notation $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm, defined as $\|\mathbf{Y}\|_{\mathrm{F}} = \sqrt{\sum_{i,j} \mathbf{Y}_{i,j}^2}$. To optimize the cost function (3.26) and (3.27), MOD alternatively solves $\mathbf{D}$ or $\mathbf{A}$ when the other is fixed. It starts with an initial dictionary, typically set by choosing a number of random atoms or training samples. When $\mathbf{D}$ is fixed, the coefficients are found using one of the ℓ_0-norm based sparse coding algorithms, such as those presented in Section 3.2. This is followed by a dictionary update, which minimizes the cost function $f(\mathbf{D}) = \|\mathbf{X} - \mathbf{DA}\|_{\mathrm{F}}^2$. Taking the derivative of $f(\mathbf{D})$ with respect to $\mathbf{D}$ and equating it to 0 we obtain $(\mathbf{X} - \mathbf{DA})\mathbf{A}^{\mathrm{T}} = 0$. Thus

$$\mathbf{D} = \mathbf{X}\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}. \tag{3.28}$$

The algorithm is described in Algorithm 3.

For the inverse in (3.28) to exist, all dictionary vectors have to be used by at least one training vector. Usually this condition can be met if the training set is reasonably large. Alternatively, a dictionary atom that is not used at all can be replaced by a training vector that is difficult to be approximated [Engan et al. 2007].

3.3.2 The K-SVD Algorithm

The K-SVD algorithm, proposed by Aharon et al. [2006], is a popular approach to solve the same dictionary learning problems as MOD does. The K-SVD algorithm (see Algorithm 4) first carries out the sparse coding of all the training examples, followed by dictionary update step. Any ℓ_0-norm based sparse coding algorithm can be used. The dictionary update is done in the block coordinate descent (BCD) [Bertsekas 1999] fashion, i.e., it updates one column at a time, setting all the other atoms in $\mathbf{D}$ fixed. During the dictionary update step, unlike MOD, the K-SVD algorithm updates the non-zero components of the sparse coefficients in matrix $\mathbf{A}$ as well. That means when updating a particular atom $\mathbf{d}_j$, the algorithm also updates the non-zero components of the sparse coefficients that use $\mathbf{d}_j$. Allowing a change in the values of coefficients during the

Algorithm 3: The MOD dictionary learning algorithm

```
  Input: The training data set $\mathbf{X} \in \mathbb{R}^{M\times N}$
  Output: Trained dictionary $\mathbf{D} \in \mathcal{C}$
1 begin
2     $k = 0$
3     Initialize $\mathbf{D}^{(0)}$ either by using random entries, or $K$ samples randomly chosen from $\mathbf{X}$
4     Normalise the columns of $\mathbf{D}^{(0)}$
5     $k \leftarrow k + 1$
6     Sparse coding: Use a pursuit algorithm to approximate the solutions of (3.11) for
      $\mathbf{x}_1, \cdots, \mathbf{x}_N$ using $\mathbf{D}^{(k-1)}$. The solutions constitute $\mathbf{A}^{(k)}$
7     Dictionary Update: Compute the updated dictionary $\mathbf{D}^{(k)}$ using (3.28) and $\mathbf{A}^{(k)}$
8     Stopping rule: If the change in $\|\mathbf{X} - \mathbf{D}^{(k)}\mathbf{A}^{(k)}\|_{\mathrm{F}}^2$ is small enough (for (3.27)), or the
      target sparsity is reached (for (3.26)), stop; otherwise go to Line 4
9     Output: The desired result is $\mathbf{D}^{(k)}$
```

dictionary update stage accelerates convergence. This kind of update can be done via singular value decomposition (SVD). Thus, the K-SVD algorithm obtains the updated dictionary by K number of SVD computations, where each computation updates one atom.

We now look more closely at the process of updating the dictionary together with the non-zero coefficients. Following the BCD strategy, at a time, we examine one atom, $\mathbf{d}_l$, in the dictionary, and the coefficients that use this atom, which is the l-th row in $\mathbf{A}$ denoted by $\boldsymbol{\alpha}_l^Z$, where the superscript Z is used to indicate the vector is a row in $\mathbf{A}$. Then the loss function can be rewritten as

$$\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_{\mathrm{F}}^2 = \left\|\mathbf{X} - \sum_{j=1}^{K} \mathbf{d}_j \boldsymbol{\alpha}_j^Z\right\|_{\mathrm{F}}^2 = \left\|\left(\mathbf{X} - \sum_{j\neq l}^{K} \mathbf{d}_j \boldsymbol{\alpha}_j^Z\right) - \mathbf{d}_l \boldsymbol{\alpha}_l^Z\right\|_{\mathrm{F}}^2 = \|\mathbf{E}_l - \mathbf{d}_l \boldsymbol{\alpha}_l^Z\|_{\mathrm{F}}^2, \tag{3.29}$$

where the matrix $\mathbf{E}_l$ is the error for all the other training samples when the atom $\mathbf{d}_l$ is removed. Instead of using SVD at this stage to find $\mathbf{d}_l$ and $\boldsymbol{\alpha}_l^Z$ in (3.29), the K-SVD algorithm takes a simple and intuitive remedy step in order to enforce sparsity constraint before applying SVD. It forces the update of $\boldsymbol{\alpha}_l^Z$ to have the same support as the original one. Let $\omega_l = \{i : 1 \leq i \leq N, \boldsymbol{\alpha}_l^Z(i) \neq 0\}$, which is the group of indices of the training samples that activate the atom $\mathbf{d}_l$ in the sparse coding. We then formulate a matrix Ω_l of size $N \times \mu$ where μ is the number of elements in ω_l, with $\Omega_{\omega_l(i),i} = 1$ and zeros elsewhere. The cost term to be minimised now is

$$\|\mathbf{E}_l\Omega_l - \mathbf{d}_l\boldsymbol{\alpha}_l^Z\Omega_l\|_{\mathrm{F}}^2 = \|\mathbf{E}_l^R - \mathbf{d}_l\boldsymbol{\alpha}_l^R\|_{\mathrm{F}}^2. \tag{3.30}$$

Note that the effect of multiplication $\mathbf{E}_l\Omega_l$ is to shrink the size of matrix $\mathbf{E}_l \in \mathbb{R}^{M\times N}$ to $\mathbf{E}_l^R \in \mathbb{R}^{M\times\mu}$, and the same for $\mathbf{d}_l\boldsymbol{\alpha}_l^Z\Omega_l$. The SVD is now applied to decompose $\mathbf{E}_l^R = \mathbf{U}\Delta\mathbf{V}^{\mathrm{T}}$. The solution for the updated $\mathbf{d}_l$ is obtained as the first column of $\mathbf{U}$, and the updated coefficient $\boldsymbol{\alpha}_l^R$ is the first column of $\mathbf{V}$ multiplied by $\Delta(1,1)$, i.e., the largest eigenvalue.

Algorithm 4: The K-SVD algorithm [Aharon et al. 2006, chap. 12]

```
   Input: The training data set $\mathbf{X} \in \mathbb{R}^{M \times N}$
   Output: Trained dictionary $\mathbf{D} \in \mathcal{C}$
 1 begin
 2     Initialize $\mathbf{D}^{(0)}$ either by using random entries, or $K$ samples randomly chosen from $\mathbf{X}$
 3     Normalise the columns of $\mathbf{D}^{(0)}$
 4     $k = 1$
 5     while not converged (stopping rule) do
 6         Sparse coding: Use a pursuit algorithm to approximate the solutions of (3.11) for
           $\mathbf{x}_1, \cdots, \mathbf{x}_N$ using $\mathbf{D}^{(k-1)}$
 7         Dictionary update: For each atom $l = 1, 2, \dots, K$ in $\mathbf{D}^{(k-1)}$, update it by
             • Define the group of indices of samples that use atom $\mathbf{d}_l$, i.e.,
               $\omega_l = \{i : 1 \le i \le N, \boldsymbol{\alpha}_l^Z(i) \neq 0\}$.
             • Compute the overall representation error matrix, $\mathbf{E}_l$, by
                   $$\mathbf{E}_l = \mathbf{X} - \sum_{j \neq l} \mathbf{d}_j \boldsymbol{\alpha}_j^Z.$$
             • Restrict $\mathbf{E}_l$ by choosing only the columns corresponding to $\omega_l$, and obtain $\mathbf{E}_l^R$.
             • Apply SVD: $\mathbf{E}_l^R = \mathbf{U}\Delta\mathbf{V}^{\mathrm{T}}$.
             • Choose the updated dictionary column $\mathbf{d}_l$ to be the first column of $\mathbf{U}$.
             • Update the coefficient vector $\boldsymbol{\alpha}_l^R$ to be the first column of $\mathbf{V}$ multiplied by $\Delta(1,1)$.
           $k = k + 1$.
```

3.3.3 Online Dictionary Learning

Both MOD and K-SVD algorithms access the whole training data at each iteration. Hence, these methods may become impractical in terms of speed and memory requirements when the training data is very large. In such a setting, stochastic approximations based online dictionary learning (ODL) [Mairal et al. 2009] becomes advantageous. Stochastic gradient descent (SGD) algorithms pick a random training example at each iteration and updates the objective function on the basis of this example only. Given a training set $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \in \mathbb{R}^{M \times N}$, online dictionary learning considers the ℓ_1-regularized problem

$$\underset{\mathbf{D}\in\mathcal{C}, \boldsymbol{\alpha}\in\mathbb{R}^K}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1 \right), \tag{3.31}$$

where λ is the regularization parameter. The empirical cost function for the problem (3.31) is

$$f_N(\mathbf{D}) = \frac{1}{N} \sum_{1}^{N} l(\mathbf{x}_i, \mathbf{D}), \tag{3.32}$$

where

$$l(\mathbf{x}, \mathbf{D}) = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1. \tag{3.33}$$

As pointed out by Bottou and Bousquet [2008], since the empirical cost $f_N(\mathbf{D})$ is an approximation of expected cost $f(\mathbf{D}) = \mathbb{E}(l(\mathbf{x}, \mathbf{D})) = \lim_{N\to\infty} f_N(\mathbf{D})$, it should not be necessary to carry out the minimization of $f_N(\mathbf{D})$ with great accuracy. Moreover, [Bottou and Bousquet 2008] has shown that the convergence of SGD is primarily limited by the stochastic noise induced by random choice of one sample at each iteration. The total number of training samples is not an essential factor for the convergence.

Algorithm 5: Online dictionary learning [Mairal et al. 2009]

```
  Input: i.i.d signals $\mathbf{x}_1, \mathbf{x}_2, \cdots \in \mathbb{R}^M$, regularization parameter $\lambda$, initial dictionary
         $\mathbf{D}_0 \in \mathcal{C}$, number of iterations $T$.
  Output: Trained dictionary $\mathbf{D} \in \mathcal{C}$
1 begin
2     Initialization: $\mathbf{A}_0 \in \mathbb{R}^{K\times K} \leftarrow 0$, $\mathbf{B}_0 \in \mathbb{R}^{M\times K} \leftarrow 0$
3     for $t = 1$ to $T$ do
4         Draw $\mathbf{x}_t$ and compute
              $\underset{\boldsymbol{\alpha}\in\mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1$    (3.34)
5         $\mathbf{C}_t \leftarrow \mathbf{C}_{t-1} + \boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^{\mathrm{T}}$
6         $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \mathbf{x}_t\boldsymbol{\alpha}_t^{\mathrm{T}}$
7         Update $\mathbf{D}$ using Algorithm 6 with $\mathbf{D}_{t-1}$ as warm restart, so that
              $\mathbf{D}_t = \underset{\mathbf{D}\in\mathcal{C}}{\operatorname{argmin}} \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1\right)$
                    $= \underset{\mathbf{D}\in\mathcal{C}}{\operatorname{argmin}} \frac{1}{t}\left(\frac{1}{2}\operatorname{tr}(\mathbf{D}^{\mathrm{T}}\mathbf{D}\mathbf{C}_t) - \operatorname{tr}(\mathbf{D}^{\mathrm{T}}\mathbf{B}_t)\right)$    (3.35)
```

The ODL algorithm is summarized in Algorithm 5. Assuming the training data set is composed of i.i.d samples of a data distribution, the algorithm draws one sample $\mathbf{x}_t$ at a time, and alternates between sparse coding and dictionary update. The sparse coding is done using the dictionary obtained from the previous iteration. The information on newly acquired sparse coefficient $\boldsymbol{\alpha}_t$ is incremented into the matrix $\mathbf{C}$ and $\mathbf{B}$ that store all the past information. Accumulating the past data through matrices $\mathbf{C}$ and $\mathbf{B}$ enables the procedure not to require storing all the training vectors and their coefficients. The new dictionary is computed by minimizing the surrogate (3.35) using $\boldsymbol{\alpha}_i$s for $i < t$ computed during the previous iterations. The function (3.35) aggregates the past information with a few statistics obtained during the previous steps, i.e., the vector $\boldsymbol{\alpha}_i$s. As it is shown in [Mairal et al. 2009] that the objective functions (3.32) and (3.35) converge almost surely to the same limit, and thus (3.35) acts as a surrogate for (3.32). The solution to problem in (3.35) amounts to the dictionary update in Algorithm 6. Each atom $\mathbf{d}_j$ is updated in turn when

Algorithm 6: Dictionary update [Mairal et al. 2009]

```
   Input: $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_K] \in \mathbb{R}^{M \times K}$ (input dictionary), $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_K] \in \mathbb{R}^{K \times K}$,
          $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_K] \in \mathbb{R}^{M \times K}$
   Output: Updated dictionary $\mathbf{D}$
1  begin
2      Repeat
3          for $j = 1$ to $K$ do
4              Update the $j$th column to optimize for (3.35)
                   $$\mathbf{u}_j \leftarrow \frac{1}{\mathbf{C}_{j,j}}(\mathbf{b}_j - \mathbf{D}\mathbf{c}_j) + \mathbf{d}_j, \quad \mathbf{d}_j \leftarrow \frac{1}{\max\{\|\mathbf{u}_j\|_2, 1\}}\mathbf{u}_j \qquad (3.36)$$
5      Until Convergence
```

keeping the others fixed in Algorithm 6. This is formulated in the matrix form as

$$\mathbf{d}_j \leftarrow \underset{\mathbf{d}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{X} - \mathbf{D}\mathbf{A} + \mathbf{d}_j\boldsymbol{\alpha}^j - \mathbf{d}\boldsymbol{\alpha}^j||_F^2, \tag{3.37}$$

where $\boldsymbol{\alpha}^j$ is the j-th row of $\mathbf{A} \in \mathbb{R}^{K \times N}$. The solution of Problem (3.37) can be given as $\mathbf{d}_j$ in (3.36) including an orthogonal projection of the vector $\mathbf{u}_j$ onto the unit ℓ_2 ball. Moreover, for large values of t, $\mathbf{D}_t$ is close to $\mathbf{D}_{t-1}$. Hence, it is efficient to use $\mathbf{D}_{t-1}$ as warm restart to compute $\mathbf{D}_t$.

In practice, the convergence of Algorithm 6 is often accelerated by drawing a mini-batch of signals, i.e., $\eta > 1$ signals, at a time. Then the Lines 5 and 6 in Algorithm 6 become

$$\mathbf{C}_t \leftarrow \mathbf{C}_{t-1} + \frac{1}{\eta}\sum_{i=1}^{\eta} \boldsymbol{\alpha}_{t,i}\boldsymbol{\alpha}_{t,i}^{\mathrm{T}}, \tag{3.38}$$

$$\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \frac{1}{\eta}\sum_{i=1}^{\eta} \mathbf{x}_{t,i}\boldsymbol{\alpha}_{t,i}^{\mathrm{T}}. \tag{3.39}$$

3.3.4 Some Examples of Dictionaries

We illustrate in this section some visual examples of dictionaries learned using the above methods, namely, MOD, K-SVD, and ODL, discussed in the preceding sections. For comparison, we also show some visualizations of principle components (PCs), DCT dictionary, and Gabor filters.

Figure 3.5 presents the visualizations and the comparison of reconstruction errors of three dictionaries learned using MOD, K-SVD, and ODL, respectively. The reconstruction errors are in terms of root mean squared error (RMSE) defined as

$$\text{RMSE} = \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_{\mathrm{F}}. \tag{3.40}$$

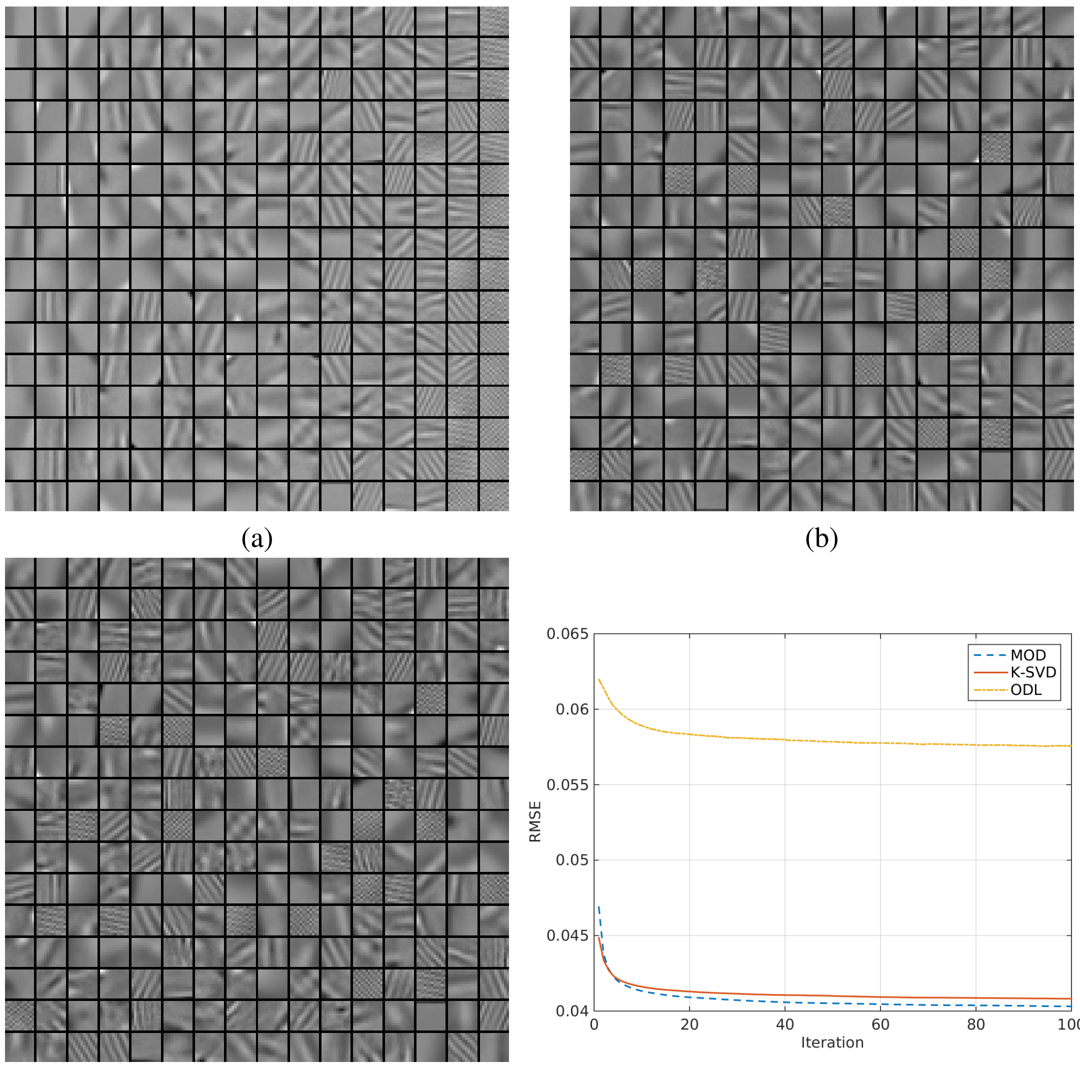


Figure 3.5: (a) Dictionary learned using MOD; (b) Dictionary learned using K-SVD; (c) Dictionary learned using ODL; (d) Comparison of the reconstruction errors among MOD, K-SVD, and ODL. The dictionaries were trained on 8 grey scale images, namely, ‘baboon’, ‘barbara’, ‘camera’, ‘elaine’, ‘lake’, ‘lena’, ‘man’, and ‘woman’, from the USC-SIPI Image Database at `http://sipi.usc.edu/database/database.php`. 12×12 image patches were extracted from these test images to formulate the training samples. The training samples were with zero means and normalized. The dictionary training parameters were the same for all three methods considered. That is, the sizes of the dictionaries were 144×256; the initial dictionaries were the same; the number of iterations was 100; and the parameter for the sparse coding $\mu = 5$.

We observe that some of the dictionary atoms in all three dictionaries resemble some of the discrete cosine bases, and some resemble Gabor wavelets (see Figure 3.8). Moreover, with less than 100 iterations, all three methods were able to bring the RMSE below 0.06. This indicates the effectiveness of the dictionaries learned using these methods. However, the RMSE of ODL is higher than the two others. This results from the differences in dictionary updates for the three methods. While MOD and K-SVD consider the entire training set at each iteration, ODL only considers a single training sample or a mini-batch of them. Further, MOD and K-SVD try to optimize the objective function (the least square error in our case) by minimizing the empirical cost. Thus, the resulting dictionaries are learned to fit the training data. In the case of ODL, the objective function is optimized by minimizing the expected cost, and the learned dictionary aims to represent not only the training samples, but also the unknown future data.

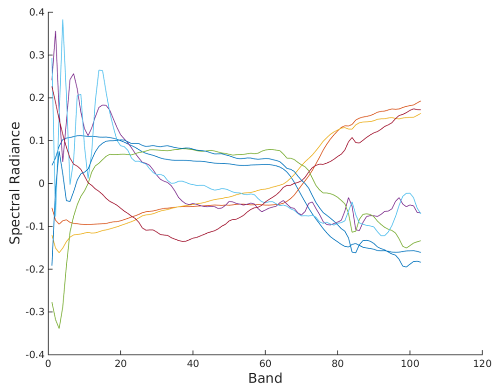
Similar experiments were also carried out for hyperspectral data. We used a sub scene of ROSIS03 Pavia University hyperspectral imagery. The dictionaries were learned in the spectral domain. Thus, only a small number of dictionary atoms were selected for display. Figure 3.6 shows 8 uniformly sampled atoms from the sorted dictionaries, as well as the reconstruction errors using these dictionaries. The result for the RMSE is similar to the one in Figure 3.5. However, the spectral features of the dictionary atoms in the resulting dictionaries using different methods need to be further investigated to establish their similarities. We also compared in Figure 3.7 among the first 8 atoms from the sorted dictionaries learned using MOD, K-SVD, and ODL, respectively, and the first 8 principle components of the training hyperspectral image data.

We present in Figure 3.8 some visual results obtained with learned dictionaries, principle components, a DCT dictionary, and Gabor filters. For this example, we used two sets of natural image data, one contains 88856 distinct 12×12 image patches, and the other 38301 distinct patches of the same size. The dictionaries were trained using ODL for the formulation of (3.31), with the regularization parameter λ set to 0.1. The dictionary atoms are sorted from most to least used, and they are in row-major order. The principle components of these two data sets are also displayed in Figure 3.8. The two data sets contain different images, however there is no significant structural difference among them. Further, both image data sets are centralized and normalized during the pre-processing stage. Similar to the work of [Olshausen and Field 1997], the dictionary elements in Figure 3.8 are localized, oriented, and with both low and high frequencies. One should also observe that the similarity between the principle components of the two sets of data is more prominent than the one between the dictionaries. There is a resemblance between principle components and DCT base elements. The Gabor filters in Figure 3.8 correspond to $g(x, y) = e^{-x^2/(2\sigma_x^2)-y^2/(2\sigma_y^2)} \cos(\omega x + \phi)$. The three Gabor filters in the top row were obtained by using different phases, i.e., $\phi = 0, \pi/2$, and π, respectively. The three others in the bottom row were obtained by rotating the first filter in the top row. That is, the x and y in function $g(x, y)$ were replaced by $x' = x \cos\theta + y \sin\theta$ and $y' = -x \sin\theta + y \cos\theta$. The similarity among (a), (b), and (f) in Figure 3.8 shows that the dictionary learning is able to recover similar structures as the Gabor wavelets with localized and oriented atoms.

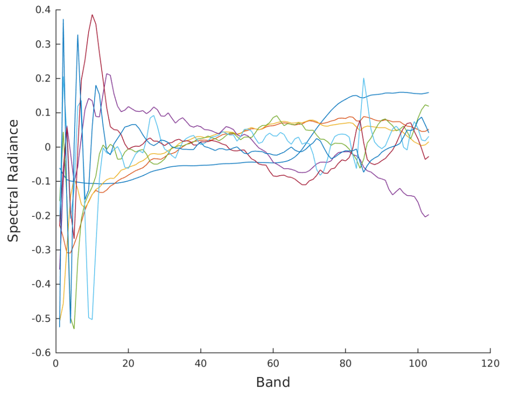
3.3.5 Summary

Three dictionary learning approaches — MOD, K-SVD, and ODL — are discussed in this section. The advantages of the MOD method are its simple way of updating the dictionary, and parameter free. The K-SVD algorithm is probably one of the most popular approaches in dictionary learning. It is related to the MOD, but the dictionary update step includes updating the nonzero coefficients as well. Furthermore, the K-SVD updates the dictionary atoms one by one, in a block coordinate manner. Unlike the MOD and K-SVD, the ODL is based on stochastic approximations. Although inferior to the MOD or K-SVD from the perspective of data fitting, the ODL is much faster than the MOD or K-SVD. This is because the ODL processes only one sample (or a small fraction) of the training set for each dictionary update step, where the MOD and K-SVD process the entire training set or a batch of it. Thus, the ODL becomes advantageous in the case of large training

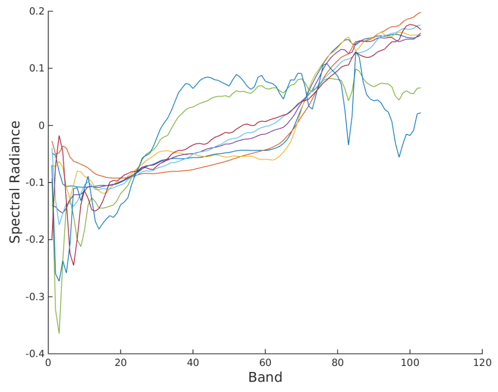
(a)



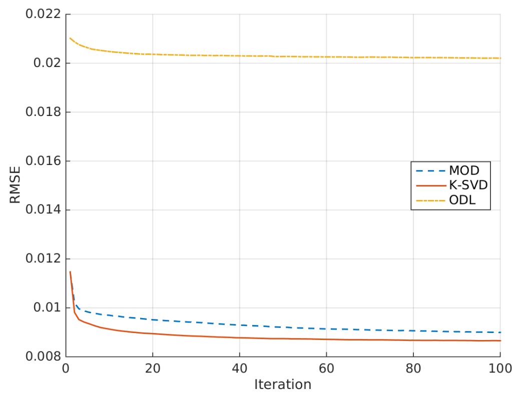
(b)



(c)



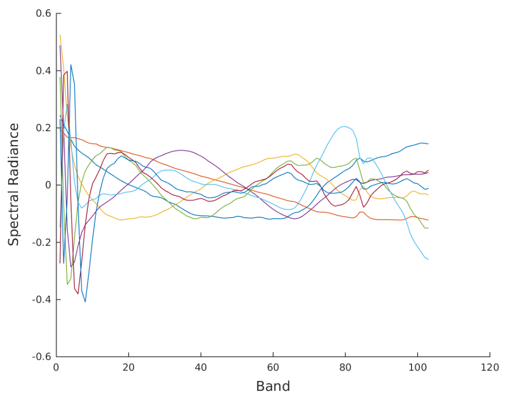
(d)



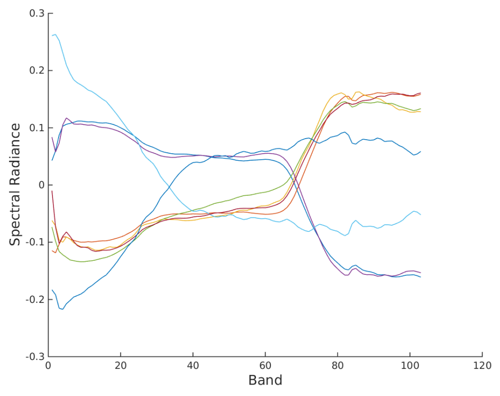
(e)

Figure 3.6: (a) A false colour image of sub-scene of Pavia University. Randomly selected atoms from (b) Dictionary learned using MOD; (c) Dictionary learned using K-SVD; (d) Dictionary learned using ODL. (e) Comparison of the reconstruction errors among MOD, K-SVD, and ODL. The dictionaries were trained using a sub cube, $128 \times 256 \times 103$, from ROSIS03 University of Pavia data set. The dictionary training parameters were the same for all three methods considered. That is, the sizes of the dictionaries were 103×256; the initial dictionaries were the same; the number of iterations was 100; and the parameter for the sparse coding $\mu = 5$.

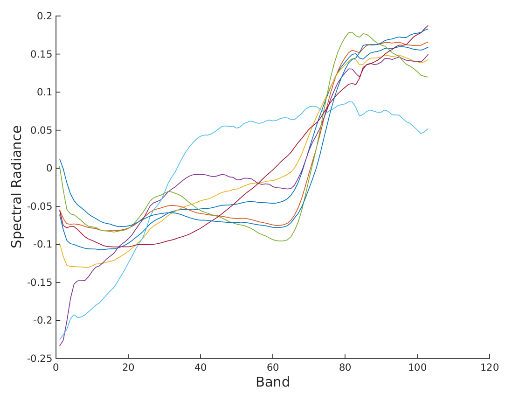
sets, where the MOD or K-SVD becomes slow in such scenario. This is the reason we adopted the ODL for learning a dictionary for hyperspectral images, as the training data set is usually large.



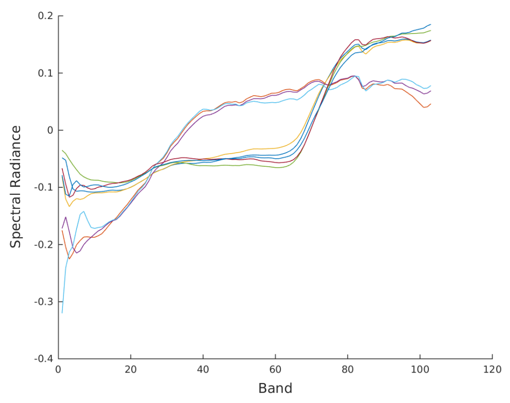
(a)



(b)



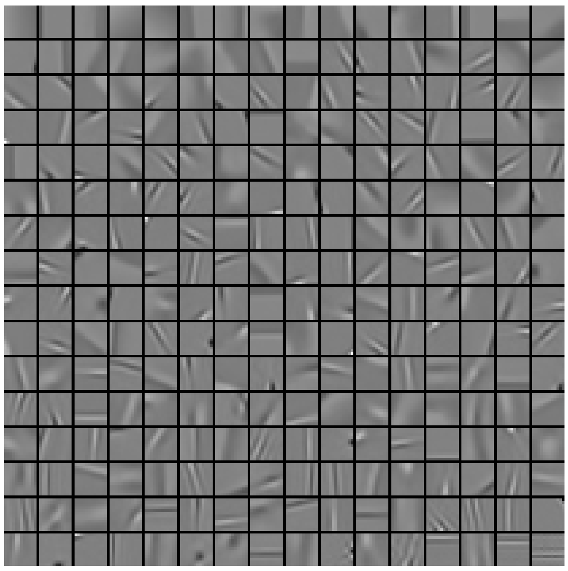
(c)

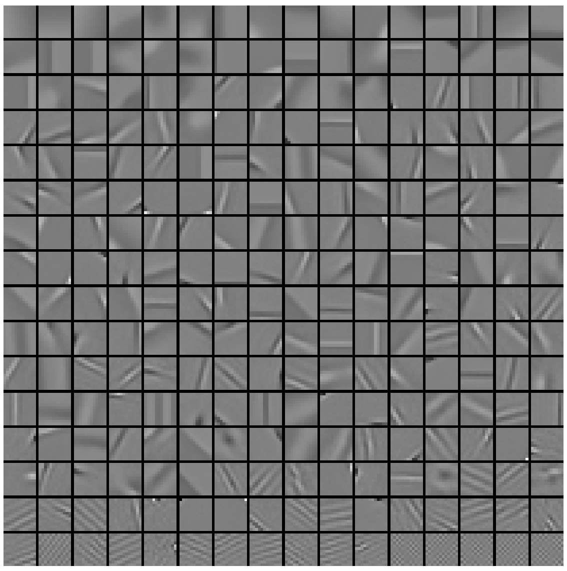


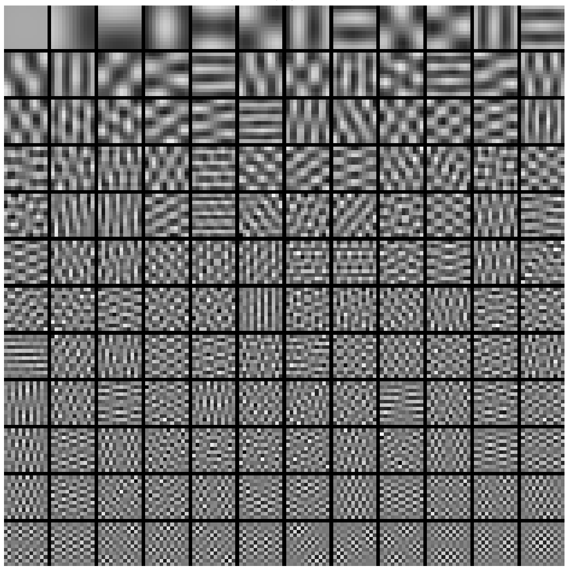
(d)

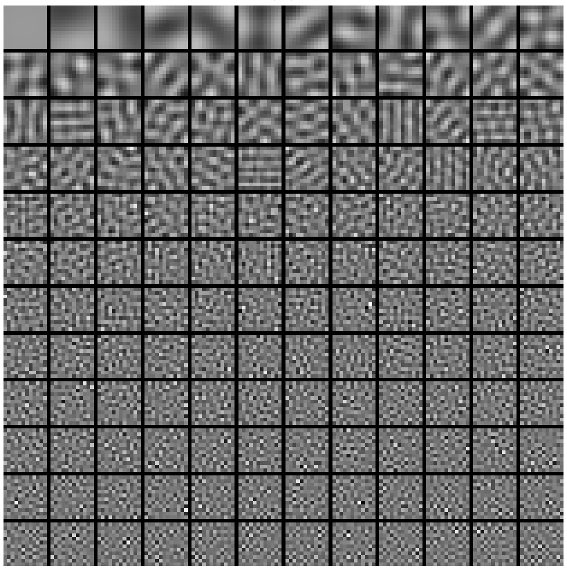
Figure 3.7: The first 8 columns in the (a) PCs; (b) Dictionary learned using MOD; (c) Dictionary learned using K-SVD; (d) Dictionary learned using ODL. The principle components are obtained on the same sub scene of University of Pavia as in Figure 3.6, and the dictionaries are the same ones as in Figure 3.6.

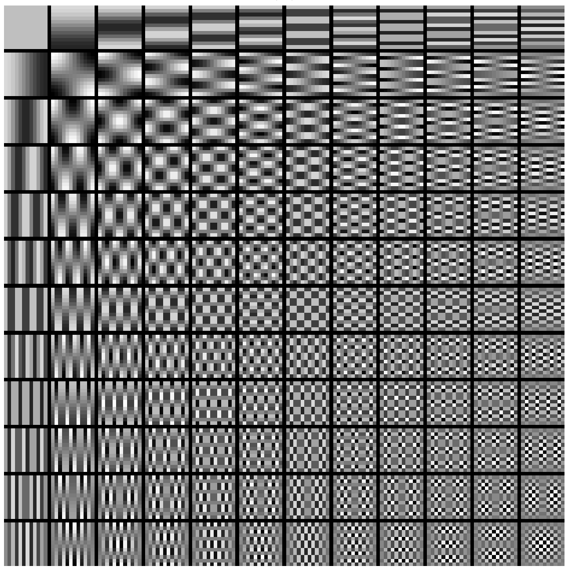
In this section, we have only presented three dictionary learning algorithms that address the formulations (3.26), (3.27), and (3.31). Other dictionary learning methods exist for these problems, their variants, or extensions. These include the stochastic gradient descent method for (3.31) in [Olshausen and Field 1997], an extension of the MOD [Engan et al. 2007], recursive least squares dictionary learning algorithm [Skretting and Engan 2010] for (3.27), hierarchical dictionary learning for (3.31) where the penalty term is replaced by a more complex one [Jenatton et al. 2010], Bayesian dictionary learning modelling [Zhou et al. 2012; Hughes et al. 2013], multiscale dictionary learning [Sallee and Olshausen 2003], and semi-multiscale extension of K-SVD [Mairal et al. 2008b].


(a)


(b)


(c)


(d)


(e)

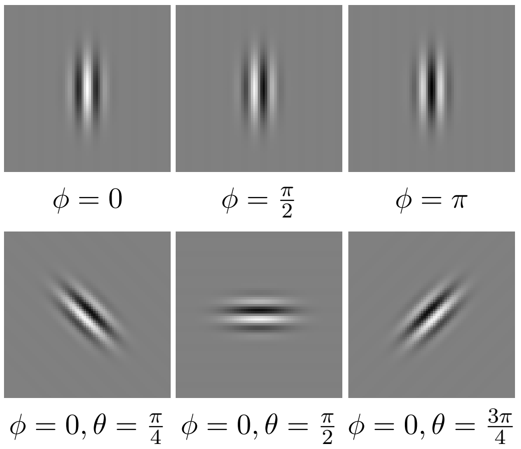

$\phi = 0$
$\phi = \frac{\pi}{2}$
$\phi = \pi$
$\phi = 0, \theta = \frac{\pi}{4}$
$\phi = 0, \theta = \frac{\pi}{2}$
$\phi = 0, \theta = \frac{3\pi}{4}$
(f)

Figure 3.8: Dictionaries and principle components obtained from two sets of image data. (a), (b): Dictionaries of size 144×256; (c),(d): principle components; (e) DCT dictionary; (f) Gabor filters.

Chapter 4
Hyperspectral Image Compression

Data compression plays an important role in efficient information storage and communication. From our daily mobile communication, high definition streaming multimedia, to the cloud storage, data compression is ubiquitous. Especially, in the current trends of big data, Internet of things, the velocity, volume and variety of data generated by all sizes of entities demand constant advances in storage and communication technologies, including data compression. Data compression involves the development of compact representation of information. Most representations of information contain large amount of redundancy. Redundancy has different forms. It can be in the form of correlation, such as spatially close pixels in an image are also close in their intensity values; consecutive video frames have little change in their contents. Redundancy can be dependant on the perceptions of the users, such as a human observer is more responsive to the lower frequencies in an image, and hence, the information in the higher frequencies become redundant. An important aspect of data compression is to find a proper model that captures the essential part of information or data, and removes the redundancy as much as possible.

Hyperspectral imagery is characterised by high dimensionality and enormous data volume due to the increasing spatial, spectral and temporal resolutions of imaging spectrometers. Efficient compression of hyperspectral data is essential in many aspects such as storage, transmission and analysis of the data. From the acquisition points on the sensors, to the ground stations, distribution points, and then to the end users, there are many places where the compression of hyperspectral image is necessary. Two types of data compression can be performed on hyperspectral data, namely lossless and lossy compression. Lossless compression eliminates unnecessary redundancy without loss of information. The decompressed, or the reconstructed data is identical to the original data. The requirement of zero loss of information puts a limit on the compression we can get. There are some applications demand lossless compression. For example, medical applications that demand lossless compression, since any errors or loss of information in medical images implicates the diagnosis or treatment of a patient. Where the storage and transmission channels allow, the ground stations or the distribution points prefer acquiring hyperspectral data without any loss of information. Thus, lossless compression is applied in this scenario.

On the other hand, lossy compression removes unwanted or insignificant information which results in entropy increase. The primary goal of lossy compression is to minimize the number of bits required to represent a sequence within the acceptable range of distortion, or loss of information. Lossy compression methods achieve remarkably higher compression ratios than lossless compression methods. The choice of compression often depends on the type of application in which the data is used.

A particular type of lossy compression, termed *near-lossless* compression, is applied where a tight control of image quality is required. This is typically achieved by imposing a bound on the ℓ_∞-norm. In some applications, such as medical imaging or remote sensing [Beerten et al. 2015], a large error in a sample could lead to failure in diagnosis or classification.

In the following sections, we discuss some common issues of data compression in general, with our focus on hyperspectral image compression. However, most of the coding components and techniques are applicable in the general domain of data compression. Hence, we describe such components or techniques from a general point of view, and more details in the case of hyperspectral image are given where necessary. Section 4.1 starts with a high level architecture of hyperspectral image compression, which gives us a framework to discuss various data compression components and related techniques in the following sections. We introduce entropy coding, an essential building block in data compression, in Section 4.2. Section 4.3 presents some details on the problem of quantization, as this is the basic technique to achieve high compression ratio in lossy compression. A popular compression technique, especially in the case of lossless compression, the predictive coding is discussed in Section 4.4. Section 4.5 introduces the transform coding technique, where we study various transforms and their applications in hyperspectral data compression. In Sections 4.6 and 4.7, we give brief overviews of the latest image compression standards from JPEG and CCSDS, respectively.

4.1 The Architecture of Hyperspectral Data Compression

A typical hyperspectral data compression system consists of two structural blocks [Pickering and Ryan 2006]: an *encoder* and a *decoder*. Besides these two blocks, some systems also include a pre-processing of the input data and a post-processing of the reconstructed data. The encoder usually consists of the following phases: *mapping* and *entropy coding*, while the decoder consists of *entropy decoding* and *inverse mapping*. The functionality of each of these stages is briefly explained as follows.

Mapping: Transforms the input data in a way so that the redundancy in it can be reduced. The raw data is decorrelated in this phase. The mapper often uses one of the standard image compression techniques, such as vector quantization and transform coding.

Entropy coding: Creates a fixed- or variable-length code to represent the output of the first phase.

Entropy decoding: Performs the inverse operation of the entropy coder.

Inverse mapping: Performs the inverse operation of the mapper.

We use the term ‘entropy coding’ here in the sense of ‘coding’ primarily. Coding is the process of mapping each possible output from an information source to a sequence of binary bits, called *code* or *codeword*. An entropy coder maps its input symbols to another set of symbols where the average number of bits per symbol can be minimised. The goal is to carry as much information as possible using a limited number of bits. For this purpose, we need to use different number of bits to represent different symbols. The codes generated from this approach are referred to as *variable length codes*. If we use fewer bits to represent the symbols that occur more often, we could reduce the number of bits per symbol on average. To utilize this idea, an entropy coder needs to know the probabilities of occurrence of the input symbols. If such statistics are known, then a fixed mapping such as *Huffman coding* can be used. Otherwise, if the statistics change over time, or unknown, methods such as *adaptive Huffman coding* should be used.

In lossy compression, a *quantization* stage is often placed between the mapper and entropy coding, where the compression occurs with some loss of information. Before we look more closely at the techniques used for the mapper stage, we examine *Huffman coding* and *quantization* in more detail.

4.2 Entropy and Huffman Coding

4.2.1 Entropy

Suppose we have a random variable (or source), X, any outcome x of X is in a set $\mathcal{A}_X$ with probability distribution $P_X(x)$. The entropy of X is given by

$$H(X) = -\sum_{x \in \mathcal{A}_X} P_X(x) \log_2 P_X(x). \tag{4.1}$$

The entropy is used to measure the amount of information. The unit of entropy in (4.1) is bits. We can see from the definition of entropy that it is a strictly decreasing function of the probability $P_X(x)$. This implies that a highly unlikely event carries considerable information when it does occur. Shannon showed that firstly, the entropy is the minimum average number of bits needed to represent each outcome of the random variable X; secondly, the fixed length coding is incapable of guaranteeing that all source outcome will be represented exactly [Taubman 2000]. Hence, the variable length coding is the solution.

4.2.2 Huffman Coding

Huffman coding procedure generates codes, called *Huffman codes*, that are variable length, optimum, and prefix codes [Huffman 1952]. Being optimum guarantees the average codeword length is minimized. A prefix code is one in which no codeword is the prefix of any other code. Huffman coding is based on the following observations [Sayood 2005].

1. In an optimum code, symbols that occur more frequently have shorter codewords than those that occur less frequently.
2. In an optimum code, the two symbols that occur least frequently have codewords of the same length.

A simple addition to these observations is that the two codewords for the two symbols with least probability of occurrence differ only in the last bit.

The Huffman coding procedure can be implemented by building a binary tree first, and then traverse the tree to generate the codewords for each symbol in the leaf nodes. Assume an information source puts out N symbols $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, with probability of occurrence $\mathcal{P} = \{P(s_1), P(s_2), \ldots, P(s_N)\}$, and an initial empty binary tree T.

Step 1. Sort the symbols in $\mathcal{S}$ according to their corresponding probabilities in $\mathcal{P}$.

Step 2. Insert the two symbols, say s_i and s_j where $1 \leq i, j \leq N$, with the least frequencies (or probabilities) as two leaves in the binary tree T, and create a parent node of an arbitrary symbol s' with $P(s') = P(s_i) + P(s_j)$.

Step 3. Remove the two symbols, i.e., s_i and s_j, identified in Step Step 2. from $\mathcal{S}$, and add the arbitrary symbol s' to it. Meanwhile, $P(s_i), P(s_j)$ are also deleted from $\mathcal{P}$, and $P(s')$ is added to $\mathcal{P}$.

Step 4. Repeat Steps Step 1., Step 2. and Step 3. until there is only one symbol left in $\mathcal{S}$, which becomes the root node of T.

To obtain the codeword for each symbol, we traverse the tree from the root to each leaf node, assigning bit 0 to left child, and 1 to the right one. Once a leaf node is reached, the concatenated bit sequence becomes the codeword for the symbol corresponding to that leaf node.

As Huffman codes are prefix codes, they are uniquely decodable. A simple decoding algorithm [Taubman 2000] for prefix codes, as shown in Algorithm 7, can be used for the decoding of Huffman codes.

Algorithm 7: Huffman decoding

```
  Input: A bit stream, the symbol list $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, and its corresponding
         codeword list, or codebook, $\mathcal{C} = \{\hat{c}(s_1), \hat{c}(s_2), \cdots, \hat{c}(s_N)\}$
  Output: Decoded sequence
1 begin
2     while not the end of bit stream do
3         for $l = 0, 1, \cdots$ do
4             for $i = 1$ to $N$ do
5                 Compare the first $l$ bits of the received bit stream with codeword $\hat{c}(s_i)$
6                 if a match is found then
7                     The prefix code condition guarantees no other value of $l$ will yield a
                        match, so the symbol must be $s_i$
8                     Remove the $l$ bits from the bit stream, and apply the algorithm
                        recursively from Line 2 to decode the next symbol
```

Huffman coding requires that the statistics of the information source is known. If this is not the case, or the statistics vary over time, Huffman coding needs to be adapted accordingly. One potential solution is to turn the Huffman coding into two-pass procedure: the statistics are collected periodically in the first pass, and the source is encoded in the second pass. This approach requires the transmission of updated codewords periodically to the decoder. Another solution is for both the encoder and decoder to periodically estimate the source statistics and construct identical Huffman codes based on the previously encoded source outcomes. This approach is known as adaptive Huffman coding [Sayood 2005].

4.3 Quantization

Quantization is an element in lossy compression systems that leads to higher compression at the expense of losing some precision of the data. The number of distinct values a source input produces is usually much larger than the number of codewords available to represent them. The process of mapping a large, or possibly infinite, number of values onto a small set of values is called quantization. If the set of inputs and outputs of a quantizer consists of scalars, we call the mapping process *scalar quantization*. A more complicated quantization could be handling vectors as inputs and outputs, and it is referred to as *vector quantization*. Some comprehensive treatments of quantization can be found in [Sayood 2005; Taubman 2000].

4.3.1 Scalar Quantization

A quantizer observes a single input value and selects the nearest approximation value from a predetermined finite set. A K-point scalar quantizer Q is a mapping $Q : \mathbb{R} \mapsto \mathcal{Y}$, and

$$\mathcal{Y} = \{y_1, y_2, \cdots, y_K\} \subset \mathbb{R} \tag{4.2}$$

is the output set with size $|\mathcal{Y}| = K$. Any K-point quantizer essentially partitions the real number line into K intervals or cells. These intervals are identified with decision boundaries $\{b_i\}_{i=0}^{K}$. That is, $[b_{i-1}, b_i)$ is an interval and defined as $\{r : b_{i-1} \leq r < b_i\}$ for $i = 1, 2, \cdots, K$. A cell that is bounded is called *granular* cell, and an unbounded one is *overload* cell. Usually, $b_0 = -\infty$, $b_K = \infty$, and the two outer cells become the overload ones. The output values, y_i, are also referred to as *output levels*, or *reproduction values*. Each output value is assigned to a sequence of binary bits, called *codeword*. The set of codewords, or *codebook*, is denoted by

$$\mathcal{C} = \{\hat{c}_{i \in \mathcal{I}} : \mathcal{I} = \{1, 2, \cdots, K\}\}. \tag{4.3}$$

Since the reproduction values and the codewords correspond to each other, the term 'codebook' can also be used to refer to $\mathcal{Y}$, and the word 'codeword' is sometimes used interchangeably with output level. In a practical data compression system, the quantizer consists of encoder mapping, $\mathcal{E}$, and decoder mapping, $\mathcal{D}$, or simply, quantizer and de-quantizer, respectively. The quantizer receives an input value and maps it to a codeword as an output, i.e, if $Q(x) = y_i$, then $\mathcal{E}(x) = \hat{c}_i$; and the de-quantizer retrieves the corresponding reconstruction value given a codeword, i.e, $\mathcal{D}(\hat{c}_i) = y_i$.

The design of a quantizer usually involves dividing the source input range into a number of consecutive non-overlapping intervals, assigning an output level to each interval, and choosing the codewords for every output values.

Example 4.3.1.1. *An input signal source, $f(t) = 2 \sin 2\pi t$ is sampled every 0.05 second. To quantize this particular input source, one of the two scalar quantizers, namely midrise quantizer and midtread quantizer, with encoder and decoder mappings shown in Figure 4.1 can be used. (The difference of these two quantizers is further discussed in the next section.) Two possible codebooks for these quantizers are shown in Table 4.1.*

The performance of a quantizer is measured by its *distortion* and *rate*. The distortion, or more often the average distortion is measured as the difference between the input and the output values. The *mean-squared quantization error*, denoted by σ_q^2, is often used for this purpose. Assume we have an input modelled by a random variable X with PDF $f_X(x)$, and a quantizer Q with K intervals identified with decision boundaries $\{b_i\}_{i=0}^{K}$. The mean squared quantization error σ_q^2 is then given by [Sayood 2005]

$$\sigma_q^2 = \sum_{i=1}^{K} \int_{b_{i-1}}^{b_i} (x - Q(x))^2 f_X(x) dx. \tag{4.4}$$

If we use fixed-length codewords, the rate of a quantizer is simply $R = \lceil \log_2 K \rceil$, where K is the number of quantizer outputs. However, if variable-length codes are used, and l_i is the length of the code $\hat{c}_i \in \mathcal{C}$, then the rate is

$$R = \sum_{i=1}^{K} l_i \int_{b_{i-1}}^{b_i} f_X(x) dx. \tag{4.5}$$

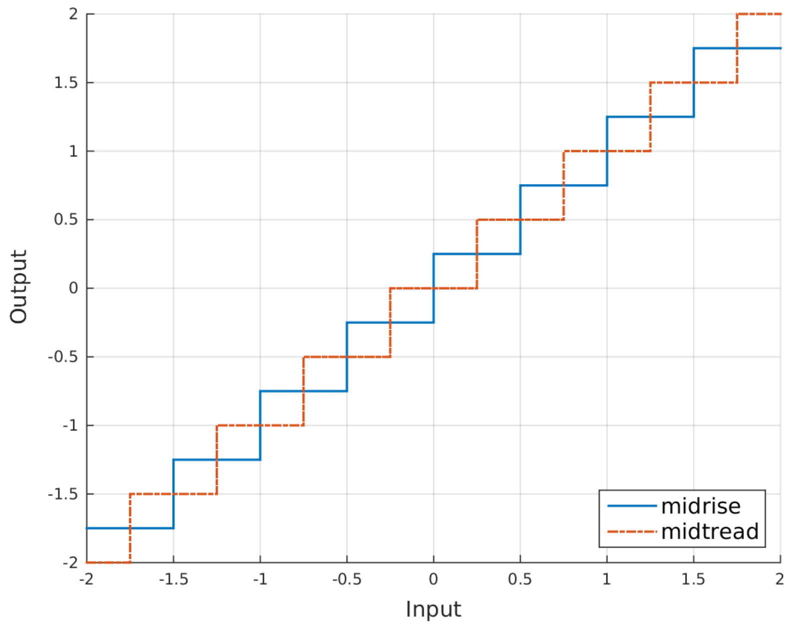


Figure 4.1: The input-output mappings for two different scalar quantizers.

Table 4.1: Decoder mappings for the two quantizers in Figure 4.1

Midrise		Midtread	
Codeword	Output	Codeword	Output
000	-1.75	0000	-2
001	-1.25	0001	-1.5
010	-0.75	0010	-1.0
011	-0.25	0011	-0.5
100	0.25	0100	0
101	0.75	0101	0.5
110	1.25	0110	1.0
111	1.75	0111	1.5
–	–	1000	2.0

– Not applicable.

4.3.2 Uniform and Nonuniform Quantizers

Uniform Quantizer For a uniform quantizer, the input range is divided into equal intervals, except possibly for the two outer intervals (see the midtread quantizer in Figure 4.1). The size of these intervals is referred to as step size, and denoted by Δ. The output values are often chosen as the midpoint of each interval, and they are spaced evenly. Thus, the step sizes for both input and output values are the same. The two quantizers in Example 4.3.1.1 are both uniform quantizers with $\Delta = 0.5$. The quantizer whose output values does not include zero is called *midrise* quantizer, and the one includes zero is *midtread* quantizer.

The simplest case is designing a uniform quantizer for a bounded and uniformly distributed input source. Suppose such an input X is distributed uniformly in the interval $[-X_{\max}, X_{\max}]$. Then the step size Δ for an K-level midrise uniform quantizer is simply

$$\Delta = \frac{2X_{\max}}{K}. \tag{4.6}$$

The output value for an interval $[(i-1)\Delta, i\Delta)$ in this case becomes $(2i-1)\Delta/2$, and the average distortion is

$$\sigma_q^2 = \sum_{i=1}^{K} \int_{(i-1)\Delta}^{i\Delta} \left(x - \frac{2i-1}{2}\Delta\right)^2 \frac{1}{2X_{\max}} dx. \tag{4.7}$$

The evaluation of (4.7) gives us $\sigma_q^2 = \Delta^2/12$. Further, the maximum possible error for a uniform quantizer is $\Delta/2$, and thus, the uniform quantizer minimizes the maximum error. It gives the uniform quantizer a useful robustness that maintains reasonably good performance for a wide variety of input signals [Gersho and Gray 1991].

Nonuniform Quantizer When the input source is not uniformly distributed, it is not practical to obtain the step size using (4.6). A quantizer with optimal mean squared quantization error can be found by minimizing (4.4) for a given number of output values. Such a quantizer is called nonuniform quantizer. Given the probability function $f_X(x)$, a nonuniform quantizer can be obtained by iteratively solving the following two equations,

$$y_j = \frac{\int_{b_{j-1}}^{b_j} x f_X(x)dx}{\int_{b_{j-1}}^{b_j} f_X(x)dx}, \tag{4.8}$$

$$b_j = \frac{y_{j+1} + y_j}{2}, \tag{4.9}$$

where y_j is a quantization output level, and b_j represents the boundary of an interval. Equation (4.8) implies that an input is assigned to the nearest output level, and (4.9) the optimal quantization levels are the centroids of an interval. The numerical solution of (4.8) and (4.9) usually proceeds iteratively solving one and then the other. The well known *Lloyed-Max algorithm*, or simply *Lloyed algorithm* [Lloyed 1982] is often used for this purpose. The Lloyed-Max algorithm can be adapted to run on different scenarios such as the one in which input PDF $f_X(x)$ is known, or a set of training samples is available without the input PDF. A detailed discussion in this regard can be found in [Taubman 2000].

4.3.3 Vector Quantization

VQ is a generalization of scalar quantization to vectors. Compared to scalar quantization, VQ allows much broader concepts, techniques, and applications due to the change from one dimension to multi-dimension. Vector quantization can be viewed as pattern matching where an input vector (or pattern) is approximated by one of the output vectors in a predetermined set. A vector quantizer Q maps k-dimensional vectors in the vector space $\mathbb{R}^k$ to a finite set of vectors $Y = \{\mathbf{y}_i, i = 1, 2, \ldots, K\}$. Each vector $\mathbf{y}_i \in \mathbb{R}^k$ is called a code vector or codeword, and the set Y is called the codebook. The performance of VQ can be measured using distortion and rate as in the case of scalar quantizer.

The application of VQ for the task of data compression usually involves the following four consecutive stages [Pickering and Ryan 2006].

Vector formation: Decomposition of the image into a set of vectors.

Training set generation: A subset of the input vectors is chosen as a training set.

Codebook generation: A codebook is generated from the training set using an iterative clustering algorithm.

Quantization: The codeword for the closest code vector in the codebook is found for each input vector and the codeword is transmitted along with the difference between the code vector and the original vector.

One of the widely used codebook design algorithms is the Linde-Buzo-Gray (LBG) algorithm, or the generalised Lloyd algorithm (GLA) [Linde et al. 1980]. The GLA algorithm takes as input a training set of N vectors of a certain dimension. K random vectors are selected, and using them as an initial codewords, the input space is partitioned into K subspaces. For each partition, a centroid is calculated by averaging all vectors in the partition; the original codewords are then replaced by these new centroids. This operation is iterated until there is no further improvement or the improvement falls under a threshold. A big challenge when applying GLA in hyperspectral data compression is its high computational complexity.

A VQ generated by GLA is locally optimal and has no structure. Locating a codeword for an input vector requires an exhaustive search in the codebook. Many fast search algorithms for VQ have been proposed. The method in [Kaukoranta et al. 1999] tries to reduce the number of distance calculations in GLA. It monitors the activities of the codevectors so that the updates are only performed on those active codevectors. A similar approach is introduced in [Qian 2004] to speed up the search process. A key observation in it is that most of the training vectors are placed in the same cluster as in the previous iteration, although the codevectors have been updated. To determine whether a full search is needed for a training vector, the distance between a training vector and the cluster is compared for the current and previous iterations. If the current distance is improved from the previous iteration, then a full search is not necessary.

Adaptive vector quantization adjusts dynamically the dimensions of the vectors in a codebook [Rizzo et al. 2005]. Alternatively, imposing a structure to the codebook is a practical method to simplify codebook design as well as to speed up the encoding.

Various hyperspectral data compression schemes based on predictive coding, transform coding, PCA, and vector quantization have been proposed. Efficient compression methods using these methods exist for 2D images. However, due to the different characteristics of spatial and spectral dimensions, simple extension of a compression method for 2D images does not necessarily perform well on 3-dimensional hyperspectral data. Further enhancement or adaptation is necessary for 2D methods to be 3-dimensional. The following sections provide a brief overview of background and techniques of hyperspectral data compression.

4.4 Predictive Coding

Predictive coding is a popular data compression technique used in both lossless and lossy coding. Sources such as images have correlations from sample to sample. This allows the prediction of a new sample based on its close past neighbours. Prediction exploits the correlation in spatial, spectral, or temporal dimensions, or context, to decorrelate the input signal or image. Rather than encoding the original sample, predictive coding encodes the difference between this sample and its prediction. If there is a large amount of correlation between neighbouring samples, the technique of predictive coding could lead to better bit rate.

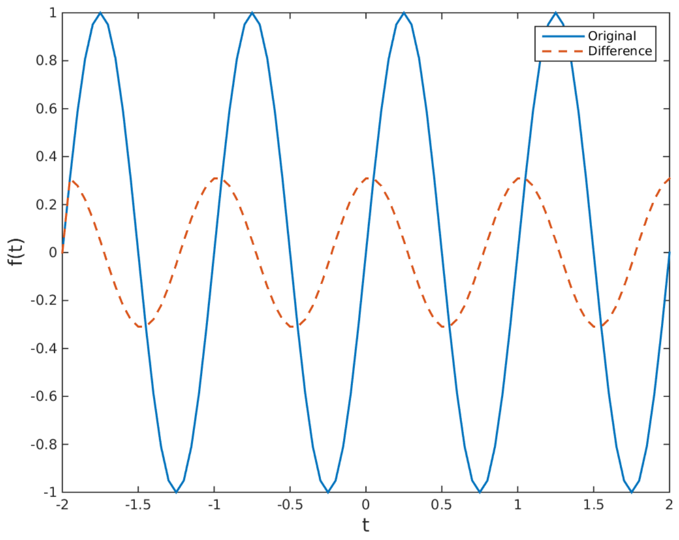


Figure 4.2: The comparison of dynamic ranges of the outputs for $f(t) = \sin 2\pi t$ and a simple predictive model of $f(t)$.

Example 4.4.0.1. *Consider an input source, $f(t) = \sin 2\pi t$, where it is sampled every 0.05 seconds. The dynamic range of the output of this source is $[-1, 1]$. Assume we predict each sample, except the first one, by its immediate previous sample, then the dynamic range of the sequence of errors (or differences) is reduced to $[-0.309, 0.309]$. Figure 4.2 shows the above claim. Further, if we use an 8-level uniform midrise quantizer for the original sequence, the step size will be 0.25, and this results in the range of quantization error of $[-0.125, 0.125]$. Using the same level of quantizer for the difference sequence, the step size will be 0.078, and the error range becomes $[-0.039, 0.039]$.*

Example 4.4.0.1 has shown the encoding of difference values uses fewer bits than the encoding of the original ones. However, the decoder only receives the quantized difference values, and use them to estimate the original signal. Since the quantization errors in the reconstructed sequence are accumulative, the error range becomes much larger than the one at the encoder. One technique to avoid this scenario is to take the difference between the current sample and the reconstructed previous sample. That is, the difference sequence is generated as

$$d_n = x_n - \hat{x}_{n-1}, \tag{4.10}$$

instead of

$$d_n = x_n - x_{n-1}, \tag{4.11}$$

where $\hat{x}_{n-1}$ is the reconstructed value for x_{n-1}. In this way, both the encoder and the decoder could use the same information. Moreover, the idea of using the immediate previous sample to predict the current one can be extended to the use of more than one previous samples. Given all the previously reconstructed samples $\hat{x}_{n-1}, \hat{x}_{n-2}, \cdots, \hat{x}_1$, a predictor function can be put as

$$p_n = f(\hat{x}_{n-1}, \hat{x}_{n-2}, \cdots, \hat{x}_1), \tag{4.12}$$

where p_n is the predicted value for the current sample x_n. For instance, a linear predictor can be constructed as

$$p_n = \sum_{i=1}^{N} a_i \hat{x}_{n-i}, \tag{4.13}$$

where $a_i, i = 1, 2, \cdots, N$, is the predictor coefficients, and N is the order of the predictor, which determines the number of past samples used in the prediction of the current one. A simple approach to find the predictor coefficients in (4.13) is to replace $\hat{x}_{n-i}, i = 1, 2, \cdots, N$, with $x_{n-i}, i = 1, 2, \cdots, N$. An analytical solution can then be obtained by minimizing the mean squared prediction error.

4.4.1 DPCM

The popular encoding system DPCM [Roger and Cavenor 1996; Wu and Memon 1997] is built upon the basic idea presented above. For instance, the DPCM for images usually consists of a spatial prediction followed by an entropy coder for the difference images. If the neighbourhood pixels are causal, i.e. those pixels are already known to both the encoder and decoder, then a predictor function can be designed such that the prediction error can be minimised. It is usually not feasible to find a single model that fits all local structures within an input source. This is especially the case for higher dimensional input sources. In practice, multiple predictors are employed to capture different structures, or to adapt the predictors to the incoming data in a progressive manner. The adaptive DPCM (ADPCM) [Roger and Cavenor 1996] takes the latter approach, where the predictors are updated as long as new data becomes available.

For hyperspectral data, two types of correlation exist. One is the spatial correlation between neighbouring pixels within one band; the other is the spectral correlation between neighbouring bands. Thus, a linear combination of values from pixels that are spatially and spectrally adjacent to the current pixel is typically used as the prediction. Roger and Cavenor [1996] developed a lossless compression of AVIRIS images based on ADPCM. A number of predictors, including both fixed and adaptive ones, are designed and evaluated. The overall compression scheme consists of two stages: predictive decorrelation and residual encoding. In the predictive decorrelation stage, ADPCM uses a selected predictor to compute the prediction value for a pixel being encoded. Here, different predictors are used to exploit spatial (intraband) correlation, spectral (interband) correlation, or the combination of both correlations. The residual images are then entropy coded using various variable length codes. The best performance is given by the adaptive predictors which exploit interband correlation or both intraband and interband correlations.

4.4.2 CALIC

Context-based adaptive lossless image codec (CALIC) is an efficient lossless coding scheme, which is also based on predictive coding [Wu and Memon 1997]. CALIC encodes and decodes a 2D image in raster scan order with a single pass through the image. It uses a neighbourhood of pixel values taken from the previous two rows of the current pixel being encoded. CALIC operates in two modes: binary and continuous tone. The latter is designed for compressing natural continuous tone images. In this mode, CALIC has four major components: gradient-adjusted prediction (GAP), context selection and quantization, context modelling of prediction errors, and entropy coding of prediction errors. The first three components are briefly explained below.

GAP: The GAP is a simple, adaptive, nonlinear predictor. It weights the neighbouring pixels of the current one, say a pixel p at 2D spatial position (i, j), according to the estimated gradients of the intensity function near (i, j). These usually include the gradients of the intensity function in the horizontal, or vertical directions. They are used to detect the edges in the input image, so that necessary adjustments can be made in the prediction.

Coding context selection and quantization: In order to further adjust the prediction from GAP, an error energy estimator is defined and used to separate the error distribution into classes of different variances. Then a quantizer is designed on the error energy estimators to minimize the errors.

Context modelling of prediction errors: This stage is to further refine prediction by exploiting higher order structures, such as texture patterns, that gradients cannot capture fully.

The binary mode is for images or sub-images containing very few widely separated grey levels where predictive coding fails. A unique feature of CALIC is the use of large number of modelling contexts to condition a nonlinear predictor and adapt the predictor to varying source statistics.

The 3-dimensional adaptation of CALIC, interband CALIC or 3D-CALIC, is introduced in [Wu and Memon 2000]. 3D-CALIC incorporates interband prediction techniques into the framework of 2D (or intraband) CALIC. Given a reference band B_{ref}, and a particular pixel y in the current band, the correlation is measured between the causal neighbourhoods of y and its counterpart pixel, x, in B_{ref}. If the correlation is high, we can use the neighbouring pixels of x to predict y. Moreover, such a predictor can be adjusted in the case of sharp edges. On the other hand, if the correlation is low or below a threshold, the prediction is carried out using the intraband CALIC.

CALIC-based lossless or near-lossless compression for hyperspectral data, called M-CALIC, is proposed in [Magli et al. 2004] to optimize 3D-CALIC for this particular type of data. Since the wavelength resolution is high in hyperspectral data, the correlation between bands is also sufficiently high. Thus, it is possible to always use interband prediction. The reference band in M-CALIC is set as the linear combination of the two previous bands.

4.5 Transform Coding

In transform coding, a reversible, linear transform is used to map the original data to a set of transform coefficients, which are quantized and entropy coded. Figure 4.3 shows the modularisation of the essence of transform coding. First, an invertible linear transform of the source vector $\mathbf{z}$ is computed to obtain the coefficient vector $\mathbf{y} = \mathbf{T}\mathbf{z}$. Each component of $\mathbf{y}$ is then scalar quantized to produce a set of quantizer indexes. Finally, the quantizer indexes are entropy coded. At the decoder, the entropy decoder $\mathbf{Y}^{-1}$ recovers the quantized indexes. The dequantizer $\boldsymbol{\beta}^{-1}$ produces an estimate vector $\mathbf{y}'$ of the transform coefficients. To complete the reconstruction, a linear transform $\mathbf{U}$ is applied to $\mathbf{y}'$ to produce the approximation $\mathbf{z}'$. Typically $\mathbf{U} = \mathbf{T}^{-1}$. Many transform coding systems based on *discrete Fourier*, *discrete cosine*, *discrete wavelet* transforms, and others have been studied extensively.

In general, the input vectors are multiplied by a set of *basis* vectors to produce the transform coefficients. In the context of data compression, the following properties in the transform domain are considered to be appealing [Vetterli and Kovacevic 1995]. Firstly, the transform coefficients are not correlated, which is equivalent to diagonalising the autocovariance matrix of the signal. Secondly, the transform coefficients are more suitable for quantization. In particular, we desire the transform coefficients to consist of a significant number of small magnitudes that can be quantized or discarded with little distortion. That is, only a small number of coefficients is sufficient to carry majority of the information in the original data. Lastly, there usually exists fast algorithms for transform decomposition.

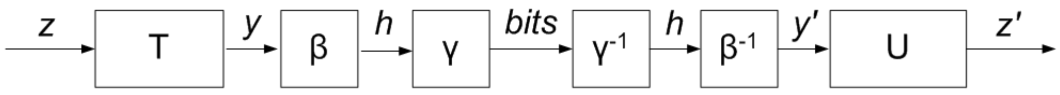


Figure 4.3: The architecture of a basic transform coding (Source: [Goyal 2001]).

4.5.1 Karhunen-Loève Transform

KLT is considered to be the most efficient transform in theory in that i) completely decorrelates the signal, and ii) optimally compacts signal energy. Given an input source $\mathbf{X}$ consisting of random variables, the KLT transform matrix $\mathbf{T}$ is found as the eigenvectors of the covariance matrix of $\mathbf{X}$, denoted by $\mathbf{C}_X$. This is typically done by singular value decomposition (SVD). As a result, the transformed vector $\mathbf{Y} = \mathbf{T}^T\mathbf{X}$ has uncorrelated components, i.e., $\mathbf{Y}$ has a diagonalized covariance matrix.

KLT, as with any other transform, may be understood as a decomposition of the input source as a linear combination of prototypes. These prototypes in the case of KLT are referred to as 'principle components'. We often desire keeping only a subset of the coefficients, such as in data compression, or dimensionality reduction. For this, we should select the first P coefficients in order to minimize the mean squared error (MSE) for the approximation. In fact, it can be shown that amongst all linear transforms, if we keep only m coefficients, KLT gives the minimum MSE. This is due to the optimal energy packing property of KLT, where it says the first few coefficients contain most of the energy of the transformed input.

PCA, being almost identical to the KLT, allows dimensionality reduction by retaining in the KLT transform matrix $(M \times M)$ only those eigenvectors corresponding to the P $(P < M)$ largest ones. Thus, the dimension in the transform domain is reduced from M to P. For hyperspectral image compression, both KLT and PCA are often used for spectral decorrelation or dimensionality reduction in conjunction with other coding strategies. PCA is employed in JPEG 2000 to provide spectral decorrelation as well as dimensionality reduction [Du and Fowler 2007]. This approach yields a better performance in terms of rate distortion and information preservation compared to the wavelet based coder (further discussion in Chapter 5). Lee et al. [2002] compared several approaches with different spectral compression methods followed by spatial compression using the JPEG 2000 coding. KLT, DCT, wavelet coding and spectral DPCM are considered for the spectral compression.

4.5.2 Discrete Cosine Transform

Both KLT and PCA are source dependent. DCT is a close approximation to KLT, however, it has fast algorithms for implementation and its transform matrix is not dependent on the source output. It expresses a finite number of data points in terms of a sum of cosine functions oscillating at different frequencies. Most of the signal information in the transform domain tends to be concentrated in a few low-frequency components. Due to this strong energy compaction property, DCT is often considered as an approximation to KLT. The computational simplicity of DCT is another appealing factor, with fast $O(M \log M)$ algorithm available for it as opposed to the $O(M^2)$ for KLT. DCT calculates the transform coefficients for 1D block signal $\mathbf{x}$ with size U as

$$\mathbf{y}_u = c_u \sum_{i=0}^{U-1} \mathbf{x}_i \cos \frac{\pi(2i+1)u}{2U}, \tag{4.14}$$

for $u = 0, 1, \cdots, U-1$, and

$$c_u = \begin{cases} \sqrt{\frac{1}{U}} & u = 0; \\ \sqrt{\frac{2}{U}} & u \neq 0. \end{cases} \tag{4.15}$$

The 2D DCT is the separable extension of 1D DCT, and computes the coefficients within each UV block as

$$\mathbf{Y}_{u,v} = c_u c_v \sum_{i=0}^{U-1} \sum_{j=0}^{V-1} \mathbf{X}_{i,j} \cos \frac{\pi(2i+1)u}{2U} \cos \frac{\pi(2j+1)v}{2V}, \tag{4.16}$$

for $u = 0, 1, \cdots, U-1, v = 0, 1, \cdots, V-1$, and c_u, c_v are defined same as (4.15).

DCT has been adopted for still image compression standard such as JPEG, as well as video compression standard such as MPEG. Typically, when the DCT is applied to correlated data, the coefficients corresponding to the low frequency DCT basis vectors have larger magnitudes, and the coefficients corresponding to the high frequency basis vectors have much smaller magnitudes. The quantization of these coefficients usually results in the coefficients for higher frequencies being quantized to zero. One shortcoming of DCT is that it is a block based transform, and this often leads to poor compression performance in the block boundaries such as blocking effects.

DCT can be applied in hyperspectral image compression in combination with other techniques such as vector quantization. Pickering and Ryan [2001] investigated applying DCT in spectral domain in order to compress the residual data produced by the mean-normalized vector quantization (M-NVQ). One-dimensional DCT is performed on each spectral vector of residual data produced by the M-NVQ process. This combination results in some improvement in terms of compression ratio over using M-NVQ alone. For low distortion levels, further improvements in compression ratio is obtained in [Baizert et al. 2001] by replacing the spatial M-NVQ by a spatial 2D DCT. A comprehensive experiment is carried out in [Penna et al. 2007] to compare the coding efficiencies of several combinations of spatial and spectral transforms. These transforms include DCT, wavelet, wavelet packet transforms and KLT. A low-complexity KLT is also proposed, and integrated with the JPEG 2000 standard to evaluate its performance. The combination of sparse representation and DCT is applied for video compression in [Kang et al. 2013]. A dictionary is adaptively trained to contain featured patterns of residual signals so that a high portion of energy in a structured residual can be efficiently coded via sparse coding. DCT is then applied to the remaining signal to improve the coding efficiency.

4.5.3 Subband Coding

It should be emphasized at this point that the transforms discussed previously are block based transforms. That is, the source output is divided into blocks of certain size, each block is treated independently in the transform process. The result of such process is that the source output is decomposed into different frequency bands. Thus, the transform coefficients display varying statistical characters, and can be utilized efficiently in the following stages of coding. A drawback of such approach lies in the arbitrary division of a source output into blocks, and the resulting coding artifacts at the block boundaries. A popular strategy to generalize the block transform to the entire source output is the ‘sliding window’ transform, or *subband* transform. Under this approach, we decompose the original sequence into a number of sequences, so that each sequence carries different characteristics. This is usually implemented using discrete time filters, which operate on a sequence of discrete values that are commonly the outcome of sampling process of a continuous source output.

Filters are used to isolate certain frequency components. In a simple case, this involves taking the weighted sum of current and past inputs to the filter. This process is called *analysis*, and can be given as

$$\mathbf{y}_t = \sum_{i=0}^{U-1} \mathbf{a}_i \mathbf{x}_{t-i}, \tag{4.17}$$

where $\mathbf{x}_t, t = 0, 1, \cdots, N-1$, is the input sequence, $\mathbf{y}_t, t = 0, 1, \cdots, N-1$, is the output sequence, $\mathbf{a}_i, i = 0, 1, \cdots, U-1$, are the filter coefficients, and U is often called the number of *taps* in the filter. Filters that only allow the components below a certain frequency, f_0, is called low-pass filters; in contrast, filters that block all frequency components below f_0 are called high-pass filters. If the filter coefficients are selected carefully, the inverse transform (or synthesis) of (4.17) almost always exists and takes the similar form as (4.17). The same decomposition approach can be applied recursively on the subsequences.

Using different filters on the same input sequence, different output sequences are generated. From (4.17), we observe that these sequences have the same length. If we send these outputs without further processing, we will end up sending more samples than the original sequence has. This is avoided by sending only a part of each output sequence, i.e., a subsequence or subband. For instance, using a pair of low-pass and high-pass filters, two output sequences will be generated. If we take half of the samples from each output sequence, then the total number of samples in the two subsequences will be the same as that of the original sequence. This process is referred to as downsampling. On the decoder side, the inverse of downsampling becomes necessary, and such a process is called upsampling.

In a basic subband coding system, the source output is passed through analysis filter bank first, the output sequences are then downsampled. This is followed by the quantization and entropy coding of subbands, where a suitable encoding scheme, such as DPCM, ADPCM, can also be applied. The quantized and coded coefficients are used for the reconstruction of the original sequence at the decoder. The inverse operations of entropy coding and quantization recover the subbands. These decoded samples in the subbands are then upsampled to pass through the synthesis filter bank. The output sequences are then combined to give the final reconstruction.

As the low-pass subband usually contains the majority of source output energy, the decomposition can be carried out for this subband only. This leads us to multiresolution transform, where only the low-pass subband is further decomposed at each level.

4.5.4 Discrete Wavelet Transform

Wavelets provide a signal representation in which some of the coefficients represent long data lags corresponding to a narrow band, low frequency range, and some of the coefficients represent short data lags corresponding to a wide band, high frequency range [Shapiro 1993]. Wavelet theory studies filter design methods such that the filter bank is perfectly reconstructing, and both the low-pass and high-pass filters have finite impulse responses [Vetterli and Kovacevic 1995]. DWT is often used to find a compact multiresolution representation of signals, including images.

One-dimensional DWT decorrelates a signal by splitting the data into two half-rate subsequences, i.e., low- and high-frequency half-bands, carrying information on the approximation and detail of the original signal, respectively. The two-channel decomposition can be repeated respectively on the low-pass and high-pass subband samples of a previous filtering stage to provide a multi-resolution decomposition of the input signal. However, in most DWT decompositions, only the low-pass output is further decomposed, and we refer to this type as *dyadic* decomposition. Figure 4.4 shows a 1D, three-level dyadic wavelet decomposition by means of a filter bank scheme with low-pass and high-pass filters denoted as $g[n]$ and $h[n]$, respectively. At each level of the decomposition, the low-pass filter preserves the low frequencies of a signal while eliminating the high frequencies, thus resulting in a coarse approximation of the original signal. The high-pass filter, conversely, preserves the high frequencies of the signal such as edges, texture and detail, while removing the low frequencies. In summary, the high-pass samples in the tree-structured transform are wavelet transform coefficients, and the low-pass samples are of vanishing importance when the number of decomposition levels becomes large. However, practical DWT implementations must include the low resolution subband samples in the reconstruction.

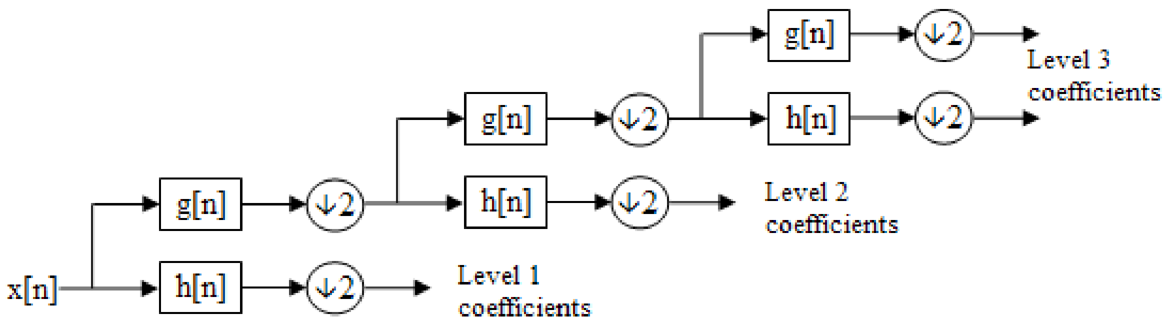


Figure 4.4: One-dimensional, 3-level wavelet decomposition. $g[n]$ and $h[n]$ denote low-pass and high-pass filters, respectively. The circles denote downsampling by a factor of 2 (Source: [Vetterli and Kovacevic 1995]).

For two-dimensional images, wavelet transform can be applied using two 1D DWT, one in the horizontal direction, and the other in the vertical direction. This approach is known as separable DWT. At the first level, an image is decomposed into four subbands, namely, $\mathrm{LL}_0, \mathrm{LH}_0, \mathrm{HL}_0$ and HH_0. To obtain the next coarser scale of wavelet coefficients, the subband LL_0 is further decomposed and subsampled at the second stage. The passband structure after two levels of 2D wavelet decomposition is shown in Figure 4.5. LL_1 is the second level coarse approximation of the original image. HH bands contain high frequency contents in both horizontal and vertical directions. They respond primarily to diagonal features. LH bands contain low frequency content in the horizontal direction and high frequency content in the vertical direction. They respond most strongly to the horizontal edges and line segments. Similarly, HL bands respond most strongly to the vertical ones. Figure 4.6 shows the three-level DWT of image 'Lena'.

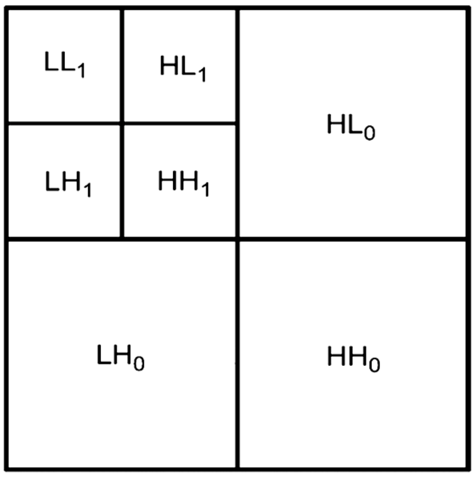


Figure 4.5: Two-dimensional, 2-level wavelet decomposition.

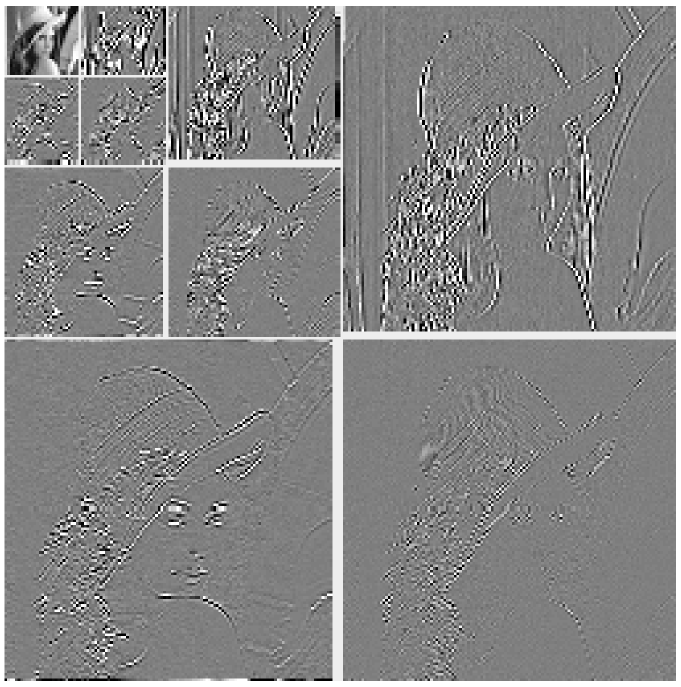


Figure 4.6: Three levels of wavelet decomposition for the image 'Lena' using the Daubechies wavelets.

4.5.5 Bitplane and Significance-Map Coding

An important characteristic of wavelet-based compression schemes is the support for progressive transmission and embedded coding. Progressive transmission allows the successive reconstructions of the image, and it is typically implemented through embedded coding. Using embedded coding for a dataset, any prefix of length l bits of M-bit ($0 < l \leq M$) coding is also a valid coding of the dataset; when the value of l increases, the quality upon reconstruction improves. Thus, embedded coding allows an encoder to terminate the encoding at any point when a target rate or distortion metric is met. DWT decomposes the source output into subbands, where these subbands carry information of varying importance. As such, some coefficients are more important than the others, and can be used to obtain an approximation of the original signal. Further, the idea is carried to the bit level such that the importance of every bit in a coefficient should also be differentiated. Specifically, to transmit a set of coefficient using embedded coding, we transmit the first most significant bit (MSB) of all coefficient magnitudes, then the second MSB of all coefficient magnitudes, and so on. This strategy is referred to as the bitplane coding.

Bitplane coding is often implemented by iteratively running two passes, namely significance pass and refinement pass, for each bitplane. Given a threshold t and using the sign-magnitude representation of the coefficients, the significance status s of a coefficient c_p at position p is determined as

$$s_p = \begin{cases} 1\,(\text{significant}), & |c_p| \geq t, \\ 0\,(\text{insignificant}), & |c_p| < t. \end{cases} \tag{4.18}$$

The significance pass determines the values of s for all the coefficients using the current t. The result of this pass is often referred to as the significance map in the case of images. The refinement pass then produces the approximations to those significant coefficients for the current bitplane. After each iteration, the threshold is decreased by half and the process is repeated for the next bitplane.

4.5.6 Set Partitioning in Hierarchical Trees

The set partitioning in hierarchical trees (SPIHT) [Said and Pearlman 1996] is a widely used wavelet-based coding scheme of images, which is based on embedded zerotree wavelet (EZW) coding introduced in [Shapiro 1993]. EZW exploits the characteristic that the wavelet coefficients in different subbands represent the same location in the image. It is feasible to use a coefficient in the smaller subband to represent those in larger subbands. For instance, a ten-band wavelet decomposition in Figure 4.7 shows the spatial relationships among wavelet coefficients in different subbands. The coefficients a in Band I, a_1 in Band II, $a_{11}, a_{12}, a_{13}, a_{14}$ in Band V, and so on, roughly represent the same spatial location at different scales. This spatial relationship is represented using the data structure *zerotree* in EZW. That is, the coefficient at the coarser level is the parent node in the zerotree, and those corresponding to the same location, but at finer level subbands become the children. Given a threshold, a coefficient is said to be an element of a zerotree if itself and all its descendants are insignificant with respect to the threshold, i.e., the magnitudes of the coefficients are less than the threshold. Thus, if a parent node is determined to be a zerotree root, then the descendant coefficients do not need to be encoded. Using self similarity, zerotree coding provides an efficient way to reduce the cost of encoding the significance map.

One important feature in SPIHT is that it sorts and transmits the transform coefficients according to their significance. This can be solved by sorting the coefficients by the MSB. The SPIHT,

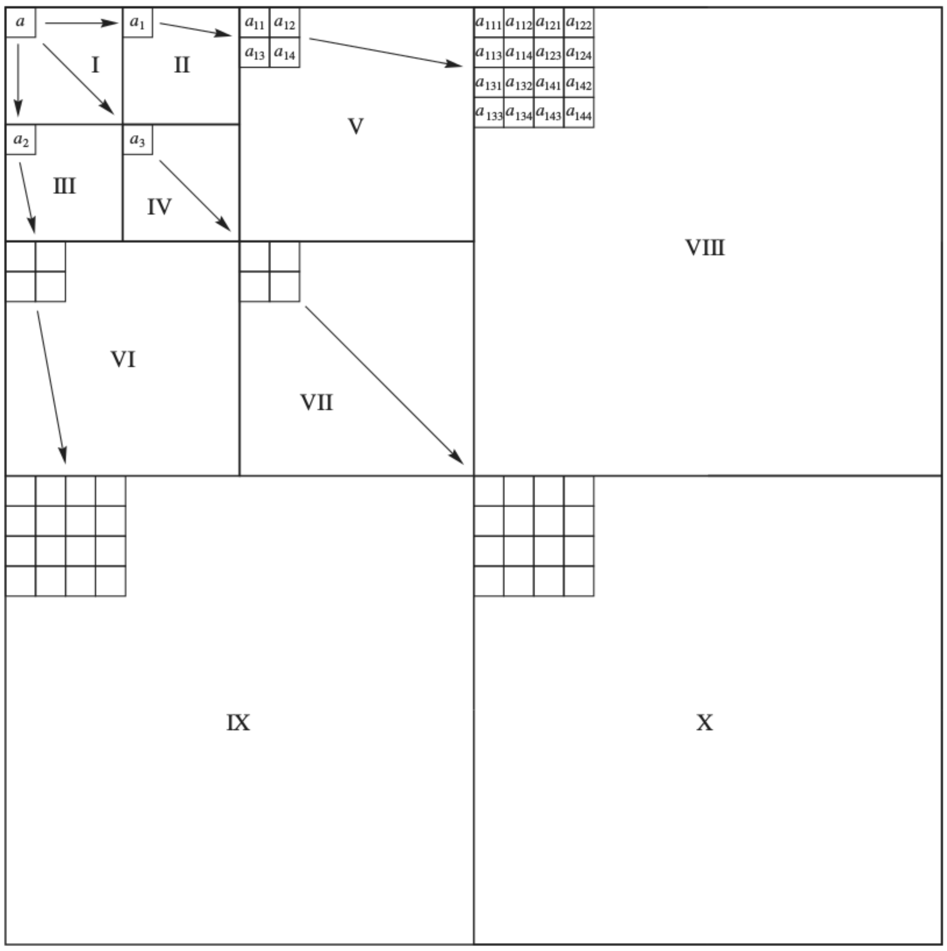

a
a_1
a_2
a_3
I
II
III
IV
a_{11} a_{12} a_{13} a_{14}
a_{111} a_{112} a_{121} a_{122}
a_{113} a_{114} a_{123} a_{124}
a_{131} a_{132} a_{141} a_{142}
a_{133} a_{134} a_{143} a_{144}
V
VI
VII
VIII
IX
X

Figure 4.7: A ten-band wavelet decomposition (Source: [Sayood 2005]).

however, solves this problem without explicitly sorting the coefficients. It uses the *set partitioning sorting algorithm* to divide the set of pixels into partitioning subsets and the magnitude test is performed on each subset. If a subset is insignificant, then all the coefficients in that subset are insignificant. On the other hand, if a subset is significant, then it is divided into new subsets. This set division process continues until the magnitude test is done to all single coordinate significant subsets in order to identify the significant coefficients. The set partitioning rule is defined in a way such that the subsets expected to be insignificant contain a large number of coefficients, and the subsets expected to be significant contain only one coefficient. SPIHT realises this idea based on a tree structure, called *spatial orientation tree*. The spatial orientation tree exploits two properties in the transform coefficients, especially the wavelet transform coefficients. One is that the low-frequency components usually contain most of a decomposed signal; the other is that there is a spatial similarity between subbands, and the magnitude of the coefficients tends to decrease along the same spatial orientation towards the finer scale subbands. An example of spatial orientation tree is shown in Figure 4.8. The tree is defined in such a way that each node has either no offspring or four offspring, which always form a group of 2×2 adjacent pixels.

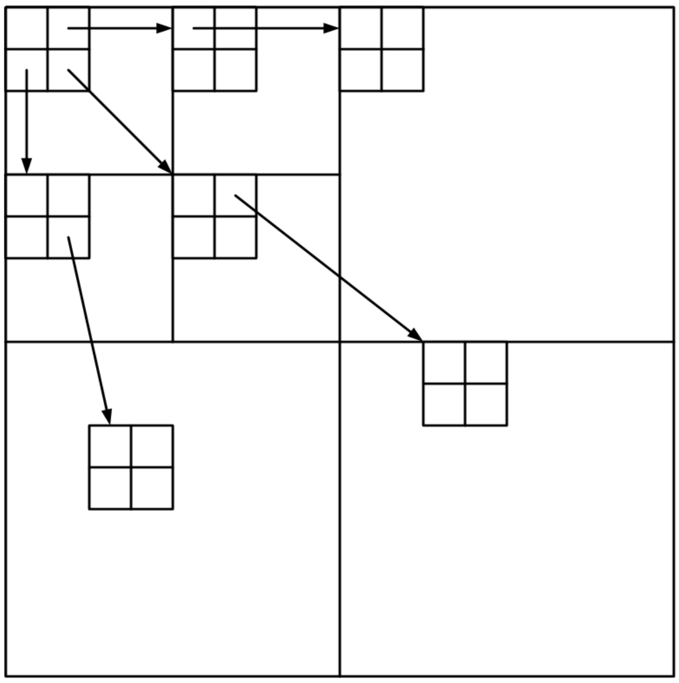


Figure 4.8: The spatial orientation tree for the dyadic wavelet decomposition (Source: [Shapiro 1993]).

The spatial orientation tree in SPIHT is partitioned into four types of sets as follows.

- $\mathcal{O}(i, j)$ Coordinates of all offspring of the node (i, j);
- $\mathcal{D}(i, j)$ Coordinates of all descendants of the node (i, j);
- $\mathcal{H}$ Coordinates of all root nodes;
- $\mathcal{L}(i, j) = \mathcal{D}(i, j) - \mathcal{O}(i, j)$.

Further, the algorithm uses three ordered lists, namely, list of insignificant sets (LIS), list of insignificant pixels (LIP), and list of significant pixels (LSP). The LSP and LIP store the coordinates of coefficients (or pixels), while the LIS represents either $\mathcal{D}(i, j)$ (type A) or $\mathcal{L}(i, j)$ (type B).

In each pass, the SPIHT algorithm consists of two main steps: sorting and refinement. The sorting step processes the lists LIP and LIS. First, it examines each element in LIP of its significant status. If an element is significant (it is greater than the threshold), the bit 1 is transmitted followed by another bit indicating the sign. Otherwise, it transmits a bit 0. After the list LIP is processed, the elements in the LIS are processed. If the set at an entry in the LIS is significant, the bit 1 is transmitted, otherwise, the bit 0 is transmitted. When the set at an entry is significant, different treatments are followed depending on the set types. If the set type is A, we examine the coefficients at coordinates in $\mathcal{O}(i, j)$. For each coefficient that is significant, the bit 1 is transmitted followed by a sign bit. The corresponding coordinate is then moved to the LSP. If a coefficient is insignificant, the bit 0 is transmitted, and the corresponding coordinate is moved to the LIP. At this stage, we processed all the coefficients corresponding to the coordinates in $\mathcal{O}(i, j)$ and removed them from the $\mathcal{D}(i, j)$, what is left now is $\mathcal{L}(i, j)$. If the set $\mathcal{L}(i, j)$ is not empty, we move (i, j) to the end of the LIS as an entry of type B. If it is empty, we remove the entry from the LIS. If the set type is B, each coordinate in $\mathcal{O}(i, j)$ is added to the end of the LIS as an entry of type A. The entry (i, j)

is then removed from the LIS. Note that each newly added entry in the LIS must be processed in the current pass. In the refinement step, for each entry (i, j) in the LSP, except those being added during the sorting step at the current pass, we output the most significant bit of the corresponding coefficient.

The SPIHT coding works by grouping image pixels in spatial orientation trees. The hierarchical structure in such trees expose the similarity across the subbands at the same spatial locations. The similar ideas are realized in other coding schemes as well. One such scheme is the embedded, block-based coding algorithm set partitioned embedded block (SPECK) [Pearlman et al. 2004]. Instead of using the spatial orientation tree, the SPECK algorithm makes use of two types of sets, namely $\mathcal{S}$ and $\mathcal{I}$. $\mathcal{S}$ is a rectangular region of an image, and $\mathcal{I}$ is obtained by removing a small square region at the top left of a larger square. A set $\mathcal{S}$ in the LIS is first tested for significance. If it is significant, the set is partitioned into four subsets. Each of these subsets is recursively tested for significance. That is, if significant, it is partitioned again, and if insignificant, it is added to the LIS. Every significance test result is then sent to the bitstream. Such a partitioning process is called quadtree partitioning, and it is capable of quickly identifying the areas of high energy, as well as locating groups of coefficients as insignificant. On the other hand, if found to be significant, the set $\mathcal{I}$ is partitioned using the octave band partitioning. The scheme partitions $\mathcal{I}$ into four sets, three of them being type $\mathcal{S}$, and one being type $\mathcal{I}$. The purpose of such partitioning is to exploit the hierarchical pyramidal structure of the wavelet decomposition, where the energy tends to be more concentrated in the top most levels of the pyramid.

4.5.7 3D Wavelet Based Compression of Hyperspectral Images

For hyperspectral imagery, the 2D DWT of Figure 4.5 is extended to 3 dimensions to take into consideration the spectral dimension. Similar to the 2D dyadic DWT, a 3D DWT can be implemented in separable fashion. That is, 1-dimensional DWT is applied for the spatial row, column, and spectral dimensions separately. Another 3D DWT approach, called wavelet-packet transform, first performs a separable 2D DWT for each spectral band, then follows a 1D decomposition in the spectral dimension. Fowler and Rucker [2007] shows that wavelet-packet transform typically yields more efficient coding for hyperspectral images than the dyadic decomposition does. This is due to the flexibility in exploiting the spatial and spectral dimensions differently.

2D SPIHT has been extended to 3D SPIHT for the task of video coding in [Kim et al. 2000]. In particular, a group of continuous video frames is analysed as 3D data cube. The 3D subband structure is obtained by decomposing the group of frames in the temporal and spatial dimensions, as well as subsampling. Within the resulting subband structure, there exists not only spatial similarity inside each frame across different scale, but also temporal similarity between neighbouring two frames. These similarities are further exploited using the 3D SPIHT algorithm. Similar to the 2D SPIHT, the 3D SPIHT also consists of sorting and refinement as the two main stages. The difference is that the pixels are now grouped in 3D rather than 2D trees. For this, a 3D spatial-temporal orientation tree is defined to describe the parent-offspring relationships in 3D SPIHT. A straightforward extension in this case is to form the nodes with 3D pixel blocks in the spatio-temporal orientation tree. For instance, in the case of using dyadic decomposition, we can formulate a node of $2 \times 2 \times 2$ pixels. The root nodes (at the highest level of the pyramid) have one pixel with no descendants, and each of the other seven points to a node with 8 offspring at the immediate next coarser level, where the parent and offspring nodes correspond to the same location at different scales. Once the spatio-temporal orientation tree is set up, the 3D SPIHT sorts the coefficients according to the magnitude (sorting), and refine the bit plane by adding necessary bits (refinement).

Dragotti et al. [2000] applied 3D SPIHT in the compression of multispectral images. Unlike the video frames, where large number of continuous frames are available, multispectral images have only a few spectral bands. Thus, applying the DWT in both the spatial and spectral dimensions does not yield good results. For this reason, Dragotti et al. [2000] applied a KLT in the spectral dimension while 2D DWT applied in the spatial domain. The transform coefficients are then organized in a 3D hierarchical structure to run the SPIHT algorithm. A second technique proposed in the same work is to apply VQ for the spectral coefficients. More precisely, after the 2D DWT of all the bands in the image is performed, coefficient vectors are formulated in the spectral dimension, and they are vector quantized. The SPIHT algorithm is then run on the vectors, where the magnitude of a scalar is replaced by the norm of a vector.

Hyperspectral images consist of hundreds of bands, and thus, the interband correlations can be exploited efficiently using DWT. In such a case, the 3D SPIHT of [Kim et al. 2000] can be applied to compress the hyperspectral images [Tang et al. 2003b; Tang and Pearlman 2006]. Moreover, it has been shown in some works that the spectral and spatial redundancies should be treated differently in order to achieve efficient compression of hyperspectral images. A popular approach involves the combination of one-dimensional spectral decorrelation followed by the JPEG 2000 being applied for the spatial decorrelation, rate allocation, and entropy coding. PCA is employed in JPEG 2000 to provide spectral decorrelation, as well as spectral dimensionality reduction [Du and Fowler 2007]. Penna et al. [2007] investigated a number of combinations of different transforms, such as wavelets, wavelet packets, DCT, and KLT, in exploiting the spatial and spectral redundancies for hyperspetral image coding. For the spectral decorrelation followed by a 2D DWT spatial decomposition, the KLT gives the best transform in terms of peak signal to noise ratio (PSNR), and the DWT performs better than the DCT at most of the bit rate regions. To exploit the superior decorrelation capability of KLT, a low-complexity KLT is proposed in [Penna et al. 2007] to reduce the computational cost involved.

Part 2 of the JPEG 2000 allows the multiple component transformations, where the standard provides two general approaches to achieve this goal. It may include a decorrelation transform, such as KLT or PCA, or a linear predictive transform. The second approach involves using a default or user specified one-dimensional DWT in the component direction.

4.6 JPEG 2000

The JPEG 2000 [Christopoulos et al. 2000; Rabbani and Joshi 2002; Taubman and Marcellin 2002] still image compression standard has been designed and developed to utilize the advances in image compression technology, and meet the demand of fast evolving markets and applications. The standard is based on discrete wavelet decomposition. It consists of multiple parts dealing with a variety of applications, from basic image compression to medical imaging and motion pictures. In this section, we attempt to have a brief look at the key components of JPEG 2000 Part 1, the core image coding system. A typical JPEG 2000 encoder consists of building blocks such as preprocessing, DWT, quantization, arithmetic coding, and bit stream organization. More details are given below regarding each of these.

4.6.1 Pre-Processing

The pre-processing deals with partitioning the input image into non-overlapping blocks called tiles. Tiles are treated independently of each other in the subsequent processes. For unsigned sample values in each tile component, the DC level shifting is performed by subtracting a fixed value (i.e., the component depth) from each sample to make its value symmetric around zero. In the cases of multiple component image, a forward intercomponent transform is applied to reduce the correlation. Two such transform choices are available in Part 1, one is the irreversible colour transform (ICT), and the other is reversible colour transform (RCT). The forward ICT is the traditional RGB to $\mathrm{YC_bC_r}$ color transform, and it is defined as

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{4.19}$$

The inverse of (4.19) is

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0 & 1.402 \\ 1.0 & -0.34413 & 0.71414 \\ 1.0 & 1.772 & 0 \end{pmatrix} \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix}. \tag{4.20}$$

The forward RCT is defined as

$$Y = \left\lfloor \frac{R + 2G + B}{4} \right\rfloor, \quad U = R - G, \quad V = B - G, \tag{4.21}$$

and the inverse RCT is

$$G = Y - \left\lfloor \frac{U + V}{4} \right\rfloor, \quad R = U + G, \quad B = V + G. \tag{4.22}$$

4.6.2 DWT

Following the pre-processing stage, the DWT is performed on each of the tile components. Part 1 of the JPEG 2000 allows the power of 2 decompositions in the form of dyadic wavelet transform. The DWT can be irreversible or reversible. The irreversible DWT is implemented using the Daubechies 9/7 filter bank [Antonini et al. 1992], and the reversible DWT is implemented using the 5/3 filter bank [Gall and Tabatabai 1988]. The low-pass and high-pass analysis filter pair is often designed such that after downsampling the output of each filter by a factor of 2, the original signal can still be recovered exactly. We say such a filter satisfies the perfect reconstruction property. Both 9/7 and 5/3 filter banks satisfy this property. Note that although Part 1 of JPEG 2000 gives only two DWT filters, Part 2 of the standard allows arbitrary filter specifications.

The DWT in the JPEG 2000 is implemented using the *lifting scheme* [Daubechies and Sweldens 1998, and references therein]. The method reduces the requirements for memory, as well as the computational complexity. The wavelet coefficients computed using the lifting scheme are identical to those computed by the traditional filter bank convolution. Consider the one dimensional two channel wavelet transform. The lifting scheme starts with splitting the 1D input sequence into two subsequences, one consists of the even indexed samples, denoted by $\mathbf{d}_i^0$, and the other the odd indexed ones, denoted by $\mathbf{s}_i^0$. This step is also referred to as the 'lazy wavelet' transform.

The values in the two subsequences are then modified in a sequence of alternating prediction and updating steps. A prediction step involves predicting each odd sample as a linear combination of some even samples, and subtracting it from the odd sample to form the prediction error. An update step, on the other hand, updates the even samples by adding to them a linear combination of the already modified odd samples. These two steps are iterated a number of times, using different weights for the linear combination at each iteration. This can be put as [Rabbani and Joshi 2002]

$$\mathbf{d}_i^l = \mathbf{d}_i^{l-1} + \sum_k \mathbf{p}_l(k)\mathbf{s}_k^{l-1}, \quad l \in [1, 2, \cdots, L], \tag{4.23}$$

$$\mathbf{s}_i^l = \mathbf{s}_i^{l-1} + \sum_k \mathbf{u}_l(k)\mathbf{d}_k^l, \quad l \in [1, 2, \cdots, L], \tag{4.24}$$

where $\mathbf{p}_l(k)$ and $\mathbf{u}_l(k)$ are the prediction and update weights at lth iteration, respectively. For the $5/3$ filter bank, $L = 1$, $\mathbf{p}_1 = -\frac{1}{2}$, $\mathbf{u}_1 = \frac{1}{4}$, while for the Daubechies $9/7$ filter bank, $L = 2$, and

$$\begin{aligned} \mathbf{p}_1 &= -1.586134342, \quad \mathbf{u}_1 = -0.052980118, \\ \mathbf{p}_2 &= +0.882911075, \quad \mathbf{u}_2 = +0.443506852. \end{aligned} \tag{4.25}$$

Upon completing all the lifting iterations, the output of the final prediction step is scaled by a factor to form the high-pass subband, and that of update step is scaled by another factor to form the low-pass subband. The inverse of DWT resulting from lifting scheme can be obtained by undoing all the prediction and update steps in the reverse order.

To realize lossless compression based on DWT, integer-to-integer transform is desired. An important feature of lifting scheme is that it allows modifying lifting filters in any desired manner. A forward integer-to-integer transform using lifting can be implemented by inserting quantizers immediately after the calculations of every prediction and update terms. Such quantizers typically perform the operation of truncation or rounding to the nearest integer. The JPEG 2000 Part 1 applies the integer-to-integer $5/3$ filter bank [Adams and Kossentni 2000] for the lossless compression. The modified forward transform is given by

$$\begin{aligned} \mathbf{y}(2k+1) &= \mathbf{x}(2k+1) - \left\lfloor \frac{\mathbf{x}(2k) + \mathbf{x}(2k+2)}{2} \right\rfloor, \\ \mathbf{y}(2k) &= \mathbf{x}(2k) + \left\lfloor \frac{\mathbf{y}(2k-1) + \mathbf{y}(2k+1) + 2}{4} \right\rfloor, \end{aligned} \tag{4.26}$$

where $\mathbf{y}$ is the output sequence, and $\mathbf{x}$ is the input sequence.

4.6.3 Quantization

The JPEG 2000 Part 1 employs a midtread uniform quantizer, where the deadzone, i.e., the central quantization interval, is twice as wide as the other intervals. Given the step size $\triangle_b$ for a subband b, the quantizer maps a wavelet coefficient $\mathbf{y}_b(u, v)$ at location (u, v) to a quantization index $\mathbf{q}_b(u, v)$ as

$$\mathbf{q}_b(u, v) = \operatorname{sign}(\mathbf{y}_b(u, v)) \left\lfloor \frac{|\mathbf{y}_b(u, v)|}{\triangle_b} \right\rfloor. \tag{4.27}$$

The step size $\triangle_b$ for each subband is specified in terms of a 5-bit exponent, ϵ_b, and an 11-bit mantissa, μ_b, as

$$\triangle_b = 2^{-\epsilon_b}\left(1 + \frac{\mu_b}{2^{11}}\right). \tag{4.28}$$

At the decoder, the dequantized value $\tilde{\mathbf{y}}_b(u,v)$ can be obtained by

$$\tilde{\mathbf{y}}_b(u,v) = \begin{cases} 0, & \text{if } \mathbf{q}_b(u,v) = 0, \\ (\mathbf{q}_b(u,v) + \delta_b)\triangle_b, & \text{if } \mathbf{q}_b(u,v) \neq 0, \end{cases} \tag{4.29}$$

where $0 \leq \delta_b < 1$ is a reconstruction parameter, and is often set to $\frac{1}{2}$.

4.6.4 Entropy Coding

The entropy coding in JPEG 2000 is based on the embedded block coding with optimised truncation (EBCOT) algorithm proposed by Taubman [2000]. After quantization, each subband is partitioned into small rectangular blocks, called *codeblocks*, and each codeblock is encoded independently. The coefficients in a codeblock are bitplane encoded to create embedded bitstream. Further, each codeblock is divided into stripes, where a stripe consists of four rows, except the last stripe with possibly less than four. For each bitplane in a stripe, the coding is carried out in three passes with the option of truncating the bit stream at the end of each pass. However, only the cleanup pass is performed for the first nonzero bitplane. This multipass coding approach enables optimized embedding of the bitstream where the information reduces the distortion most is encoded first. The three coding passes include significance propagation pass, magnitude refinement pass, and cleanup pass. During the significance propagation pass, only the bits of the coefficients with known significant immediate neighbours are encoded. The reason is that the neighbours of coefficients that are found significant in previous bitplane are most likely to become significant in the current one. In the second pass, the bits corresponding to the coefficients that have been found significant in the previous bitplane are encoded. All bits not encoded during the first two passes are encoded in the cleanup pass. As such, each coefficient bit in a bitplane is encoded only in one of the three passes.

The coder used in the JPEG 2000 is known as the MQ-coder [Taubman 2000] which is an implementation of context-based adaptive arithmetic coding. The probability of a binary symbol is estimated from a *context* formed from its current significance state, as well as the significance states of its immediate eight neighbours corresponding to the current and previous bitplanes. The MQ-coder can choose from 18 different adaptive probability context models. Nine of these contexts are used significance coding during the significance propagation and cleanup passes, five are used for sign coding, three are used for magnitude refinement pass, and one is used for run-length coding typically used in the cleanup pass. The relevant model adaptively estimates the probability distribution of the bit to be coded, and the arithmetic coder assigns a representation of suitable length accordingly. Each codeblock employs an MQ-coder to generate a single codeword for the entire codeblock.

The arithmetic encoding process creates a sequence of nested intervals in the form $\Phi(S) = [\alpha_k, \beta_k), k = 0, 1, \cdots, N$, where S is the source sequence, $0 \leq \alpha_k \leq \alpha_{k+1}$, and $\beta_{k+1} \leq \beta_k \leq 1$ [Said 2003]. Using an alternative representation of the interval $[\alpha, \beta)$ as $|b, l\rangle$, where $b = \alpha$ and $l = \beta - \alpha$, the intervals result from the arithmetic coding are defined recursively by [Jelinek

1968]

$$\begin{aligned}\Phi_0(S) =&|b_0, l_0\rangle = |0, 1\rangle,\\ \Phi_k(S) =&|b_k, l_k\rangle = |b_{k-1} + c(s_k)l_{k-1}, p(s_k)l_{k-1}\rangle, \quad k = 1, 2, \cdots, N.\end{aligned} \tag{4.30}$$

In (4.30), $p(s_k)$ is the probability, and $c(s_k)$ is the cumulative probability of symbol s_k. The codeword developed by this recursive interval partitioning maps the entire source sequence to a subinterval of $[0, 1)$. The final codeword is usually determined as the shortest decimal number in the subinterval, and represented by the shortest binary fraction. The decoded sequence is determined by the code value $\hat{v}$ of the compressed sequence.

4.6.5 Bit Stream Organization

The arithmetic coding of bitplanes in the JPEG 2000 is referred to as *Tier-1* coding. The compressed data results from this coding stage is subject to the *Tier-2* coding stage, where they are organized into units known as *packets* and *layers* [Rabbani and Joshi 2002; Taubman and Marcellin 2002]. Each DWT resolution level of a tile is partitioned into rectangular regions called *precincts*. The precinct partition at a given resolution induces a partitioning of the subbands at the same resolution level. We know that each subband is also partitioned into codeblocks. A precinct may consist of multiple codeblocks, and its boundary coincides with a codeblock boundary. The purpose of precinct is to organize codeblocks based on their spatial and resolution groupings. It makes it easier to access the wavelet coefficients for a particular spatial region of an image.

In order to manage a large number of codeblocks in the compressed bit stream, the JPEG 2000 uses *layers* to represent image quality incrementally. All of the codeblocks from all subbands and components of a tile contribute compressed data to each layer. Each codeblock contributes a number of coding passes (or none) to a specific layer. Thus, the coded data of each codeblock is distributed over one or more layers in the code stream.

The compressed data belonging to a specific tile, component, resolution, layer, and precinct is organized into a *packet*. A group of packets, one from each precinct of each resolution level, makes up a layer. The final bit stream of the JPEG 2000 is a succession of layers. Each layer successively and monotonically increases the image quality.

4.7 CCSDS Recommendation for Data Compression

Data compression problem is an important issue for the remote sensing systems that acquire data using on-board cameras. Along the development of technologies for acquiring data at tens, hundreds, or thousands of wavelengths, it is essential to employ image compression to match the available on-board storage, and downlink transmission bandwidth. While the JPEG 2000 or its predecessor, the JPEG, are the standard techniques for image compression, they are more suitable for ground systems. On the other hand, the CCSDS has been working toward developing standards for space data handling [CCSDS 2017a] since 1982. Among these, two most recent standards for space data compression, namely the new recommended standard CCSDS-122 [CCSDS 2017b] for monoband image, and CCSDS-123 [CCSDS 2012] for lossless compression of multispectral and hyperspectral images.

4.7.1 CCSDS-122

In the new image data compression (IDC) recommended standard, CCSDS-122, a number of differences compared to the JPEG 2000 are pointed out. In particular, the CCSDS-122 recommended standard specifically targets use aboard spacecraft, and focuses on lower-complexity in both software and hardware implementations. The technique described in the CCSDS-122 can be used to achieve both lossless and lossy compression of 2D (or monoband) spatial image data. The compressor consists of two functional components, the DWT and the bit plane encoder (BPE). As the recommended decorrelation transform, the DWT makes use of the 9/7 floating point filters for lossy compression, and the 9/7 integer filters for the lossless compression. Following the DWT, the wavelet coefficients are either rounded to the nearest integers for lossy compression, or scaled using weight factors in the case of lossless compression.

The BPE processes the wavelet coefficients in *blocks* of 8×8 coefficients. Each block comprises of coefficients from the lowest resolution image, and all the subbands at different resolutions. A block roughly corresponds to a localized region in the original image.The blocks are then grouped into *segments*, each segment can be processed independently. Further, 16 consecutive blocks within a segment are gathered in a *gaggle* for entropy coding.

4.7.2 CCSDS-123

The standard is intended for onboard lossless compression of multispectral and hyperspectral images. The encoder in the CCSDS-123 is also structured in two functional parts, namely the predictor and encoder. The predictor uses an adaptive linear prediction method to provide an estimate of the current pixel based on the previously scanned pixels. Given an original pixel $x_{i,j,k}$, at spatial location (i, j) and component k, and its predicted value $\tilde{x}_{i,j,k}$, the error $e_{i,j,k} = x_{i,j,k} - \tilde{x}_{i,j,k}$ is computed, and then mapped to a non-negative integer, called *mapped prediction residual*. The predicted pixel value $\tilde{x}_{i,j,k}$ is estimated based on the nearby pixels in the current spectral band and P preceding bands. Two prediction modes, namely *full* and *reduced* modes, are possible depending on whether both spectral and spatial neighbours are considered, or only spectral ones are considered. In particular, the neighbouring pixels are combined to produce a local sum $\sigma_{i,j,k}$. To improve the accuracy of the prediction, the difference between the local sum and its corresponding scaled original pixel is tracked and stored in a *local difference vector*, $U_{i,j,k}$ for some previous pixels. The local difference vector is further scaled through an inner product with an adaptive weight vector to produce a weighted local difference. Finally, the local sum $\sigma_{i,j,k}$ and the weighted local difference are employed to obtain a scaled predicted pixel value $\tilde{x}_{i,j,k}$.

The second part, the encoder, encodes the mapped prediction residuals on a sample or block adaptive approach. Under the sample adaptive approach, each mapped prediction residual is coded using variable length codeword. Using the block adaptive approach, the mapped prediction residuals are grouped into blocks, which are then encoded independently.

Further details on the implementations of these two recommended standards and their performance evaluations can be found in [García-Vilchez and Serra-Sagristà 2009; Blanes et al. 2014].

4.8 Summary

In this chapter we discussed some essential components, common techniques in hyperspectral data compression. We began with the description of an entropy coding technique. The Huffman coding produces variable length, optimum prefix codes for a given source model. Its coding and decoding processes are discussed in detail. Two quantization approaches, namely scalar quantization and vector quantization, are discussed, where more focus is put on uniform scalar quantizer. Two popular predictive coding methods, the DPCM and CALIC, and their extensions to hyperspectral image compression are described. Transform coding plays a key role in data compression. The common transforms, such as DCT, KLT, PCA, subband coding, and DWT lead to state-of-the-art compression methods that are applicable to hyperspectral image compression. We also investigated the new image compression standard JPEG 2000, and two CCSDS image compression standards for remote sensing images with single or multiple bands.

Chapter 5

Sparse Representation Based Lossy Hyperspectral Image Compression

5.1 Introduction

Hyperspectral imaging, as a part of mainstream remote sensing, has been an area of active research, development and application recently. Modern hyperspectral sensors have spectral resolutions covering up to several hundred bands. The transmission and storage of the resultant large volume of data collected by such sensors require data compression to be an essential feature of systems incorporating hyperspectral sensors. In hyperspectral data compression, finding an appropriate model for the data is the heart of the problem. Sparsity based models have been widely adopted as the choice of models in this field. These models assume that the signal or image vectors usually lie in a combination of subspaces, where each subspace is spanned by a few elements from a set of basis vectors. Using sparsity based models, many types of signals can be sparsely represented through linear transforms. A common goal in such a transform is to find a set of basis vectors that span the vector space of input signals so that some criterion of optimality, such as decorrelation and sparseness, can be met. Indeed, this type of approach leads to the state-of-the-art image compression standards such as JPEG and its successor JPEG 2000.

A common transform coding strategy is applying a linear transform to a signal or image vector $\mathbf{x} \in \mathbb{R}^M$ as $\mathbf{b} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$, where the matrix $\mathbf{W} \in \mathbb{R}^{M\times K}$ consists of the basis vectors, and $\mathbf{b} \in \mathbb{R}^K$ is the coefficient vector. This type of transform is referred to as decomposition or analysis. On the other hand, the linear transform in sparse representation [Olshausen and Field 1997] takes the superposition or synthesis approach. It is based on learning a dictionary using a set of training data. The dictionary is learned in a way such that it yields sparse structures in the transform domain. Simply put, a signal or image vector $\mathbf{x}$ is modelled as $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$, where the columns in $\mathbf{D} \in \mathbb{R}^{M\times K}$ correspond to a set of learned dictionary atoms, and $\boldsymbol{\alpha} \in \mathbb{R}^K$ is the coefficient vector.

Using adaptive dictionaries and sparsity as an á priori, sparse representation has found many applications in vision tasks such as image classification [Bahrampour et al. 2016, and references therein]; image denoising, deblurring, inpainting [Elad et al. 2010]; image restoration [Mairal et al. 2008a]. Compression of a specific family of images, or more general images using sparse representation can be found in [Bryt and Elad 2008; Skretting and Engan 2011; Shao et al. 2014]. In [Huo et al. 2012; Wu et al. 2014; Ülkü and Töreyin 2014 2015], various lossy hyperspectral data compression methods using sparse representation are presented.

In this chapter, we present a study on developing a lossy hyperspectral image compression framework based on sparse representation. A preliminary work on this topic appears in [Wang and Celik 2016], and part of the work in this chapter is published in [Wang and Celik 2017]. The dictionary in the sparse representation is learned either in the original pixel domain or in the multi-scale pixel domain. The former exploits the spectral redundancy in a hyperspectral image, and the latter both the spatial and spectral redundancies. We employ online dictionary learning algorithm for dictionary learning, and orthogonal matching pursuit algorithm for sparse coding. The final bit stream is formulated by applying quantization and entropy coding of the nonzero components in the sparse coefficients. The performance of the proposed method is evaluated and compared to two wavelet transform based methods, namely, JPEG 2000 and 3D-SPIHT.

We intend to make the following contributions in this chapter. A hyperspectral data compression framework incorporating sparse representation is developed, implemented, and also evaluated against two of the state-of-the-art existing methods. The experimental results show that the proposed methods are competitive in terms of rate-distortion performance to the other approaches considered in this thesis. This indicates the effectiveness of sparse representation in exploiting the spectral correlations in hyperspectral images for data compression.

The rest of this chapter is organized as follows. Section 5.2 presents a brief overview of the dictionary learning problem, and some related work in data compression based on sparse representation. The proposed lossy compression framework for hyperspectral image is given in Section 5.3. The proposed compression approach is evaluated experimentally in Section 5.4, and finally, Section 5.5 includes some concluding remarks.

5.2 Background and Related Work

5.2.1 Dictionary Learning

Following the work by [Mairal et al. 2009], the dictionary learning problem is formulated as

$$\underset{\mathbf{D},\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 + \lambda\mathcal{S}\left(\boldsymbol{\alpha}_i\right)\right), \tag{5.1}$$

where $\mathbf{x}_i \in \mathbb{R}^M, \boldsymbol{\alpha}_i \in \mathbb{R}^K$ for $1 \leq i \leq N$, and $\mathbf{D} \in \mathbb{R}^{M\times K}$ $(M < K)$ is an overcomplete dictionary. $\mathcal{S}()$ is a sparsity inducing regularizer such as ℓ_1-norm or ℓ_0-pseudo norm, and finally, λ is a regularization parameter. The aim of dictionary learning in (5.1) is to reduce the error in representing each sample $\mathbf{x}$ while inducing sparsity in its coefficient $\boldsymbol{\alpha}$.

The optimization problem in (5.1) is not convex considering both $\mathbf{D}$ and $\boldsymbol{\alpha}$ varying. However, it is convex with respect to either one of the two variables while the other one is fixed. This results in an iterative two-step strategy, namely *sparse coding* and *dictionary learning*, is commonly applied to solve (5.1) [Mairal et al. 2009; Aharon et al. 2006].

Sparse coding: This step refers to finding the coefficient vectors when a dictionary $\mathbf{D}$ is given. Sparse coding is solved in a separable manner where for each sample $\mathbf{x}$, its coefficient vector $\boldsymbol{\alpha}$ can be found independently of the other samples. A greedy algorithm such as matching pursuit [Mallat and Zhang 1993] or orthogonal matching pursuit [Davis et al. 1997] can be applied when ℓ_0-norm is used to induce sparsity in the coefficient vectors. Efficient algorithms, such as basis pursuit [Chen et al. 2001] and LARS-Lasso algorithm [Efron et al. 2004] have also been proposed for ℓ_1 normalization setting in (5.1).

Dictionary learning: In dictionary learning (or update) step, the coefficient vectors are known and the optimization problem (5.1) becomes quadratic in terms of $\mathbf{D}$. An analytic solution of the quadratic problem, known as method of optimal direction (MOD), was introduced by [Engan et al. 1999]. Although efficient, MOD may suffer from relatively high computational complexity which involves matrix inversion. Rather than using matrix inversion, the K-SVD algorithm developed by [Aharon et al. 2006] performs the dictionary update atom by atom in a simple and efficient process. Both MOD and K-SVD algorithms access the whole training data at each iteration. Hence, these methods may become impractical in terms of speed and memory requirements when the training data is very large. In such a setting, online dictionary learning [Mairal et al. 2009] becomes advantageous. Online dictionary learning algorithm is based on stochastic gradient algorithms [Bottou and Bousquet 2008], which allow training dictionaries from large data sets, and is able to achieve fast convergence with low expected cost and good trained result. This leads us to use online dictionary learning in modelling hyperspectral data using sparse representation, where the number of training samples used for dictionary learning can easily exceed millions.

5.2.2 Related Work

Sparse representation has found some applications in image compression problem, both in specific and general image compressions. Bryt and Elad [2008] proposed a method for the compression of facial images using the K-SVD dictionary learning [Aharon et al. 2006]. The dictionary is employed in the sparse approximation of image patches following the geometric alignment of facial images. This method shows that sparse representation based compression scheme could outperform the state-of-the-art wavelet based compression methods, such as JPEG 2000, for some specific types of images. Shao et al. [2014] demonstrates the compression of fingerprint images by constructing overcomplete dictionaries using a number of different algorithms including the K-SVD. The results show that the K-SVD based compression method achieves similar performance to that of JPEG 2000, or superior performance at high compression ratios.

A general image compression scheme using trained dictionaries is presented in [Skretting and Engan 2011]. This compression scheme explores training dictionaries using different algorithms, as well as in different input domains, namely, pixel and wavelet domains. On comparing the results to DCT and wavelet based schemes, the dictionary based scheme is superior to JPEG (DCT based) and slightly inferior to JPEG 2000.

Hyperspectral image consists of high degree of redundancies both in spectral and spatial domains, which are amenable for sparse modelling. In [Huo et al. 2012; Wu et al. 2014; Ülkü and Töreyin 2014], sparse representation based hyperspectral image compression frameworks are proposed. The basic idea in [Huo et al. 2012] is to learn a dictionary using a selected band, and the rest of the bands are approximated using the learned dictionary. This approach results in the need to transmit the selected band to the decoder for reproducing the dictionary. A number of issues are raised regarding this approach. These include: i) how to choose the band to be used for training a dictionary; and ii) the possibility of updating the dictionary after the sparse coding of each band. Wu et al. [2014] developed a hyperspectral data compression method based on the double sparsity model introduced in [Horev et al. 2012]. Besides transmitting the compressed bit stream, this approach also transmits information about the dictionary itself. This is done by sparse coding the dictionary using a predefined one, such as DCT. Ülkü and Töreyin [2014 2015] presented another hyperspectral data compression framework using sparse representation. Online dictionary learning algorithm is used in their framework, while the K-SVD dictionary learning algorithm is used in [Huo et al. 2012; Wu et al. 2014]. The authors in [Ülkü and Töreyin 2014 2015] compare the performance of the proposed scheme with a predictive lossy compression and

JPEG 2000, showing some competitive results. However, the bit rates under their framework are within 0.7 bits per sample.

Similar to sparse representation, a linear mixture model

$$\mathbf{X} = \mathbf{\Phi}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{5.2}$$

has been used in [Du and Chang 2004, and references therein] for hyperspectral image compression with target detection and classification as the main optimal compression criteria. In (5.2), $\mathbf{\Phi}$ is a dictionary of approximation elements, $\boldsymbol{\beta}$ consists of decomposition coefficients and $\boldsymbol{\epsilon}$ is the model error. When the dictionary in (5.2) represents the *endmembers*, that is the spectral signatures of the various material components in the scene, the resulting coefficients indicate the material abundances of their corresponding pixels. Thus, these coefficients are often used for target detection or classification. To determine the endmembers for a linear mixture model dictionary, a priori of the targets in the data is often needed. An unsupervised process is proposed in [Du and Chang 2004] to learn the target information from the data to formulate the dictionary $\mathbf{\Phi}$ without a priori information. The number of dictionary elements, or spectral signatures, is usually much smaller than the number of spectral bands in hyperspectral images. Thus, significant compression can be achieved. Moreover, using this type of compression information, the abundance coefficients can be directly used in the applications of target detection or classification without utilizing the entire hyperspectral image cube.

Compared to linear mixture model described above, the sparse representation model does not assume the data lie in the space spanned by a number of endmembers in a linear mixture dictionary. Instead, the sparse representation dictionary is trained to adapt to the data to some degree, so that it is possible for several dictionary elements to represent a single type of material. We can view this scenario as forming a subspace to represent a spectral signature instead of using a single endmember. In fact, this type of flexibility may be more advantageous considering the fact that hyperspectral image pixels are hardly pure in most scenes. Another advantage of sparse representation based method is that unlike other coefficient vectors result from a transform such as wavelet, sparse representation coefficients have not only energy compacting feature, they can be used as other types of feature in various tasks such as classification [Charles et al. 2011].

Apart from the work discussed above, a number of highly efficient compression methods has been developed for hyperspectral image and other remote sensing data. These techniques include transform based coding, vector quantization, predictive coding and others [Motta et al. 2009; Penna et al. 2007]. Since sparse representation based compression method is closely related to transform based coding approaches, we compare the proposed method with some representative transform based techniques, where we chose JPEG 2000 and 3D SPIHT. Typically, the development of a transform based compression algorithm for hyperspectral data involves separate spectral and spatial decorrelations. A commonly applied transform in this regard is the DWT. This type of approaches leads to a number of successful hyperspectral image compression techniques. These include but not limited to 3D-SPIHT [Kim et al. 2000], three-dimensional set partitioned embedded block (3D-SPECK) [Tang et al. 2003a], and three-dimensional tarp coding (3D-TARP) [Wang et al. 2004]. Further, the part 2 of JPEG 2000 standard supports multi-component transform to be applied to high dimensional images, including hyperspectral image. It has been shown that JPEG 2000 plus a spectral decorrelation, such as PCA or DWT, could achieve rate-distortion performance superior to other wavelet based methods [Du and Fowler 2007; Fowler and Rucker 2007].

Using dictionary based sparse coding, we represent a high dimensional data in a lower dimensional space of a few active dictionary atoms. Similar to PCA, this is a type of dimensionality reduction. The difference is in sparse representation, the dimensionality of dictionaries is usually higher than that of input pixels, and only a few of the dictionary atoms are activated to represent one pixel. Using PCA, however, almost all of the principle components (PCs) in the transform matrix are used to represent a pixel. (Note that the transform matrix here may be formed by all of the PCs or a subset of the PCs of the data.) In another words, PCA allows us to capture pairwise correlations while sparse representation is not restricted to pairwise.

5.3 Sparse Representation Based Hyperspectral Image Compression

In this section, we discuss the details on applying sparse representation in modelling hyperspectral data, and the overall structure of the proposed lossy compression framework. A common issue in the sparse representation based data compression methods is the choice of online or offline dictionary training. The difference lies in whether to include (online training) the dictionary in the compressed bit stream or not (offline training). We adopt a hybrid approach in our framework. In the hybrid approach, we first train a base dictionary offline, then update the base dictionary using the data to be compressed. This approach requires transmitting the updated dictionary to the decoder. To reduce the overhead of transmitting the updated dictionary, we transmit only the difference between the base and updated dictionaries. As in [Horev et al. 2012], we may also set the base dictionary as a pre-defined one, such as DCT. However, the base dictionaries in our method are pre-trained due to the following considerations. Firstly, the remote sensing data is often acquired over the same region during different time periods. Therefore, it is feasible to train a dictionary on the previously acquired data, and then use it as the base dictionary in encoding of an input data. At the decoder, a dictionary is constructed using the same base dictionary and the transmitted difference between the base and updated ones at the encoder. Secondly, the overhead of transmitting a dictionary is reduced by sending the difference between the base and updated dictionaries. Lastly, the dictionary can be conveniently updated. Compared to training a dictionary without a pre-trained base dictionary, the hybrid approach can reduce the computational cost in the encoder substantially.

A block diagram of the proposed lossy compression framework is given in Figure 5.1. The framework consists of three main processes: dictionary training, data compression, and data decompression. The following subsections discuss each process in details.

5.3.1 Dictionary Training

The main task at this stage is to train a dictionary offline using a set of training samples from a hyperspectral data set. The learned dictionary is then considered as the base dictionary for the subsequent data compression stage. Training a representative dictionary is crucial for the overall performance of the compression method. We consider using online dictionary learning algorithm [Mairal et al. 2010] to train a dictionary. It is computationally prohibitive to learn a dictionary using batch based training method, such as K-SVD, when a data set has a large quantity of samples. Compared to batch based training method, online dictionary learning has the advantage of being capable of handling large data sets.

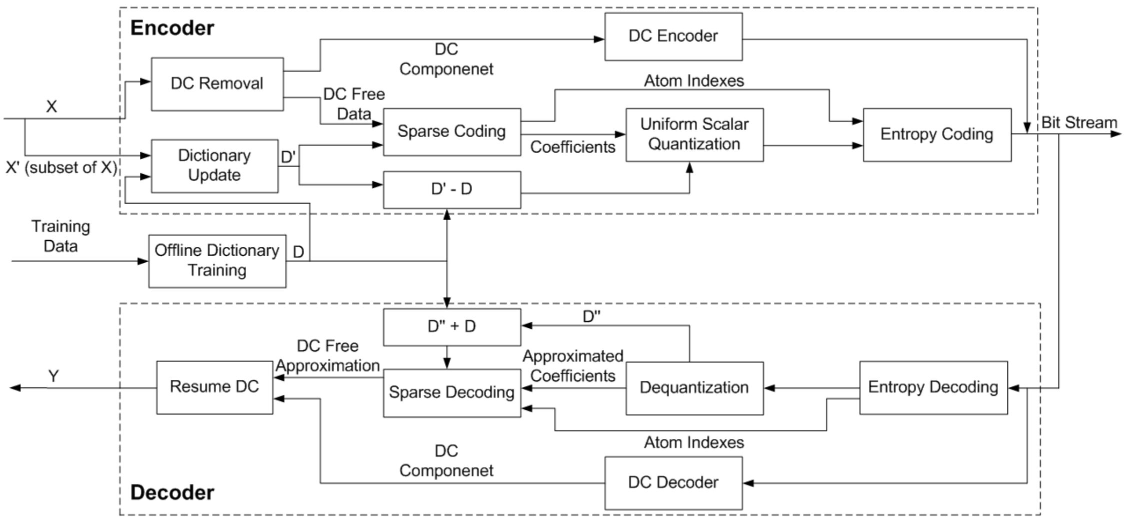


Figure 5.1: The proposed hyperspectral image lossy compression framework based on dictionary and sparse coding.

The dictionary learning problem in our approach is set using matrix form as follows.

$$\underset{\mathbf{D},\mathbf{A}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{X}-\mathbf{DA}||_{\mathrm{F}}^{2} \text{ subject to } ||\boldsymbol{\alpha}_i||_0 \leq \mu,\ 1 \leq i \leq N, \tag{5.3}$$

where each column in matrix $\mathbf{X} \in \mathbb{R}^{M\times N}$ represents one of the training samples, and the corresponding column in sparse matrix $\mathbf{A} \in \mathbb{R}^{K\times N}$ is the coefficient vector. The parameter μ indicates the maximum number of nonzero components in each coefficient vector. The optimal criterion in Problem (5.3) is the least squared error for representing each input signal, and $||\boldsymbol{\alpha}_i||_0 \leq \mu$ is the sparsity constraint on the coefficient vectors. The dictionary $\mathbf{D} \in \mathbb{R}^{M\times K}$ consists of atoms (i.e. columns) that have unit ℓ_2-norms. Furthermore, $\mathbf{D}$ is overcomplete, that is the number (i.e., K) of atoms is greater then the number (i.e., M) of dimensions of an input data. The reason for desiring an overcomplete dictionary is its flexibility in modelling an input structure, as well as greater robustness in the case of noise [Olshausen and Field 1997].

Online dictionary learning algorithm [Mairal et al. 2010] learns a dictionary through a number of iterations. At each iteration, one training sample (or a small group of samples) is drawn and its sparse decomposition is computed over the dictionary learned during the previous iteration. This is followed by a dictionary update step where a new dictionary is computed based upon the sparse decompositions computed in previous iterations and the current dictionary. In the dictionary update step, each atom in the dictionary is updated sequentially by minimizing the quadratic error in (5.3), while keeping the other atoms and the coefficient matrix $\mathbf{A}$ fixed. This learning process is a stochastic approximation approach that aims at obtaining sufficient statistics by aggregating the past information during each iteration. The statistics are then utilized to minimize an expected cost function. This approach often yields faster convergence with a sufficiently good expected cost.

In (5.3), the dimension of pixel vectors, i.e., M, decides the size of dictionary atoms in $\mathbf{D}$. Since we expect the learned dictionary to be overcomplete, i.e. $K = cM,\ c > 1$, consequently the number of atoms, K, is set according to M. On the other hand, the dimension of coefficient vectors is dependent on the value K. When it comes to encoding the sparse coefficient matrix $\mathbf{A}$ in data compression stage, the larger the value of K becomes, the more negative impact it gives in the resulting bit rate. The choice of an optimal value for K is mostly done experimentally in related works.

As the trained dictionaries are going to be applied in hyperspectral image compression, they should be learned in such a way that can exploit the correlations in the input data. There exists significant correlations between consecutive bands in a hyperspectral image. A decorrelation in the spectral domain is usually an important step in other hyperspectral image compression methods including 3D-SPIHT and JPEG 2000. To exploit the spectral redundancies in a hyperspectral image, we train a dictionary in the spectral domain and refer to it as *spectral dictionary* (SD). More concretely, an SD is trained on data set $\mathbf{X}$, where $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}, \mathbf{x}_i \in \mathbb{R}^M$, M is the number of bands in hyperspectral image, and N is the total number of training pixels. For hyperspectral images, it is common for N to be large.

On the other hand, the neighbouring pixels in a hyperspectral image usually consist of similar materials. Thus, hyperspectral images possess not only a high degree of spectral correlations, it also has rich spatial correlation among neighbouring pixels. In order to train a dictionary that exploits both spatial and spectral correlations, we employed dictionaries learned in wavelet domain. This is achieved by applying a wavelet decomposition to a hyperspectral image in the spatial domain, followed by training the dictionary using the spectral wavelet coefficients. We refer to this type of dictionary as *multiscale spatial-spectral dictionary* (MSSD). Further details on MSSD learning are given in the following subsection.

5.3.1.1 Multiscale Spatial Spectral Dictionary Learning

Multiscale analysis of images has taken a prominent role in image processing during the last several decades. Several different families of multiscale decompositions of 2D images have been proposed and developed. They include Laplacian Pyramid, Wavelet decomposition, Wavelet Packet decomposition, and advanced multiscale decompositions such as Counterlets, Curvelets, Bandlets, and more. In this work, we applied discrete wavelet decomposition, or DWT, for our purpose.

Any spectral band, $\mathbf{B}$, of a hyperspectral image can be viewed as a 2D image. We apply DWT to $\mathbf{B}$ to obtain the corresponding wavelet coefficients. In wavelet domain, these coefficients can be organized into a collection of images, also referred to as subbands, at different scales and orientations (horizontal, vertical and diagonal). Figure 4.6 on Pg. 55 shows the three-level DWT of image 'Lena'. Specifically, given the number of decomposition levels, S, we decompose each hyperspectral band image independently. This process results in $3S + 1$ subbands for each hyperspectral band, and they are denoted as $LL_0, LH_0, HL_0, HH_0, LH_1, \ldots, HL_{3S}, HH_{3S}$, respectively. We group the same subbands from all the hyperspectral bands and formulate hyperspectral images of different scales and orientations in wavelet domain. A sub-dictionary is then trained on each of these subband hyperspectral images, and subsequently, $3S + 1$ sub-dictionaries are trained.

Through multiscale dictionary learning, we aim to achieve decorrelations in both spatial and spectral domains. The spatial decorrelation is achieved by wavelet decomposition of each hyperspectral band, and the spectral decorrelation is obtained via sparse coding the spectral wavelet coefficients using the MSSDs. This approach offers a number of flexible options due to the multiscale nature of DWT. These include the number of decomposition levels, the choices of wavelet filters, and the ways to group subbands, such as according to the orientation or decomposition level etc., to train dictionaries. In this thesis, we investigated the following options.

1. Limit the number of decomposition levels, in order to reduce the number of dictionaries to be trained.
2. Group each subband hyperspectral image to train a sub-dictionary. This allows the allocation of more bits to the more significant subbands.
3. Two choices of wavelet filters, namely 5/3 filter bank and 9/7 filter bank, are applied for reversible and irreversible DWT, respectively. These two choices are commonly applied as standard filter banks in other DWT based hyperspectral image compression schemes. Some details of the above filters were also discussed in Sections 4.5.4 and 4.6.2. Tables 5.1 and 5.2 show the corresponding coefficients.

Table 5.1: Coefficients for 5/3 Filter Bank

Low-pass filter		High-pass filter	
n	$g[n]$	n	$h[n]$
0	$\frac{3}{4}$	0	1
±1	$\frac{1}{4}$	±1	$-\frac{1}{2}$
±2	$-\frac{1}{8}$		

Table 5.2: Coefficients for 9/7 Filter Bank

Low-pass filter		High-pass filter	
n	$g[n]$	n	$h[n]$
0	0.602949018236	0	0.557543526229
±1	0.266864118443	±1	-0.29635881557
±2	-0.078223266529	±2	-0.028771763114
±3	-0.016864118443	±3	0.045635881557
±4	0.026748757411		

5.3.1.2 Dictionary Atom Sorting

The dictionaries applied in our compression framework are overcomplete and unstructured. This is in contrast to the bases in Fourier or wavelet transforms, where they are well ordered in terms of frequency, and hence, have intuitive physical interpretations. To enforce similar interpretations for the dictionary atoms, Li et al. [2013] proposed a solution by generalizing the concept of frequency for overcomplete dictionaries. The proposed solution is based on the observation that the magnitudes of coefficients at a certain frequency f in transforms such as Fourier or wavelet are proportional to $1/f$ [Field 1987]. That is, small magnitudes are more common then the larger ones. Thus, Li et al. [2013] sorts the atoms in an overcomplete dictionary in a way such that the magnitudes of the sparse coefficients are in descending order. In this manner, the indexes of atoms can be interpreted as frequencies. Specifically, the above sorting algorithm proceeds in the following steps:

Step 1 Determine the order of the active atoms for each training sample by sorting them according to the magnitudes of the sparse coefficients in descending order;
Step 2 Record all the orders of the activated atoms and obtain an average frequency order;
Step 3 Sort the atoms in the dictionary in an ascending order based on their frequency orders.

The main purpose of dictionary atom sorting discussed above is to impose a similar understanding of the atoms in terms of frequency to that of Fourier or wavelet bases. In DCT or DWT based data compression, the data can be well approximated using the low frequency coefficients, and the high frequency ones can be discarded. However, in our sparse representation based approach, the number of coefficients during sparse coding can be controlled using parameters, such as the maximum number of nonzero coefficients. Thus, using a sorted dictionary has not shown a significant impact on the compression performance of hyperspectral data.

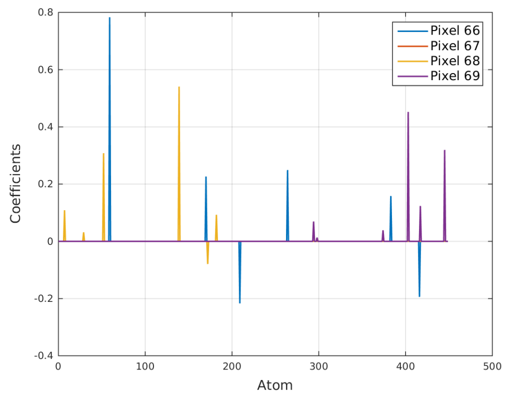
5.3.2 Hyperspectral Image Compression

Given a learned dictionary $\mathbf{D}$, and a hyperspectral image $\mathbf{X}$, the proposed method also performs the following tasks.

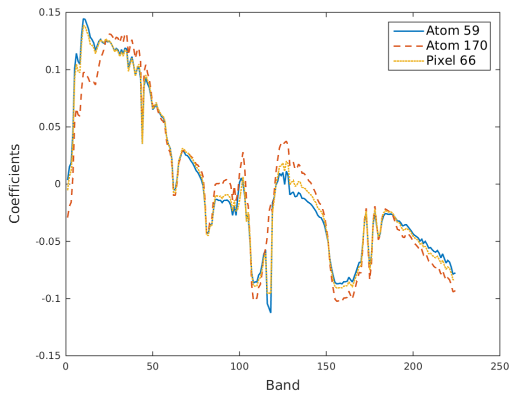
Sparse coding: we find the sparse representation matrix $\mathbf{A}$ for $\mathbf{X}$ using OMP. This problem is set as in (5.3) with the dictionary fixed. OMP, as shown in Algorithm 2 on Pg. 26, is a greedy algorithm based on *matching pursuit* [Mallat and Zhang 1993] that refines the signal approximation with an iterative procedure. At each iteration, the current residual is decomposed by projecting the signal on each dictionary atom in $\mathbf{D}$, and selecting the atom that minimizes the residual. Once an atom is selected at each iteration, the signal is orthogonally projected to the span of the current selected atoms, and the residual is recomputed for the next iteration. The iteration stops when the number of nonzero coefficients satisfies the prescribed threshold, i.e., the value μ. Note that the residual monotonically decreases at each iteration. The computational complexity of the sparse coding using OMP can be estimated based on the number of input pixel vectors, the number of dictionary atoms and the number of nonzero coefficients, which is $O(NK\mu)$. The major computational cost at each iteration comes from finding the coefficients corresponding to the current selected atoms, which involves inversion of matrix comprised of the selected dictionary atoms.

As an example, Figure 5.2 (a) illustrates the sparse coefficients of a few hyperspectral image pixels. The dictionary is learned from the hyperspectral image Lunar Lake, and its size is 224×448. The sparse coding is done using $\mu = 6$, i.e., the maximum number of nonzero coefficients in each sparse representation is 6. We can see that for `pixel 66` (the number indicates the column number in the first band of the subcube being used), the atoms being activated during the sparse coding are with indexes 59, 170, 209, 264, 383, 416. We then compared in Figure 5.2 (b) the first two activated atoms with the original pixel. In this case, the first atom (with index 59) activated is more similar to the original pixel than the second atom (with index 170) being activated. We further compared the original pixels with the reconstructed ones in Figure 5.2 (c) – (f), where we used two different μ values of 1 and 6. From these comparison, we can see that when the value μ increases, the reconstructed pixels fit more closely to the original ones.

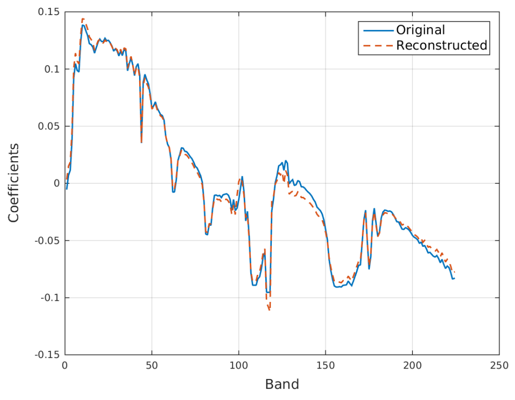
Quantization and entropy coding: Due to the nature of sparse coding, matrix $\mathbf{A}$ is sparse. In order to transmit matrix $\mathbf{A}$ to the decoder, we first employ a common sparse matrix representation scheme *index-value pair* to represent matrix $\mathbf{A}$. That is, for each nonzero component in $\mathbf{A}$, we record its position information (row and column indexes), and the nonzero value itself by traversing the matrix in a certain order such as row-major or column-major. We need to compress



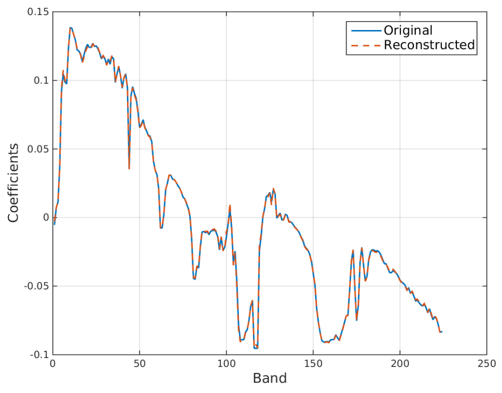
(a)



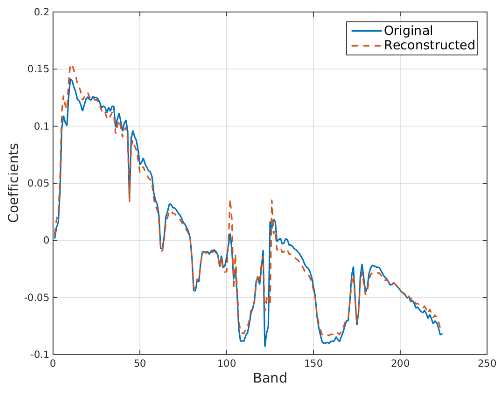
(b)



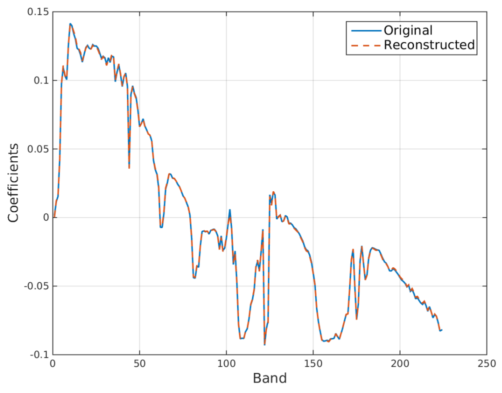
(c) $\mu = 1$



(d) $\mu = 6$



(e) $\mu = 1$



(f) $\mu = 6$

Figure 5.2: (a) Sparse coefficients of 4 hyperspectral image pixels after sparse coding using $\mu = 6$, (b) Comparison of two dictionary atoms and a hyperspectral image pixel, (c), (d), (e), (f) Comparisons of two original hyperspectral image pixels and their reconstructed ones using sparse coding with $\mu = 1$ and 6, respectively.

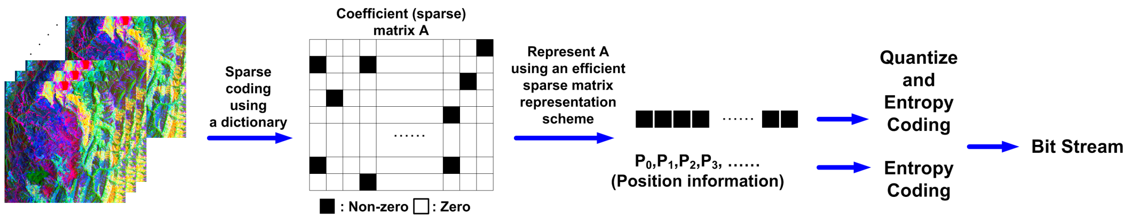

Sparse coding using a dictionary
Coefficient (sparse) matrix A
■ : Non-zero □: Zero
Represent A using an efficient sparse matrix representation scheme
P0,P1,P2,P3, ...
(Position information)
Quantize and Entropy Coding
Entropy Coding
Bit Stream

Figure 5.3: The process of hyperspectral image compression using dictionary and sparse coding

Algorithm 8: Encoding the sparse representation matrix $\mathbf{A}$

Input: Sparse representation matrix $\mathbf{A}$

Output: Bit streams for dense coefficient matrix $\mathbf{V} \in \mathbb{R}^{\mu \times N}$, and the corresponding row index matrix $\mathbf{I} \in \mathbb{N}^{\mu \times N}$

```
1 begin
2     $\mathbf{V} \leftarrow \mathbf{0}, \mathbf{I} \leftarrow \mathbf{0}$
3     $\mathbf{A} \leftarrow$ Convert $\mathbf{A}$ to an 1D array
4     Store the nonzero coefficients and their array indexes of $\mathbf{A}$ in $\mathbf{V}$ and $\mathbf{I}$, respectively
5     $\mathbf{V} \leftarrow$ Scalar quantize $\mathbf{V}$
      /* Difference coding of I                                   */
      /* $N$ is the number of elements in I                        */
6     for $i = 2$ to $N$ do
7         $\mathbf{I}[i] = \mathbf{I}[i] - \mathbf{I}[i-1]$
8     Entropy code $\mathbf{V}$
9     Entropy code $\mathbf{I}$
```

the position information without any loss to recover the matrix $\mathbf{A}$ at decoder. Figure 5.3 gives an illustration of this process. If we encode the position information without any further processing, the overhead of this part in the compressed bit stream could be significant. However, the constraint we applied in our sparse coding (see problem formulation in (5.3)) is $||\boldsymbol{\alpha}_i||_0 \leq \mu$. This means the number of nonzero components in each column of $\mathbf{A}$ is less than or equal to μ. Moreover, each column of $\mathbf{A}$ must have a few nonzero components, except the case where the input sample is a zero vector. Thus, the sparse matrix $\mathbf{A}$ is simply turned into a vector with the nonzero coefficients and their corresponding one-dimensional position indexes. The position indexes were encoded first using difference coding followed by entropy coding. For entropy coding, Huffman coding (details are given in Section 4.2) was employed in the implementation. Algorithm 8 depicts the process of encoding of matrix $\mathbf{A}$.

A uniform midrise scalar quantizer was used in our experiments. This is out of the consideration of not to quantize small coefficient values to zero. As we increase the number of nonzero entries in the sparse coefficients, some of the entries become smaller. If a uniform midtread quantizer is used, some of these smaller coefficient entries are quantized to zero, and subsequently, it increases the reconstruction error. The step size Δ in our uniform midrise quantizer was chosen

Algorithm 9: A uniform scalar quantizer for forward and inverse quantization

```
Input: V, $\Delta$, bin_size, threshold $t$, indicator $q$ (if $q$ is TRUE, it is a forward quantization;
       otherwise, it is a inverse quantization.)
Output: V
1  begin
2      for each element $v$ of V do
3          if $q$ is TRUE then
4              $y = \lfloor \frac{|v|-t}{\Delta} \rfloor + 1$
5              if $y > bin\_size$ then
6                  $y = bin\_size$
7              $v = \mathrm{sign}(v) * y$
8          else
9              $y = |v| * \Delta + \frac{t}{2}$
10             $v = \mathrm{sign}(v) * y$
```

according to the dynamic range of the nonzero components of sparse coefficients. That is,

$$\Delta = \frac{\mathrm{sign}(\max\{\mathbf{V}\}) * 2^{\lceil \log_2 |\max\{\mathbf{V}\}| \rceil} - \mathrm{sign}(\min\{\mathbf{V}\}) * 2^{\lceil \log_2 |\min\{\mathbf{V}\}| \rceil}}{\mathrm{bin\_size}}, \tag{5.4}$$

where

$$\mathrm{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0, \end{cases}$$

and bin_size is the number of distinct values in the quantizer. The forward and inverse quantizers were implemented as in Algorithm 9.

Coding the DC component: The mean value, referred to as DC component hereafter, of each data sample is removed before sparse coding. Subsequently, an overhead of lossy coding of the DC component occurs. In this thesis, the DC component is coded using predictive coding. Prediction can be viewed as a context modelling technique of very low model cost that is highly effective under the assumption of smoothness [Wu and Memon 1997]. Following the work in [Wu and Memon 1997], a simplified gradient-adjusted predictor plus context modelling of prediction error is applied in the coding of the DC component. We describe the method in the following.

For example, given the following context, predict the pixel value of p.

	x_7	x_8	x_9	x_{10}	x_{11}	
	x_6	x_3	x_2	x_4		
	x_5	x_1	$p =?$			

In order to predict the value of p, we first estimate the gradients in horizontal, vertical, and three diagonal directions as in (5.5). We may apply a weight vector in (5.6) as the scaling factors on different directional gradients. The pixel value is predicted by the neighbour which gives the minimum gradient. The method is summarized in more details as follows. (Note that, a, b in (5.5), (5.7) are values set according to the priority of different orientations, and $0 \leq c < 1$.)

Step 1 Estimate the gradients of the intensity function at the pixel in question as (5.5).

$$\mathbf{d} = \begin{pmatrix} |x_1 - x_5| & |x_2 - x_3| & |x_3 - x_6| & |x_4 - x_2| \\ |x_1 - x_3| & |x_2 - x_9| & |x_3 - x_8| & |x_4 - x_{10}| \\ |x_1 - x_6| & |x_2 - x_8| & |x_3 - x_7| & |x_4 - x_9| \\ |x_1 - x_2| & |x_2 - x_{10}| & |x_3 - x_9| & |x_4 - x_{11}| \\ |x_1 - \frac{ax_5+bx_2}{5}| & |x_2 - \frac{ax_3+bx_{10}}{5}| & |x_3 - \frac{ax_6+bx_9}{5}| & |x_4 - \frac{ax_2+bx_{11}}{5}| \end{pmatrix} \tag{5.5}$$

In (5.5), the first row estimates the gradients of the intensity function at horizontal direction, the second row at vertical direction, the third and fourth rows at two diagonal directions, and the last row at a weak diagonal direction.

Step 2 Find the row number r ($1 \leq r \leq 5$ in (5.5)) with the minimum value in the multiplication of $\mathbf{dw}$; $\mathbf{w}$, as in (5.6), is a weight vector;

$$\mathbf{w} = (w_1\, w_2\, w_3\, w_4)^T \tag{5.6}$$

Step 3 Predict the pixel value p in question by

$$\hat{p} = \begin{cases} x_1 & \text{if} \quad r = 1 \\ x_2 & \text{if} \quad r = 2 \\ x_3 & \text{if} \quad r = 3 \\ x_4 & \text{if} \quad r = 4 \\ \lfloor \frac{ax_1+bx_2}{5} + c \rfloor & \text{if} \quad r = 5 \end{cases} \tag{5.7}$$

Step 4 Compute the prediction error as the mean value of row r, where r is determined in Step 3;

Step 5 Context modelling of prediction error — Gradients characterize the relationships between the predicted pixel and its surrounding to some extent only. Context modelling of prediction errors can exploit higher order structures such as textures and local activity in an image for further compression gains [Wu and Memon 1997]. Wu [1997] has proposed an effective way of context modelling by estimating only the conditional expectations using the corresponding sample means within a context. Due to its effectiveness and low model cost, we adopted this approach in context modelling of the prediction errors as in Step 4. The prediction error sequence is then divided into a number of smaller sequences based on the context models obtained. In this way, the standard deviation within each smaller group is reduced, and compression gain can be achieved during the entropy coding stage.

We have, so far, discussed only the interior pixels. The boundary pixels need to be treated specifically, and many different options can be applied. For instance, A pixel, excluding the first two ones, at the first row can be predicted using its immediate two previous pixels; the pixels at second row can be predicted using the previous neighbours from the first and second rows etc.

The inverse of predictive coding is simply the inverse operations of the various prediction rules applied in the forward operation. Especially, for the interior pixels, the forward and the inverse operations are identical.

5.3.3 Decompression

Decompression stage consists of the following operations.

1. The inverse operation of entropy coding. It is the Huffman decoding in our implementation.
2. The inverse quantization. In our experiment, we adopted the approach presented in Algorithm 9.
3. Construct the approximated sparse representation matrix $\hat{\mathbf{A}}$ according to the recovered sparse coefficients and their corresponding array indexes.
4. Approximate the dictionary $\hat{\mathbf{D}}$ as $\hat{\mathbf{D}} = \mathbf{D}+\mathbf{D}''$, where $\mathbf{D}$ is the base dictionary (offline) and $\mathbf{D}''$ is the approximation to the difference between original updated and base dictionaries.
5. Reconstruct the hyperspectral image as

$$\mathbf{X}' = \hat{\mathbf{D}}\hat{\mathbf{A}}, \tag{5.8}$$

6. Resume the DC component to $\mathbf{X}'$ in order to obtain the final reconstructed hyperspectral image $\hat{\mathbf{X}}$.

5.4 Experiments

To evaluate the proposed method, a compression framework is implemented. For dictionary learning and sparse coding, we used SPAMS (v2.5) [Mairal et al. 2009], an open source optimization toolbox for sparse estimation. The compressed bit stream mainly consists of three parts. These include the DC component, the quantized nonzero components of the coefficient vectors and their position information (only the row indexes) in $\mathbf{A}$. The DC component is compressed using JPEG 2000 lossless coding. The quantized nonzero components and position information are encoded using a Huffman coder. The experiments are run on a PC with Intel®Core®i7-3770 CPU @ 3.40GHz with 7.8 GB memory.

5.4.1 Experimental Data Sets

The data sets used in the experiments are mainly AVIRIS images. They are grouped as follows.

Group 1 The first three data sets are the ones used in [Du and Fowler 2007], and they are termed as Cuprite I, Jasper Ridge, and Moffet Field I here. Figure 5.4 shows the grey scale images of the three data sets. These data sets are commonly used as benchmark data for hyperspectral image compression in the related works. The sizes of the images are 512×512. The data is available online from `https://aviris.jpl.nasa.gov/data/free_data.html`.

Group 2 The next three data sets, named Indian Pines, University of Pavia, and Salinas, are commonly used for the task of hyperspectral image classification. Their respective sizes are $145 \times 145 \times 200$, $610 \times 340 \times 103$, and $512 \times 217 \times 204$ after removing some noise bands. The data is available online from `https://engineering.purdue.edu/~biehl/MultiSpec/`. Further details of these three data sets are given in Chapter 6.

Group 3 The last group of data sets is obtained at `https://aviris.jpl.nasa.gov/alt_locator/`. These data sets are radiance data stored in 16 bit signed integers with 224 bands. The dictionaries concerned in this thesis are trained on these data sets using cropped sub-cubes of size $1024 \times 512 \times 224$. The site names used in this thesis for these data sets are listed below, together with their respective file names and the scopes of the training and testing sub-cubes. For instance, “Train: 51 - 1074; 51 - 562; 1 - 224” in the following list means this sub-cube starts at row (or line) 51, column (or sample) 51, and ends at row 1074, column 562 including all the bands, i.e., 224 bands. Further, the sub-cube was used only for training a dictionary. When a sub-cube was used for compression, it is indicated as “Test”. The images of some of these data sets are displayed in Figure 5.6.

a) Lunar Lake (f090819t01p00r06 — Train: 51 - 1074; 51 - 562; 1 - 224. Test: 1537 - 2048; 51 - 562; 1 - 224);
b) Moffett Field II (f080611t01p00r07 — Train: 51 - 1074; 51 - 562; 1 - 224. Test: 1025 - 1536; 51 - 562; 1 - 224);
c) Bay Area I (f140418t01p00r08 — Train: 1 - 1024; 51 - 562; 1 - 224. Test: 1025 - 1536; 51 - 562; 1 - 224);
d) Bay Area II (f141124t01p00r08 — Test: 51 - 562; 51 - 562; 1 - 224);
e) Bay Area III (f140528t01p00r16 — Test: 100 - 611; 512 - 1023; 1 - 224);
f) Cuprite II (f090819t01p00r11 — Test: 51 - 562; 51 - 562; 1 -224);
g) Cuprite III (f080611t01p00r06 — Train: 51 - 1074; 51 - 562; 1 - 224);

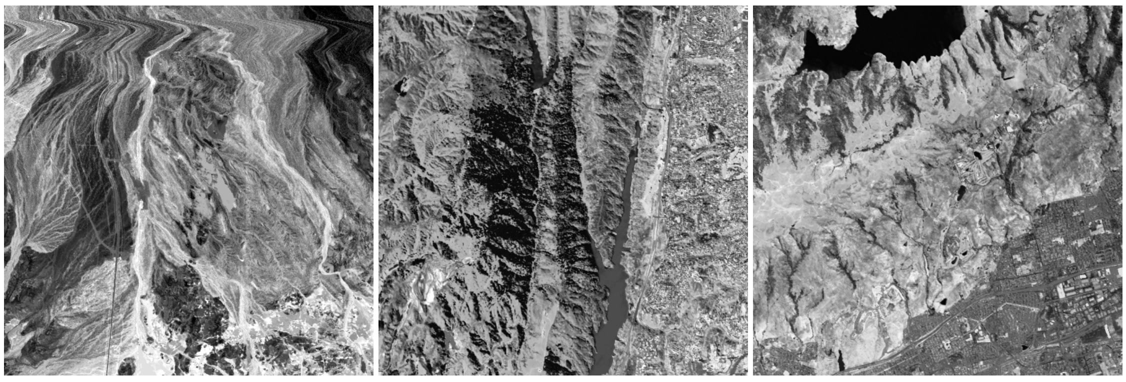


Figure 5.4: Grayscale images of Group 1 data sets (a) Cuprite I, (b) Jasper Ridge, and (c) Moffett Field I.

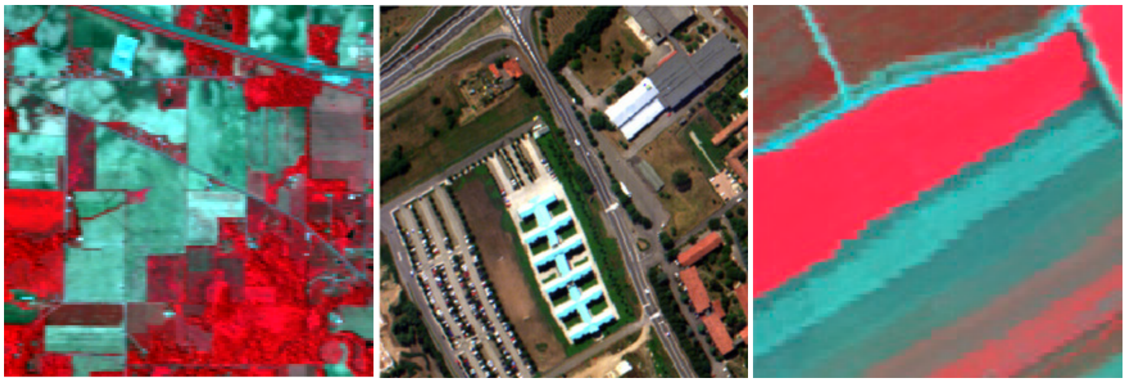


Figure 5.5: False colour images of Group 2 data sets (a) Indian Pines, (b) University of Pavia (cropped), and Salinas (cropped)

5.4.2 Experiment Setup

The numerical results to be presented use an MSE based distortion measure, which is signal to noise ratio (SNR), as well as two measures of data quality. The SNR is computed as

$$\mathrm{SNR} = 10\log_{10}(\sigma^2/\mathrm{MSE}), \tag{5.9}$$

where

$$\mathrm{MSE} = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(x_{i,j}-\hat{x}_{i,j})^2, \tag{5.10}$$

σ^2 is the variance of input signal, and computed as

$$\sigma^2 = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(x_{i,j}-\overline{\mathbf{X}}), \tag{5.11}$$

where $\overline{\mathbf{X}}$ is the mean value of $\mathbf{X}$.

The two quality measures include spectral angle mapper (SAM) for evaluating spectral fidelity, and anomaly detection based *spatial correlation coefficient (ρ)* for data analysis purpose. The SAM measures the similarity between two vectors by computing the angle between them, i.e.,

$$\mathrm{SAM}(\mathbf{x},\hat{\mathbf{x}}) = \arccos\frac{\mathbf{x}^{\mathrm{T}}\hat{\mathbf{x}}}{\|\mathbf{x}\|_2\|\hat{\mathbf{x}}\|_2}. \tag{5.12}$$

The SAM distortions were evaluated in terms of

- p_{SAM} — the largest SAM distortion;
- μ_{SAM} — the mean of SAM;
- σ_{SAM} — the standard deviation of SAM

in degrees over the entire data set.

The data quality measure with regard to anomaly detection is based on the anomaly detector described in Section 2.2.2 on Pg. 15. An anomaly in a hyperspectral image is likely a target, and its spectral signature is usually very different from those of its background. It is important to preserve such information in the lossy compression of hyperspectral images for data analysis purpose. To obtain a measure of such quality of the reconstructed data, we considered the detection map from the original data to be the ground truth, and the one resulting from the reconstructed data was compared to it. The detection maps for the original and reconstructed data were computed using (2.6) on Pg. 16. The similarity of these two maps was then measured using the spatial correlation coefficient, ρ, which is computed as

$$\rho(\mathbf{Y},\hat{\mathbf{Y}}) = \frac{\sum_i\sum_j(y_{i,j}-\overline{\mathbf{Y}})(\hat{y}_{i,j}-\overline{\hat{\mathbf{Y}}})}{\sqrt{\sum_i\sum_j(y_{i,j}-\overline{\mathbf{Y}})^2\sum_i\sum_j(\hat{y}_{i,j}-\overline{\hat{\mathbf{Y}}})^2}}, \tag{5.13}$$

where $\mathbf{Y}$, $\hat{\mathbf{Y}}$ are the detection maps for the original and reconstructed data, respectively, $0 \le \rho \le 1$, and the similarity increases when ρ increases to 1.

The rate measurements are taken as bits per pixel per band (bpppb). The rate control in the experiments is achieved by setting the maximum number of nonzero coefficients during sparse coding.

The compression performances on the selected experimental data sets are compared to those using 3D-SPIHT and JPEG 2000 with a multicomponent transform, both are embedded wavelet based coders. For JPEG 2000 multicomponent, PCA and DWT are chosen for spectral decorrelation, and they are termed PCA+JPEG2000 and DWT+JPEG2000, respectively. The 3D-SPIHT implementation we use is *QccPack* (`http://qccpack.sourceforge.net`) [Fowler 2000]. We use *Kakadu* version 7.7 (`http://kakadusoftware.com`), which has an implementation of Part 2 of JPEG 2000 standard. For both 3D-SPIHT and JPEG 2000, we use the *Cohen-Daubechies-Feauveau 9-7 filter* with symmetric extension. The quantization step size used in JPEG 2000 is 10^{-5}.

With a fixed dictionary, the efficiency of sparse coding is fundamental to the data representation based on the given dictionary. Hence, besides the data compression performance, we also present the performance of sparse coding by giving some experimental results on the MSE corresponding to the sparsity of the coefficient vectors.

For data sets in Group 1 and Group 2, the dictionary training and compression were performed on the same data sets, and hence, the trained dictionaries were included in the compressed bit streams. For data sets in Group 3, however, the dictionary training and compression were conducted on different sub-cubes. In this case, the base dictionaries were updated according to the data to be compressed, and the difference between the base and updated dictionaries were transmitted to the decoder.

5.4.3 Results of Experiments and Discussion

This section demonstrates the compression performance of the proposed framework. The performance is inspected on various aspects, such as compression performance, comparison between the effectiveness of base and updated dictionaries, the bit distribution, and computational complexity. We discuss results that illustrate each of these separately in the following subsections.

5.4.3.1 Compression Performance

The comparisons of rate-distortions and other quality measures using various compression methods are illustrated in Tables 5.3, 5.5, and 5.6. In these tables, the SNR, spectral fidelity, and spatial correlation coefficients are compared for DWT+JPEG2000, PCA+JPEG2000, 3D-SPIHT, the sparse representation based method using spectral dictionary (SRSD), and using multi-scale spectral dictionary (MSSD).

Due to the high computational cost in dictionary training, it is desirable to employ dictionaries trained offline in the proposed compression framework. However, in order to compare the performance of the proposed method with the DWT or PCA coupled with JPEG 2000, we trained spectral dictionaries using each of the 3 sets of data in Group 1, namely, Cuprite I, Jasper Ridge and Moffett I, used in [Du and Fowler 2007]. The data sets were then compressed using their corresponding dictionaries, respectively. The resulting performance is shown in Table 5.3, where it is compared to the results using PCA+JPEG2000 and DWT+JPEG2000. We observe from Table 5.3 that the sparse representation based methods perform inferior to both DWT+JPEG2000 and PCA+JPEG2000 at lower bit rates. Note that the performance is compared on all the criteria that these methods are evaluated on. At higher bit rates (> 0.25 bpppb), both SRSD and MSSD perform similar or slightly better than DWT+JPEG2000, while PCA+JPEG2000 performs the best on the evaluated criteria.

Table 5.3: Compression Performance of Various Methods for Group 1 Data Sets

		0.10 bpppb				0.25 bpppb				0.50 bpppb				1.0 bpppb			
		DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD
Cuprite I	SNR	38.7	**44.4**	32.7	33.4	43.1	**47.9**	41.5	43.7	46.7	**50.4**	47.1	47.3	51.0	**54.0**	50.0	49.7
	ρ	0.62	**0.65**	0.27	0.56	**0.74**	0.69	0.66	0.66	**0.74**	0.65	0.69	0.71	**0.79**	—	0.72	0.69
	p_{SAM}	1.43	**0.94**	7.20	5.12	0.61	**0.59**	1.92	1.92	**0.33**	0.37	0.92	0.98	**0.32**	—	0.62	0.55
	μ_{SAM}	0.44	**0.25**	0.91	0.61	0.28	**0.17**	0.33	0.25	0.19	**0.13**	0.17	0.17	**0.12**	—	0.13	0.13
	σ_{SAM}	0.13	**0.04**	0.20	0.36	0.06	**0.02**	0.07	0.07	0.13	**0.01**	0.02	0.03	0.02	—	**0.01**	**0.01**
Jasper Ridge	SNR	28.9	**36.8**	26.2	24.3	34.5	**43.0**	35.5	36.4	39.2	**46.5**	42.4	42.4	44.8	**50.3**	46.1	45.7
	ρ	0.63	**0.95**	0.41	0.51	0.90	**0.91**	0.81	0.77	**0.94**	0.90	0.86	0.80	**0.95**	—	0.93	0.86
	p_{SAM}	4.38	**2.37**	9.74	3.89	1.71	**0.66**	5.09	5.81	0.95	**0.34**	1.39	3.75	**0.90**	—	0.91	2.94
	μ_{SAM}	1.47	**0.62**	1.94	1.99	0.81	**0.32**	0.68	0.59	0.48	**0.21**	0.31	0.32	0.26	—	**0.21**	0.22
	σ_{SAM}	0.56	**0.17**	0.66	1.27	0.27	**0.07**	0.20	0.23	0.14	**0.05**	0.07	0.08	0.06	—	**0.04**	0.05
Moffett I	SNR	29.7	**37.5**	26.2	25.0	35.5	**43.6**	35.9	36.7	40.2	**47.0**	43.0	42.6	45.5	**50.9**	46.5	46.2
	ρ	0.57	**0.74**	0.35	0.48	**0.68**	0.43	0.58	0.62	**0.77**	**0.77**	0.68	0.68	**0.86**	—	0.79	0.80
	p_{SAM}	2.66	1.77	9.30	**1.24**	1.17	**0.67**	4.42	3.50	0.54	**0.23**	3.47	2.72	0.99	—	3.38	**0.30**
	μ_{SAM}	1.21	**0.52**	1.81	1.70	0.66	**0.28**	0.60	0.53	0.40	**0.19**	0.27	0.29	0.22	—	**0.18**	0.20
	σ_{SAM}	0.48	**0.15**	0.53	1.12	0.21	**0.07**	0.20	0.23	0.11	**0.05**	0.06	0.08	0.05	—	**0.03**	0.04

DWT: DWT+JPEG2000; PCA: PCA+JPEG2000; SRSD: sparse representation using SD; MSSD: sparse representation using multi-scale SD.
—: For PCA+JPEG2000, only up to 54 principle components were used due to the line restrictions on giving the transform matrix in demo version of Kakadu 7.7. Thus the results for PCA+JPEG2000 at bit rate 1.0 were not presented here.
Dictionary sizes: i) SRSD for all three data sets: $224 * 448$; MSSD for all three data sets: $224 * 448$ for the LL component, and $224 * 269$ for the LH, HL, HH components.

Recall that only a few of the dictionary atoms are chosen to represent one pixel in sparse representation, while in PCA, essentially all of the principle components in the transform matrix are used to represent a pixel. This could be one of the reasons why the methods based on sparse representation perform poorly at lower bit rates, where usually only a couple of dictionary atoms are chosen for representing an input sample.

Although MSSD is expected to improve the performance of SRSD, due to the cost of transmitting the multiple dictionaries, the performance gain is likely traded off. In particular, the performance of MSSD at lower bit rates is very limited as the dictionaries make up a significant fraction of the final bit stream. At higher bit rates, however, the effect becomes less significant. To illustrate the impact of transmitting the dictionaries in MSSD, we compared the performance of MSSD from Table 5.3 to the one where the dictionaries were not transmitted (MSSD^{-}). These results are shown in Table 5.4. The dictionaries used for MSSD (in Tables 5.3 and 5.4) were 224×448 for the wavelet LL component, and 224×269 for the 3 other components (only one level of DWT was considered here).

Table 5.5 shows the results for the compression of the data sets in Group 3. For Bay Area I and Bay Area II, an SD or MSSD learned on a sub-block of Bay Area I is used as the base dictionary; For Lunar Lake and Moffett Field II data, SDs or MSSDs trained on the sub blocks of these two data are employed as the base dictionaries, respectively. The data sets being compressed in the experiments are chosen as a sub cube of $512 \times 512 \times 224$ excluding the data for training dictionaries. For multi-scale dictionaries in wavelet domain, only one level of wavelet decomposition is considered due to the overhead of transmitting the information on updated dictionaries. In Table 5.5, the performance of SRSD is comparable or better (at higher bit rates) than that of DWT+JPEG2000 in terms of SNR. However, the improved performance on SNR does not necessarily translate to other quality criteria, especially in the case of spectral fidelity for the strongest distortion pixel, i.e., p_{SAM}. The method of MSSD improves on most of the assessment criteria, except p_{SAM}, compared to SRSD for most of the data sets considered. However, this is not the case for Moffett II data set. One possible reason for this scenario is that this particular data

Table 5.4: Compression Performance of MSSD with or without the Dictionaries Being Transmitted

		0.1 bpppb		0.25 bpppb		0.5 bpppb		1.0 bpppb	
		MSSD	MSSD$^-$	MSSD	MSSD$^-$	MSSD	MSSD$^-$	MSSD	MSSD$^-$
Cuprite I	SNR	33.4	36.6	43.7	44.8	47.3	47.7	49.7	50.2
	ρ	0.56	0.56	0.66	0.67	0.71	0.71	0.69	0.76
	p_{SAM}	5.12	2.54	1.92	1.45	0.98	0.81	0.55	0.54
	μ_{SAM}	0.61	0.50	0.25	0.22	0.17	0.17	0.13	0.13
	σ_{SAM}	0.36	0.22	0.07	0.05	0.03	0.02	0.01	0.01
Jasper Ridge	SNR	24.3	28.0	36.4	38.1	42.4	42.9	45.7	46.1
	ρ	0.51	0.68	0.77	0.75	0.80	0.79	0.86	0.84
	p_{SAM}	3.89	5.0	5.81	4.92	3.75	3.55	2.94	2.94
	μ_{SAM}	1.99	1.46	0.59	0.49	0.32	0.30	0.22	0.21
	σ_{SAM}	1.27	0.69	0.23	0.16	0.08	0.07	0.05	0.04
Moffett I	SNR	25.0	28.2	36.7	38.7	42.6	43.4	46.2	46.8
	ρ	0.48	0.47	0.62	0.58	0.68	0.68	0.80	0.81
	p_{SAM}	1.24	1.70	3.50	3.66	2.72	2.52	0.30	0.27
	μ_{SAM}	1.70	1.34	0.53	0.44	0.29	0.27	0.20	0.19
	σ_{SAM}	1.12	0.64	0.23	0.16	0.08	0.07	0.04	0.04

The results for MSSD are the same as in Table 5.3.
MSSD$^-$: The dictionaries were not included in the bit stream.

set features mainly of urban area, where wavelet decomposition is not able to exploit the spatial redundancies in the scene adequately.

Compared to the others, PCA+JPEG2000 outperforms the other methods on all assessment criteria by considerable margin. One of the main reasons that the method SRSD performs worse than PCA+JPEG2000 is that it exploits only spectral correlations, while both spectral and spatial correlations are exploited in PCA+JPEG2000. After sparse coding using a dictionary, the sparse representation coefficients are highly sparse and it makes it challenging to further exploit spatial correlations within a band using SRSD. Although MSSD does exploit both spatial and spectral redundancy in hyperspectral images, the extent in the case of spatial redundancy is limited to very few wavelet decomposition levels due to the transmission overhead of dictionary.

The advantage of MSSD is that it offers a certain degree of flexibility. On one hand, the flexibility could be in terms of the choices for dictionaries. It is possible to train a dictionary for each component, or train a single dictionary on different components combined together. For instance, we can train a dictionary for the LL component, and train another dictionary on the rest of the components combined. In our experiments, we observed that individually trained dictionaries for different wavelet components gave better compression performance than using the ones trained on combined components. On the other hand, the flexibility could also be bit allocations. Using MSSD, we can allocate bits to the wavelet components according to their importance or relevance. In our experiments, more bits were allocated to the LL components than others. This is done by assigning different μs, i.e., the maximum number of nonzero coefficients, for different components in sparse coding. For example, if $\mu = 6$ for the sparse coding of the LL component, we can assign μs that are less than 6 for the other components.

Compared to the data sets in Groups 1 and 3, the data sets in Group 2 are extensively used in hyperspectral image classification, as labelled data samples exist for these data (Chapter 6 presents the details). Since the hyperspectral images in this group feature land covers such as agriculture and urban area, they present different image features from those in Groups 1 and 3. As such, we investigated the compression performance of various methods for the three hyperspectral images in Group 2. The results are compared in Table 5.6. A notable difference from the experiments on two other groups of data is that the dictionaries for data sets in Group 2 were initialized using the

Table 5.5: Compression Performance of Various Methods for Group 3 Data Sets

		0.2 bpppb					0.4 bpppb					0.6 bpppb				
		DWT	PCA	SRSD	MSSD	SPIHT	DWT	PCA	SRSD	MSSD	SPIHT	DWT	PCA	SRSD	MSSD	SPIHT
	SNR	34.4	**45.6**	32.0	34.9	25.5	39.5	**48.5**	39.6	41.8	30.3	42.8	**50.0**	42.8	45.1	33.9
	ρ	0.47	**0.81**	0.31	0.46	0.28	0.61	**0.88**	0.45	0.52	0.32	0.67	**0.88**	0.57	0.59	0.37
Bay	p_{SAM}	4.37	**1.08**	13.7	7.41	10.1	2.05	**0.80**	6.87	4.36	4.44	1.34	**0.78**	5.79	2.95	2.84
Area I	μ_{SAM}	0.71	**0.21**	0.91	0.64	1.98	0.41	**0.15**	0.38	0.30	1.17	0.28	**0.13**	0.27	0.21	0.78
	σ_{SAM}	0.32	**0.05**	0.70	0.38	0.66	0.15	**0.03**	0.22	0.15	0.38	0.09	**0.02**	0.13	0.07	0.23
	SNR	29.6	**40.3**	29.1	31.7	21.2	34.3	**43.2**	37.4	38.4	25.1	37.1	**44.6**	40.3	40.7	28.0
	ρ	0.56	**0.82**	0.53	0.47	0.49	0.66	**0.88**	0.59	0.51	0.49	0.71	**0.88**	0.63	0.53	0.53
Bay	p_{SAM}	5.66	**1.56**	5.96	5.82	13.3	3.01	**1.44**	2.75	4.64	7.85	2.14	**1.42**	1.77	3.54	4.58
Area II	μ_{SAM}	1.27	**0.39**	1.23	1.02	3.07	0.75	**0.28**	0.48	0.48	2.03	0.55	**0.23**	0.35	0.37	1.49
	σ_{SAM}	0.49	**0.11**	0.45	0.36	0.95	0.26	**0.06**	0.14	0.13	0.57	0.17	**0.05**	0.08	0.09	0.40
	SNR	36.2	**46.5**	35.8	38.7	25.7	40.9	**48.9**	41.6	43.9	29.5	43.6	**50.3**	45.3	46.1	32.4
	ρ	0.55	**0.76**	0.47	0.63	0.49	0.66	**0.80**	0.59	0.64	0.59	0.72	**0.81**	0.67	0.57	0.60
Lunar	p_{SAM}	3.39	**0.86**	4.90	2.22	7.62	1.67	**0.84**	1.41	1.09	4.21	1.10	**0.83**	0.85	0.93	2.67
Lake	μ_{SAM}	0.54	**0.17**	0.55	0.40	1.65	0.32	**0.13**	0.28	0.22	1.12	0.24	**0.11**	0.19	0.17	0.81
	σ_{SAM}	0.17	**0.03**	0.15	0.12	0.39	0.09	**0.02**	0.06	0.05	0.29	0.05	**0.02**	0.03	0.03	0.20
	SNR	39.0	**47.7**	38.8	39.9	27.4	43.5	**50.0**	43.8	44.9	31.7	46.0	**51.3**	46.1	47.2	34.7
	ρ	0.84	**0.94**	0.72	0.74	0.35	0.83	**0.95**	0.71	0.75	0.46	0.81	**0.95**	0.67	0.77	0.51
Cuprite II	p_{SAM}	3.34	**1.14**	5.44	6.98	8.75	1.58	**1.05**	4.11	3.67	4.36	1.05	**1.04**	2.06	2.99	3.30
	μ_{SAM}	0.40	**0.16**	0.32	0.32	1.36	0.24	**0.12**	0.21	0.19	0.87	0.19	**0.10**	0.17	0.15	0.64
	σ_{SAM}	0.13	**0.02**	0.14	0.15	0.30	0.06	**0.02**	0.07	0.06	0.21	0.04	**0.01**	0.04	0.03	0.15
	SNR	24.3	**31.5**	25.3	25.0	17.9	28.8	**37.4**	33.9	31.6	21.4	32.2	**41.7**	38.1	34.6	24.2
	ρ	0.84	**0.93**	0.86	0.61	0.75	0.85	0.91	**0.96**	0.78	0.74	0.89	0.90	**0.96**	0.77	0.75
Moffett II	p_{SAM}	12.7	**8.59**	16.8	30.7	17.9	**7.31**	8.29	7.33	26.0	13.6	**4.93**	8.27	5.00	22.5	9.88
	μ_{SAM}	2.47	**1.12**	1.84	2.02	4.77	1.50	**0.57**	0.68	0.96	3.41	1.02	**0.34**	0.43	0.66	2.54
	σ_{SAM}	0.76	**0.33**	0.86	0.89	1.44	0.41	**0.16**	0.29	0.39	0.92	0.27	**0.09**	0.14	0.28	0.70

DWT: DWT+JPEG2000; PCA: PCA+JPEG2000; SRSD: sparse representation using SD; MSSD: sparse representation using multi-scale SD; SPIHT: 3D-SPIHT.
ρ: spatial correlation coefficient; p_{SAM}: SAM distortion for the strongest anomalous pixel; $\mu_{\mathrm{SAM}}, \sigma_{\mathrm{SAM}}$: mean and standard deviation of SAM over entire data set; (SAM related values are in degrees.)
For PCA+JPEG2000, only up to 54 principle components were used due to the line restrictions on giving the transform matrix in demo version of Kakadu 7.7. Thus the results at bit rate 0.6*bpppb* for PCA+JPEG2000 should be slightly worse than using all or optimal number of principle components.
Dictionary sizes: SRSD – The dictionaries for all the data sets were of size $224 * 448$; MSSD dictionaries for all the data sets were $224 * 448$ for all the components.

labelled samples, and trained over the entire data. In particular, the dictionaries were initialized using a certain fraction of the labelled samples from each class. Although the PCA+JPEG2000 mostly outperformed the other methods, it performed inferior to SRSD on Indian Pines data set. A possible reason is that performing DWT on this particular data set failed to exploit adequately the spatial correlations among the pixels.

5.4.3.2 The Comparison of Base and Updated Dictionaries

In sparse representation, a dictionary is learned so that it adapts to the training data optimally. It is prohibitive to learn a dictionary for each newly acquired hyperspectral image both computationally and out of bit rate considerations in data compression. In the proposed approach, this problem is addressed by updating a base dictionary, where the base dictionary is trained offline using previously obtained hyperspectral images from the same region. Thus, using the proposed framework, the first step is to update the base dictionary using the data to be compressed. This is effective since the base dictionary is obtained using the hyperspectral image from a similar region. This assumption was verified in our experiments. After a small number of iterations, the base dictionary can be updated to a well adapted one to represent the current data. Table 5.7 shows the reconstruction performance comparisons between the base and updated dictionaries. In this particular case, the base dictionary was trained using Bay Area I, and the compression was performed using Bay Area II and Bay Area III data sets, respectively. The updated dictionaries

Table 5.6: Compression Performance of Various Methods for Group 2 Data Sets

		0.20 bpppb				0.40 bpppb				0.60 bpppb				1.0 bpppb			
		DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD	DWT	PCA	SRSD	MSSD
Indian Pines	SNR	26.7	27.8	**31.4**	27.9	29.8	31.5	**33.8**	30.7	32.7	34.3	**35.3**	32.4	**37.2**	—	**37.2**	34.9
	ρ	0.73	0.39	**0.78**	0.64	0.76	0.75	**0.83**	0.67	0.84	0.79	**0.85**	0.68	**0.93**	—	0.86	0.74
	p_{SAM}	4.51	5.73	**4.25**	7.00	**3.30**	5.22	3.62	4.32	**2.18**	5.08	3.25	4.19	**0.92**	—	2.87	3.83
	μ_{SAM}	1.32	1.17	**0.74**	1.06	0.93	0.76	**0.56**	0.78	0.67	0.55	**0.48**	0.64	0.40	—	**0.39**	0.49
	σ_{SAM}	0.33	**0.32**	**0.32**	0.55	0.20	**0.19**	0.25	0.39	**0.12**	0.15	0.20	0.32	**0.06**	—	0.14	0.22
University of Pavia	SNR	26.3	**27.1**	23.6	23.2	30.3	**30.8**	27.1	26.1	32.6	**32.9**	28.7	28.9	35.8	**36.2**	30.3	31.6
	ρ	**0.91**	0.88	0.65	0.67	**0.96**	0.86	0.78	0.70	**0.97**	0.89	0.79	0.71	**0.98**	0.96	0.86	0.77
	p_{SAM}	22.6	**20.7**	29.5	25.3	12.8	**12.6**	28.5	22.9	**9.79**	9.86	26.7	20.5	**6.39**	6.40	22.8	17.9
	μ_{SAM}	1.80	**1.66**	2.17	2.28	1.16	**1.10**	1.46	1.64	0.90	**0.86**	1.21	1.20	0.63	**0.60**	0.99	0.87
	σ_{SAM}	1.23	**1.12**	1.20	1.45	0.77	**0.72**	0.85	1.11	0.59	**0.56**	0.72	0.77	0.41	**0.39**	0.61	0.55
Salinas	SNR	37.8	**43.6**	38.7	39.4	41.7	**46.0**	42.4	42.7	44.1	**47.5**	43.6	44.1	**47.2**	—	45.4	45.4
	ρ	0.93	**0.98**	0.92	0.66	0.94	**0.98**	0.94	0.68	0.95	**0.98**	0.94	0.74	**0.97**	—	0.93	0.78
	p_{SAM}	6.15	**3.48**	9.01	30.0	**3.16**	3.36	4.29	28.9	**2.04**	3.35	3.96	27.7	**1.12**	—	3.62	26.7
	μ_{SAM}	0.50	**0.27**	0.43	0.38	0.33	**0.21**	0.29	0.27	0.25	**0.17**	0.25	0.24	**0.18**	—	0.21	0.20
	σ_{SAM}	0.23	**0.08**	0.22	0.27	0.12	**0.06**	0.12	0.17	0.08	**0.05**	0.08	0.14	**0.05**	—	**0.05**	0.12

DWT: DWT+JPEG2000; PCA: PCA+JPEG2000; SRSD: sparse representation using SD; MSSD: sparse representation using multi-scale SD.
—: For PCA+JPEG2000, only up to 54 principle components were used due to the line restrictions on giving the transform matrix in demo version of Kakadu 7.7. Thus the results for PCA+JPEG2000 at bit rate 1.0 were not presented here.
Dictionary sizes: i) Indian Pines – SRSD: $200 * 828$; MSSD: $200 * 600$ for the LL component, and $200 * 360$ for the LH, HL, HH components. ii) University of Pavia – SRSD: $103 * 288$ MSSD: $103 * 206$ for the LL component, and $103 * 165$ for the LH, HL, HH components. iii) Salinas – SRSD: $204 * 384$ MSSD: $204 * 408$ for the LL component, and $204 * 327$ for the LH, HL, HH components.

Table 5.7: Performance Comparison of the Base and Updated Dictionaries

		Bay Area II			Bay Area III		
	Bit Rate	SNR	ρ	μ_{SAM}	SNR	ρ	μ_{SAM}
Base D	0.25	22.9	0.26	2.48	20.7	0.37	3.08
Updated D		32.8	0.50	0.79	34.8	0.58	0.55
Base D	0.35	24.2	0.35	2.06	22.4	0.35	2.52
Updated D		35.4	0.54	0.59	38.2	0.67	0.41

were trained for 500 iterations given the base dictionary. As a result, all the performance criteria were significantly improved using the updated dictionary.

5.4.3.3 Tuning Dictionary Learning Parameters

We expect that the larger the dictionary size is, the better the compression performance becomes. This is due to the increase in representative capacity of a dictionary when the size grows. However, a larger dictionary could raise the following challenges in sparse representation based hyperspectral image compression.

1. It increases the dimension of the coefficient vectors, which results in increased bits for encoding the wider range of position indexes, as well as encoding more information on the dictionary.
2. The demands for both computational and available training data size increase in learning a larger dictionary. On the other hand, the impact of larger dictionary on the compression performance was not significant in our experiments. This is shown in Figure 5.7, where we hardly observe any significant difference of SNR performance using different sizes of dictionary.

Table 5.8: Bit Allocation for Various Components using SRSD

	0.25 bpppb			0.5 bpppb			1.0 bpppb		
	Coefficient	Position	Other	Coefficient	Position	Other	Coefficient	Position	Other
Cuprite I	0.47	0.39	0.14	0.48	0.46	0.06	0.50	0.46	0.04
Jasper Ridge	0.44	0.41	0.15	0.46	0.47	0.07	0.49	0.47	0.04
Moffett I	0.43	0.43	0.14	0.44	0.49	0.07	0.47	0.48	0.05

The numerical values are in $\times 100\%$.
Coefficient: sparse coefficients; Position: position indexes; Other: the remaining components.

In the proposed framework, the difference between the base and updated dictionaries is encoded in the bit stream. Thus, it is more advantageous to employ a smaller dictionary than a larger one. For these reasons, we mostly chose the parameter c (the ratio between the size and the dimension of the dictionary atoms) as $2 \leq c \leq 4$ to achieve a balance among SNR performance, bit rate and computational complexity.

Different values for sparsity inducing parameter μ during the stage of dictionary training (see Eq. (5.3)) may have different effects on sparse coding, and consequently, on compression performance. Some experiment results in this regard are presented in Figure 5.8. For the results in Figure 5.8, the dictionaries were trained using $\mu = 10, 12, 16$, respectively. The dictionaries were then used to compress a sub cube of Bay Area I. In Figure 5.8 (a), the dictionary size is 224×672, and in Figure 5.8(b), it is 224×896. Although the compression performances using various values for μ are similar to one another, it is observed that the lower the value of μ is, the slightly higher the SNR value for lower bit rates are. At the same time, the growth rates of SNR values at higher bit rates become lower than those with higher μ values.

5.4.3.4 The Drawbacks of the Proposed Method

The compression results for SRSD and MSSD presented in this section were obtained from a naive implementation of the proposed approach. Hence, the results are still not optimal. Besides the implementation issues, there exist a number of overheads in the bit stream. The encoding of the position indexes for the nonzero coefficients is prominent as such an overhead. Table 5.8 shows the bit allocations for different components in the final bit stream. The bit stream mainly consists of sparse coefficients, their corresponding position indexes, the DC component and dictionary information. It is evident that the position indexes make up a significant portion of the bit stream. However, we believe this information can be compressed more efficiently, which might result in better SNR to bit rate performance.

Compared to other transform based methods, the computational cost of sparse coding is relatively high. Table 5.9 shows the performance of sparse coding for some data sets using different sizes of dictionaries. The performance of sparse coding of each data set is measured for sparsity constraint, i.e., the number of nonzero coefficients μ, and the resulting MSE, p_{SAM}, ρ, bit rate, and time. If μ remains constant, then the values for MSE decrease when larger dictionaries are used. However, this also results in increased bit rates and more time for sparse coding. The performance for p_{SAM} and ρ gets better when μ increases. The dictionary size, however, shows little effect on these criteria.

Table 5.9: Sparse Coding Performance for 3 Selected Data Sets using SRSD

		$\mu = 4$					$\mu = 8$					$\mu = 12$				
	Size	MSE	p_{SAM}	ρ	Bit Rate	Time	MSE	p_{SAM}	ρ	Bit Rate	Time	MSE	p_{SAM}	ρ	Bit Rate	Time
Cup-	$c = 2$	136.02	1.31	0.67	0.32	2.13	52.17	0.81	0.69	0.62	3.46	34.67	0.64	0.71	0.94	4.69
rite I	$c = 3$	127.63	1.35	0.48	0.33	3.10	49.85	0.79	0.69	0.64	4.66	32.58	0.55	0.69	0.96	6.24
	$c = 4$	120.15	0.91	0.68	0.35	4.06	48.20	0.46	0.71	0.65	6.09	31.20	0.34	0.71	1.00	7.82
Moff-	$c = 2$	236.69	3.76	0.63	0.33	2.13	67.50	3.46	0.71	0.61	3.22	40.31	3.40	0.78	0.90	4.39
ett II	$c = 3$	206.20	3.58	0.70	0.34	3.07	62.63	2.61	0.66	0.63	4.62	37.92	2.04	0.81	0.93	6.38
	$c = 4$	193.25	3.68	0.71	0.35	4.06	59.44	3.02	0.77	0.65	6.11	36.33	2.61	0.84	0.96	7.89
Jasper	$c = 2$	208.13	2.89	0.84	0.34	2.13	60.21	1.26	0.87	0.64	3.21	35.19	0.94	0.92	0.95	4.38
Ridge	$c = 3$	187.86	2.76	0.86	0.35	3.07	55.77	1.30	0.88	0.66	4.64	33.04	0.64	0.93	0.98	6.22
	$c = 4$	175.04	3.01	0.87	0.36	4.10	53.27	1.23	0.90	0.67	6.10	31.85	0.71	0.94	1.00	8.37

Size: the parameter related to dictionary size. Given a dictionary of size $n \times k$, $k = cn$.
Time is in seconds using MATLAB 2014b implementation; Bit Rate: in bpppb.

5.5 Conclusion

Sparse representation models a set of data based on dictionary and sparse coding. The dictionary is learned in a way that adapts well to the data, and a sample in the data can be represented by just a few dictionary atoms. This type of transform often lends itself as a good candidate to the purpose of data compression due to its data compaction character. On the other hand, hyperspectral image compression presents its own challenges and demands methods beyond the existing data compression approaches designed for regular images. This chapter presents an investigation of a lossy hyperspectral image compression scheme based on sparse representation. In the proposed scheme, a hyperspectral data cube is modelled using a learned dictionary via sparse coding. The dictionaries are acquired by updating a base dictionary trained offline, or learning a new dictionary online from the data. In particular, two types of dictionaries – spectral dictionary and multi-scale spectral dictionary – are designed in order to exploit the rich correlations, respectively, in spectral or both spatial and spectral domains in hyperspectral images. In addition, various aspects of data compression, such as quantization and entropy coding, predictive coding, are investigated to formulate a feasible hyperspectral image compression framework. Despite the simplicity in implementing the framework itself, the experimental results compare favourably with some of the state-of-the-art compression methods, such as JPEG 2000 with multiple component couples with DWT or PCA, and 3D-SPIHT. The following conclusions are made.

1) The spectral dictionaries in SRSD method are effective for hyperspectral data compression, and could lead to competitive results compared to 3D-SPIHT and JPEG 2000 coupled with DWT.
2) Compared to SRSD, the dictionaries in MSSD exploits both spatial and spectral correlations in hyperspectral images. However, the benefit of this approach is often reduced due to the overhead of encoding the dictionaries.
3) Although training a data adaptive dictionary is computationally expensive, especially in the context of an encoder, it is feasible to incorporate a dictionary update step with a base dictionary trained offline. This could not only reduce the computational demand for dictionary training, but also reduce the overhead of dictionary transmission. If the base dictionary is trained offline from the data previously obtained in a similar region, a new dictionary can be learned with very little cost compared to learning the dictionary without such base dictionary.

A notable appeal of the proposed framework is that different approaches may be taken in learning a dictionary using sparse representation. For example, a task driven dictionary can be learned with a specific task in mind, such as classification or target detection. Under this scenario, besides energy compaction feature, the sparse coefficients might embody other features, e.g., discriminative, than energy compaction feature only. Such features can be exploited in other hyperspectral data processing problems. With this notion, the following chapter explores hyperspectral image classification based on sparse representation.

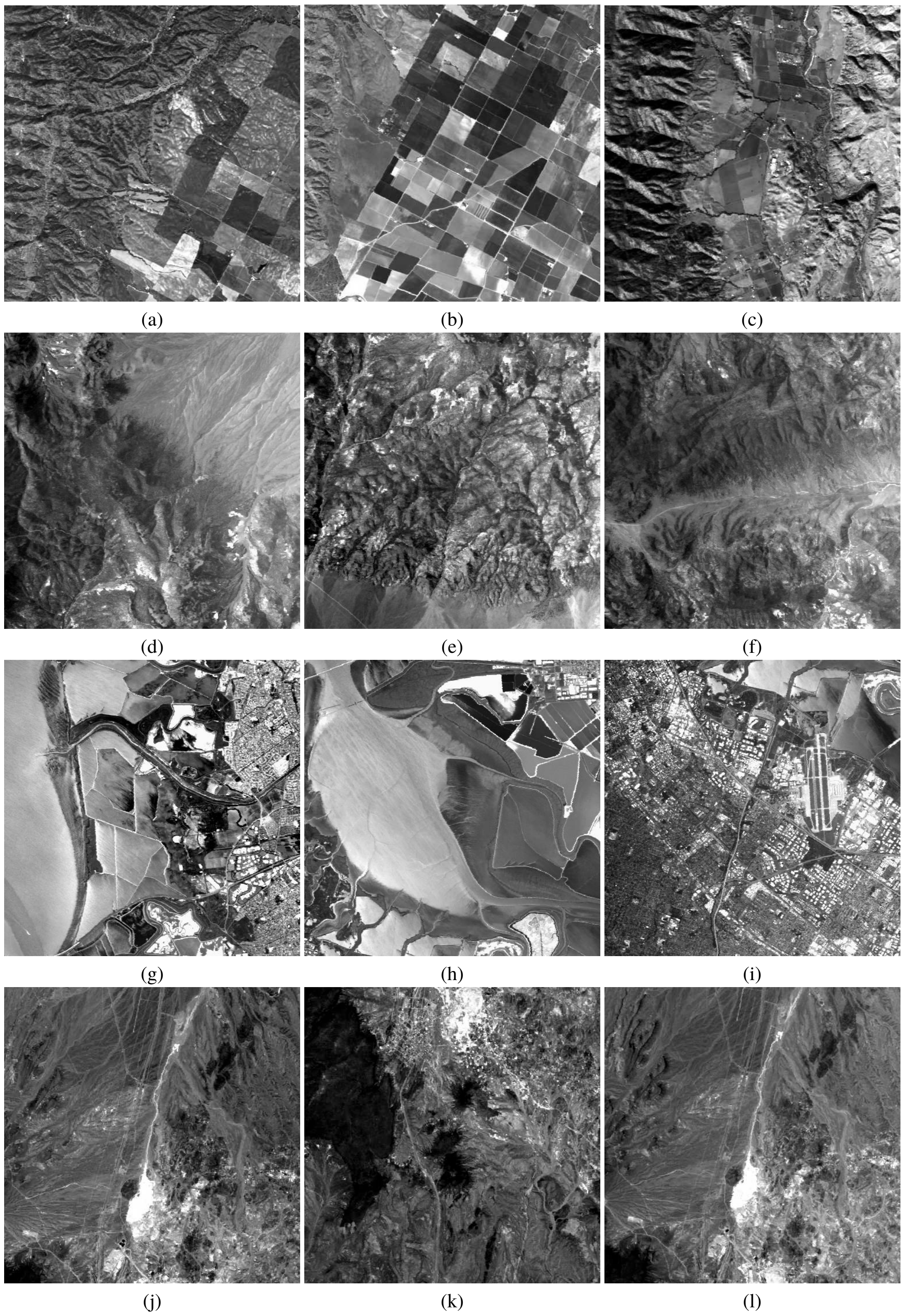
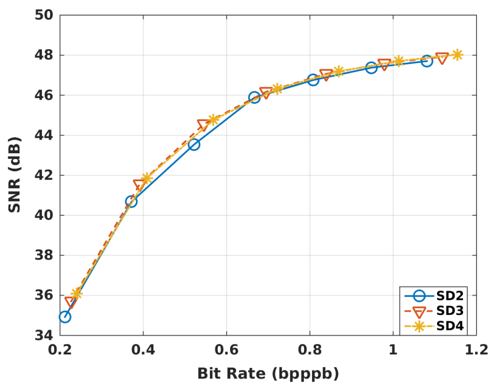
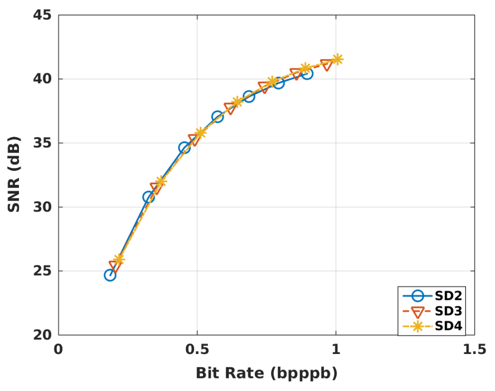


Figure 5.6: Grayscale images of Group 3 data sets used for dictionary training and for compression. Dictionary training: (a), (b) Bay Area I; (d), (e) Lunar Lake; (g), (h) Moffett Field II; (j), (k) Cuprite III; Compression: (c) Bay Area II; (f) Lunar Lake; (i) Moffett Field II; (l) Cuprite II.

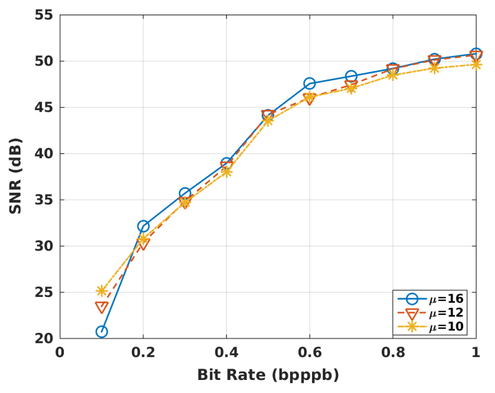


(a)

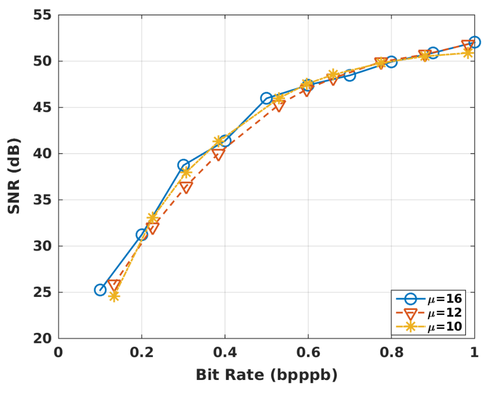


(b)

Figure 5.7: Rate-distortion curves for the SRSD compression of sub cubes from data sets (a) Lunar Lake and (b) Moffett Field II using different dictionary sizes — SD2 : 224×448; SD3 : 224×672; SD4 : 224×896. The results for SD2 here slightly differ from those in Table 5.5 due to the difference between the numbers of iterations used for dictionary update.



(a)



(b)

Figure 5.8: Rate-distortion curves for the compression of sub cubes from Bay Area I using various values of μ: (a) Dictionary size: 224×672, (b) Dictionary size: 224×896. (The results shown in this figure were obtained by using the base dictionaries only.)

Chapter 6

Sparse Representation Based Hyperspectral Image Classification

6.1 Introduction

Hyperspectral imaging is concerned with the measurement, analysis, and interpretation of spectra acquired from a given scene at a short, medium or long distance by an airborne or spaceborne sensor [Plaza et al. 2009a]. The information contained in hyperspectral data allows characterization, identification, and classification of materials in land, water, environment, and the atmosphere [Green et al. 1998; Goetz 2009]. The reflectance spectrum in hyperspectral data can be used to distinguish among a large range of land surface cover materials [Goetz et al. 1985]. With the recent advances of data acquisition technologies in geoscience and remote sensing, the spatial and spectral resolutions become higher. This leads to various applications of these data [Green et al. 1998], and active research in hyperspectral image processing. One of such applications and research is the hyperspectral image classification where pixels in hyperspectral image are assigned to one of a known set of classes. Intensive work has been done in the remote sensing community to develop effective classifiers for hyperspectral image [Fauvel et al. 2013].

Among the previous approaches, support vector machines (SVMs) [Schölkopf and Smola 2001] have become a powerful tool for supervised hyperspectral image classification [Mercier and Lenon 2003; Melgani and Bruzzone 2004; Fauvel et al. 2008]. The SVMs based on the spectral signatures of the classes often produce results with noisy looking classification maps. This shortcoming is overcome by using the contextual information in the classifiers, based on the fact that hyperspectral image pixels are often surrounded by those of the same class. Methods that exploit both spectral and spatial features are developed to further improve the hyperspectral image classification performance. In [Camps-Valls et al. 2006], a family of composite kernels is constructed to combine spatial and spectral information. It provides enhanced classification performance compared to the approaches using only the spectral or spatial information. A simple yet effective way to embed spatial characters in spectral pixel vectors is to consider for each pixel the features of its neighbouring ones. This is often done by using moving windows or filters applied to each spectral band. The filtered hyperspectral image pixels are then used to train a classifier.

In spite of the advances made in hyperspectral image classification, challenges still exist. One such challenge is the high dimensionality of hyperspectral image. As long as the number of training sets is limited, the abundant spectral or spatial information does not necessarily lead to high classification accuracy due to the Hughes effects [Hughes 1968]. It is desirable to distinguish a small number of spectral bands as the discriminative factor for the pixels belong to the same class. As such, sparsity based models naturally fit into this purpose. These models assume that the signal vectors lie in a combination of independent or dependent subspaces, where each subspace is spanned by a few elements from a set of basis vectors (here the term basis is used in a broader sense that the basis vectors are not necessarily orthogonal to each other). Using sparsity models, many types of signals can be sparsely represented through linear transforms. A common goal in such a transform is to find a set of vectors that span the vector space of input signals so that some criterion of optimality, such as discrimination or sparsity, can be met.

Sparse representation has found applications in hyperspectral image processing, and compression with encouraging results. Chen et al. [2011a] introduced a sparsity-based approach to classify hyperspectral image, where a test pixel is represented by a few training samples from a structured dictionary. The test pixel is then labelled as the class whose training samples yield the minimum reconstruction error in representing the test pixel. The classification performance is further improved by incorporating the contextual information in formulating the sparse recovery optimization problem.

While the dictionary in [Chen et al. 2011a] is formulated by the set of training samples, Soltani-Farani et al. [2013] is able to train a structured dictionary that incorporates both spectral and spatial characteristics of training hyperspectral image data. The sparse coefficients were obtained for the hyperspectral image samples through sparse coding using the learned dictionary. This is followed by training the SVMs using the sparse coefficients for classification. Both [Chen et al. 2011a] and [Soltani-Farani et al. 2013] utilize the joint sparsity [Tropp et al. 2006] in sparse coding or dictionary learning to combine spectral and contextual information. By incorporating additional structured sparsity priors such as Laplacian and group sparsity [Sun et al. 2014 2015; Qian et al. 2013], or adopting nearest subspace classification coupled with nonuniform regularization approach [Li and Du 2014; Li et al. 2014; Li and Du 2015], the sparse representation based hyperspectral image classifiers can be enhanced.

Wang and Celik [2017] investigates the lossy compression of hyperspectral image based on sparse representation with encouraging results. A special point of interest in [Wang and Celik 2017] is that the authors argued that the sparse coefficients could be used as feature vectors for hyperspectral image processing, such as classification. In such cases, the decoded sparse coefficients in the lossy compression framework can be analysed without reconstructing the original hyperspectral data. This is advantageous in terms of obtaining optimal bit rates in storing hyperspectral data on a remote sensor.

Nearest neighbour (NN) and k-nearest neighbour (kNN) [Blanzieri and Melgani 2008] classifiers are simple nonparametric ones in the sense that they do not assume any prior of the input features. Given an unlabelled sample, the classification is carried out by consulting within a set of labelled reference samples. In particular, kNN finds the k training samples that are closest to it. Various strategies can then be applied for the decision making process. Zou et al. [2015] proposed a sparse representation-based nearest neighbour (SRNN) classifier. The NN classifier is applied to the sparse coefficients obtained in the way similar to that of [Chen et al. 2011a]. The SRNN is also extended to utilize class-specific sparse coefficients, as well as to impose smoothness constraint in the neighbourhood of a pixel during sparse coding.

In this chapter, we propose a classification approach for hyperspectral image that utilizes sparse representation. The proposed method starts with learning a dictionary that induces sparsity in the representation coefficients, while minimizes the reconstruction error. In order to incorporate the contextual characteristics into the spectral sparse coefficients, we enforce smooth variations in the neighbourhood of pixels by applying a local filter on sparse coefficients. These contextual sparse coefficients are then used for two simple classifiers, namely SVMs and kNN, for their robustness and simplicity. The distinction between our approach and the existing methods is that the spatial correlation is considered in the sparse coefficient domain in our strategy, while in the previous classifiers, it is considered in the original pixel domain.

The contributions of our work in this chapter are twofold. Firstly, we showed that sparse representation could be used as an effective way to extract features that were both sparse and discriminative; the characteristic of sparsity allows us to address the problem of high dimensionality in hyperspectral image, and the discriminative feature leads to good performance in the classification of hyperspectral image. Secondly, the contextual information can be incorporated into the spectral features by applying a local mean filter in the sparse coefficient domain instead of in the original pixel domain. This approach yields a similar or better classification performance in terms of accuracy compared to other sparse representation based classifiers of hyperspectral image.

The remainder of this chapter is organized as follows. Section 6.2 introduces a number of hyperspectral image classifiers. Section 6.3 gives details of the proposed method. Experimental evaluation of various methods are reported in Section 6.4, and finally, Section 6.5 concludes the chapter.

6.2 Related Work

The previous work on classification of hyperspectral image is vast [Plaza et al. 2009a; Fauvel et al. 2013]. The purpose of this section is to introduce a number of selected hyperspectral image classifiers that are closely related to our work. The main focus is on SVM, kNN, and sparse representation based classifiers.

6.2.1 SVM Classification Approach

SVMs are considered as excellent classifiers for hyperspectral image due to their capacity to handle relatively few training samples and high dimensional input space, as well as the limited effort required for the design (or training) process [Melgani and Bruzzone 2004; Camps-Valls et al. 2006; Fauvel et al. 2008]. The technique involves a search of an optimal hyperplane that separates different classes.

6.2.1.1 SVM Formulation

The SVM was first introduced as a supervised binary classifier [Boser et al. 1992; Cortes and Vapnik 1995]. Given a labelled training data set $\{(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_N}, y_N)\}$, where $\mathbf{x}_i \in \mathbb{R}^M$, $y_i \in \{-1, +1\}$ for $i = 1, \cdots, N$, an SVM binary classifier tries to find a hyperplane that separates the two classes. Let us consider the following cases [Vapnik 1999; Melgani and Bruzzone 2004].

(1) Linear SVM: For linear SVM, the optimal case is that the two classes are linearly separable. That is, it is possible to find a hyperplane that separates the two classes without errors. Assume such a hyperplane is defined by a vector $\mathbf{w} \in \mathbb{R}^M$ and a bias $b \in \mathbb{R}$. Then the classification of a test sample can be determined by the sign of the discriminant function (or separating hyperplane)

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \tag{6.1}$$

where the symbol $\cdot$ indicates the vector dot product. Further, $\mathbf{w}$ and b satisfy

$$y_i(\mathbf{w} \cdot \mathbf{x} + b) > 0, \quad i = 1, 2, \cdots, N, \tag{6.2}$$

in order to find such a hyperplane.

With the above formulation, the SVM attempts to find an optimal hyperplane that maximizes the distance between the closest training sample and the separating hyperplane. Such a distance is often set to $1/\|\mathbf{w}\|$, hence, the margin between two classes becomes $2/\|\mathbf{w}\|$. The optimal hyperplane can then be found by solving the following convex quadratic programming (QP) problem

$$\operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{s.t.} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \cdots, N, \tag{6.3}$$

where 's.t.' stands for 'subject to'. The linearly constrained optimization problem in (6.3) is usually solved through its Lagrangian dual problem

$$\begin{gathered} \operatorname*{argmax}_{\alpha_i} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \right\}, \\ s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0, \quad i = 1, 2, \cdots, N, \end{gathered} \tag{6.4}$$

where $\alpha_i, i = 1, 2, \cdots, N$, is the Lagrange multiplier, and solved using QP methods. Accordingly, the discriminant function (6.1) becomes

$$f(\mathbf{x}) = \sum_{i \in \mathcal{S}} \alpha_i y_i (\mathbf{x_i} \cdot \mathbf{x}) + b, \tag{6.5}$$

where $\mathcal{S}$ is the subset of training samples with nonzero Lagrange multipliers, and these training samples are called *support vectors*.

The optimal case is, however, not always true for real world data. In the case that the two classes are not linearly separable, the concept of optimal separating hyperplane can be generalized to the solution of

$$\operatorname*{argmin}_{\mathbf{w}, \xi_i, b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right\}, \tag{6.6}$$

subject to

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, 2, \cdots, N, \tag{6.7}$$

$$\xi_i \geq 0, \quad i = 1, 2, \cdots, N, \tag{6.8}$$

where $\xi_i, i = 1, 2, \cdots, N$, are the slack variables, and C is the regularization parameter. The larger the C value is, the higher the penalty associated to the misclassified samples becomes. The same Lagrange formulation above can be applied to (6.6), and the Lagrange multipliers α_i are found by solving (6.4) with slightly different constraints

$$\begin{gathered} 0 \leq \alpha_i \leq C, \quad i = 1, \cdots, N, \\ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{gathered} \tag{6.9}$$

(2) Nonlinear SVM – Kernel Method: The linear discriminant functions in the previous case can be generalized to the category of nonlinear discriminant functions. The idea is to map the data to a higher dimensional space through a nonlinear transform, where a separation between two classes can be found in the transformed domain following the method of linear SVM. Given a nonlinear mapping function $\Phi(\cdot)$ that usually maps to a higher dimensional (Hilbert) space $\Phi : \mathbb{R}^M \mapsto \mathcal{H}$, the nonlinear SVM kernel method solves the dual problem as the one defined in (6.4) by replacing the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ with $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i)$. The kernel method allows us to define a kernel function K,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i \cdot \mathbf{x}_j), \tag{6.10}$$

such that the nonlinear SVM can be formulated using only the kernel function, without computing $\Phi(\mathbf{x})$ and the dot products in the transformed space explicitly. With the kernel method, the problem (6.4) becomes

$$\underset{\alpha_i}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \right\}, \tag{6.11}$$

under constraints (6.9). The discriminant function $f(x)$ is then expressed as

$$f(\mathbf{x}) = \sum_{i \in \mathcal{S}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b. \tag{6.12}$$

Different kernel functions lead to different SVMs. Some popular kernel functions are: linear — $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$, polynomial — $K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i \cdot \mathbf{x}_j) + 1)^d, d \in z^+$, or radial basis function (RBF) — $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2), \sigma \in \mathbb{R}^+$.

6.2.1.2 SVMs for Hyperspectral Image Classification

SVMs generally implement the idea of mapping the input vectors into a high dimensional feature space through some nonlinear functions, and construct an optimal hyperplane in this space [Vapnik 1999]. Unlike lower dimensional spaces such as 3D Euclidean space, it is difficult to establish the properties of high dimensional space, hence, the behaviour of data in it [Jimenez and Landgrebe 1998].

One of the characteristics of hyperspectral space, as discussed in [Jimenez and Landgrebe 1998; Melgani and Bruzzone 2004], is that the increase in dimensionality makes the hyperspace almost empty such that data has a tendency to concentrate close to the tails of the distribution. Thus, it is likely that the data is in the proximity of decision boundaries of information classes. SVM based approach, from the geometric point of view, finds a hyperplane that is close to the decision boundaries. This is indicative of the potentials of SVMs for hyperspectral image classification. Further, due to the geometrical nature of SVMs, the methodology does not rely on the statistical distributions of the information classes over the entire hyperspace. This is an appealing property of SVMs compared to the statistical classifiers. In the latter case, the number of required training samples grows exponentially with the increase of dimension, in order to obtain accurate density estimation. On the other hand, SVMs do not require density estimation. Hence, the approach of SVMs is less sensitive to to data with an increasing dimension. This often leads to a satisfactory classification performance of SVMs even in the case of small number of training data.

Hyperspectral data often consists of multiple classes, and thus the SVMs need to be extended for such cases. This is usually done by combining several binary classifiers. Two such popular strategies are *one against all* and *one against one* [Hsu and Lin 2002]. Assume that the number of distinct classes is T in a given data.

One against all: T binary classifiers, one for each class, are trained. The mth classifier is trained from all of the samples in the mth class with positive labels, and all the other samples with negative labels. The class of a test sample $\mathbf{x}$ is determined by the class whose discriminant function yields the largest value.

One against one: $T(T-1)/2$ binary classifiers are constructed where each one is trained on data from one pair of classes. The classification of a test sample $\mathbf{x}$ can be determined by a voting strategy: each binary classifier between classes i and j votes for the class of $\mathbf{x}$ to be either i or j; in the end, $\mathbf{x}$ belongs to the class whose number of votes is maximum.

6.2.2 Sparse Representation Based Classification Approach

Sparse representation has found a variety of applications in hyperspectral image processing. These include but are not limited to unmixing [Iordache et al. 2011 2014], classification [Chen et al. 2011a; Soltani-Farani et al. 2013; Sun et al. 2015; Zou et al. 2015], denoising [Zhao and Yang 2015; Lu et al. 2016], anomaly and target detection [Xu et al. 2016; Chen et al. 2011b]. In particular, the following approaches to the classification of hyperspectral image are the main concern of this thesis.

The basic sparsity model for hyperspectral image proposed by [Chen et al. 2011a] assumes that a test sample can be modelled as a linear combination of a few training samples in the same class. To exploit the spatial correlation across neighbouring pixels, the joint sparsity model is applied in [Chen et al. 2011a] to achieve a small neighbourhood of a pixel of interest sharing a common sparsity pattern. That is, hyperspectral image pixels in a small neighbourhood are approximated by a linear combination of a few common dictionary atoms. A generalized OMP algorithm, called simultaneous OMP (SOMP) [Chen et al. 2011a] is employed to solve the aforementioned joint sparsity model, i.e.,

$$\underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{X}-\mathbf{D}\mathbf{A}\|_{\mathrm{F}} \quad \text{s.t.} \quad \|\mathbf{A}\|_{\mathrm{row},0} \leq \mu, \tag{6.13}$$

where μ is the number of non-zero rows. Under this joint sparsity model, a dictionary is formulated by training samples from each class. Given a test pixel, we first find the corresponding sparse coefficient vector using the structured dictionary. The sparse coefficients corresponding to each class are then used to compute the residuals, i.e., reconstruction errors. The class of the test pixel is determined by the class which gives the minimum residual. A similar idea appears in [Li and Du 2014], where a nearest subspace classification method, coupled with the distance-weighted Tikhonov regularization [Tikhonov and Arsenin 1977], is developed. In this classifier, each test sample is approximated using the training samples from the same class independently. While this nearest subspace classifier relies on the spectral information only, it is enhanced in [Li et al. 2014] by exploiting the spatial characteristic of a test sample as well.

Structured dictionaries are also employed in Charles et al. [2011]; Soltani-Farani et al. [2013] for the classification of hyperspectral image data. The difference is that the dictionary in [Charles et al. 2011] is formed by a number of training samples from each class, and in [Soltani-Farani et al. 2013] the dictionary is trained using the training samples. The sparse coefficients are then used as feature vectors for training SVM classifiers. While the classification method in [Charles et al. 2011], termed SDL, is a simple demonstration of an application of sparse coefficients as discriminative features, Soltani-Farani et al. [2013] improved the SDL to incorporate the contextual information in the dictionary learning. In [Soltani-Farani et al. 2013], two approaches are taken to learn a dictionary that exploit both spectral and spatial correlations among the pixels for classification of hyperspectral image. One is to learn a contextual dictionary (CDL), and the other is to learn a spectral-contextual dictionary (SCDL). The CDL transforms the hyperspectral image pixels $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ into contextual representation $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_N$. The pixels in contextual domain are obtained for each pixel, taking the first moment, or the concatenation of first and second moment. The dictionaries are then trained in the contextual domain. The SCDL trains a structured dictionary on a contextual group pixels, where the contextual groups are non-overlapping image patches form each band. The sparse coefficients obtained from both CDL and SCDL are then used to train SVMs for the classification of hyperspectral image.

6.2.3 KNN Classification Approach

The kNN classification is a powerful non-parametric technique for pattern recognition [Cover and Hart 1967]. The basic assumption of this approach is that the pixels close to each other in a feature space tend to belong to the same class. As a non-parametric method, the kNN classifiers do not rely on the analysis of statistical distribution of the feature space. Instead, they simply employ a decision rule to assign a class to a test sample.

Given a set of N training data $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^M\}$, and labels $y_i \in \{\omega_1, \omega_2, \cdots, \omega_T\}$ for each of the data points, to classify a test sample $\mathbf{x}$, the kNN classifier first finds the k nearest neighbours of $\mathbf{x}$ in $\mathbf{X}$ according to a distance metric d. Among these k neighbours, we have $\sum_{i=1}^{T} k_i = k$, where k_i is the number of training samples from class ω_i. At this point, a majority voting rule

$$\text{Class}(\mathbf{x}) = \omega_i, \quad \text{where} \quad i = \underset{i}{\text{argmax}}\, k_i \tag{6.14}$$

can be applied to determine the class of $\mathbf{x}$.

Hyperspectral image classification concerns pixel classification, and the pixels spatially close to each other usually consist of the same or similar land cover materials. This is consistent with the basic assumption of the kNN classifiers. Another appealing factor of kNN based approaches is that they need very few parameter tuning, and thus reduce the requirement of large number of training samples, especially for high dimensional input feature spaces such as hyperspectral image. Variant kNN rule based classification approaches exist in the literature. To address the influence of the outliers in kNN classifiers, Mitani and Hamamoto [2006] explores the use of local mean vectors. Given a test sample, a local mean vector is computed for each class using its k nearest neighbours in the same class. The distances from the test sample to these class specific mean vectors are then computed to determine the membership of the test sample, which is the class whose mean vector is the closest.

Blanzieri and Melgani [2008] proposed a variant of kNN based on the SVM classification principle in order to further exploit the discriminative power in the k nearest neighbours. The main idea in [Blanzieri and Melgani 2008] is to partition the entire transformed feature space through a group of local maximal margin hyperplanes. This means that an SVM is built over the neighbourhood of a test sample, and the neighbourhood is found in the transformed feature space. This approach results in the combination of locality and search for a large margin separating hyperplane.

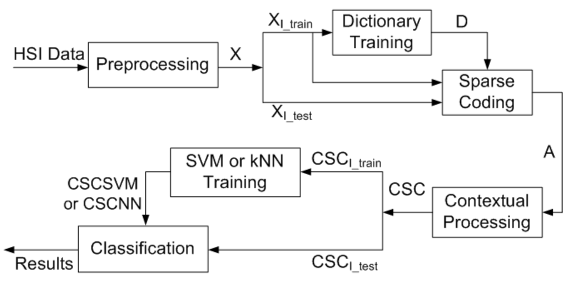


Figure 6.1: The diagram of the proposed classification framework. $\mathbf{X}$ – the HSI data after preprocessing; $\mathbf{D}$ – the dictionary; $\mathbf{A}$ – the matrix consists of the sparse coefficients; CSC – contextual sparse coefficients;

The kNN classifier is extended to a sparse representation based NN classifier (SRNN) [Zou et al. 2015]. SRNN first finds the sparse coefficient for a test sample using all the training samples. The k nearest neighbours are found as those training samples corresponding to the k largest components in the sparse coefficient. The majority voting rule in (6.14) is then applied to determine the class. Similarly, the local mean vector kNN in [Mitani and Hamamoto 2006] is extended by using the sparse coefficient for each class, where the average of the k largest components are used for the majority voting [Zou et al. 2015].

6.3 The Proposed Sparse Representation Based Classifiers

In this section, we discuss the details on applying sparse representation for the classification of hyperspectral data in our proposed methods. Figure 6.1 shows the diagram of the proposed framework. It consists of major components such as *preprocessing*, *dictionary training*, *sparse coding*, *contextual processing*, and *training the classifier*. The preprocessing involves removing the mean from, and the normalization of each spectral vector. That is, each spectral vector has zero mean and ℓ_2-norm of 1 after the preprocessing. We discuss the remaining components as follows.

6.3.1 Dictionary Training

Training a representative dictionary is crucial for the overall performance of the sparse representation based classification method. To harness the rich spectral information in hyperspectral image, we train dictionaries in the spectral domain and refer to it as *spectral dictionary* (SD). The dictionary learning problem in our approach is

$$\underset{\mathbf{D},\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1\right). \tag{6.15}$$

The optimization problem (6.15) is not convex as both $\mathbf{D}$ and $\boldsymbol{\alpha}$ vary. However, it is convex with respect to either one of the two variables, while the other one is fixed. This results in an iterative two-step strategy [Mairal et al. 2009; Aharon et al. 2006], namely *sparse coding* and *dictionary update*, being commonly applied to solve (6.15).

Dictionary update: The ODL algorithm [Mairal et al. 2010] is applied to update a dictionary in this thesis. ODL algorithm learns a dictionary through a number of iterations. During each iteration, one training sample (or a small group of samples) is drawn and its sparse decomposition is computed over the dictionary learned during the previous iteration. This is followed by a dictionary update step where a new dictionary is computed based upon the sparse decompositions obtained in previous iterations and the current dictionary. This learning process is a stochastic approximation approach in view of obtaining sufficient statistics by aggregating the past information during each iteration. The statistics are then utilized to minimize an expected cost function. This approach often yields faster convergence with a sufficiently good expected cost.

Sparse coding: Given a vector $\mathbf{x} \in \mathbb{R}^M$ and a dictionary $\mathbf{D} \in \mathbb{R}^{M\times K}$, sparse coding finds the coefficient vector $\boldsymbol{\alpha} \in \mathbb{R}^K$ — with á priori of sparsity, i.e., only a few of the components in $\boldsymbol{\alpha}$ are non-zero, so that $\mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha}$. Sparse coding is solved in a separable manner, where for each sample $\mathbf{x}$, its coefficient vector $\boldsymbol{\alpha}$ can be found independently of the other samples. The *Lasso*, or *basis pursuit* [Chen et al. 2001],

$$\underset{\boldsymbol{\alpha}_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1, \tag{6.16}$$

is an efficient approach to solve the sparse coding. The penalty term in (6.16) is an ℓ_1-norm. Instead of a penalty, a constraint, such as ℓ_0-norm, can be added to solve the problem of sparse coding, that is

$$\underset{\boldsymbol{\alpha}_i}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2, \quad \text{s.t.} \quad \|\boldsymbol{\alpha}_i\|_0 \le \mu, \tag{6.17}$$

where ℓ_0-norm is used to induce sparsity in the coefficient vectors, and μ is the maximum number of non-zero components in the sparse coefficients. A greedy algorithm such as matching pursuit [Mallat and Zhang 1993] or orthogonal matching pursuit (OMP) [Davis et al. 1997] solves (6.17) efficiently.

6.3.2 Hyperspectral Image Classification Using Contextual Sparse Coefficients

Given the trained dictionary $\mathbf{D}$, we find the sparse representation matrix $\mathbf{A}$ for $\mathbf{X}$ using OMP algorithm. This problem is set as (6.17), and the parameter μ indicates the maximum number of non-zero components in the coefficients. OMP is a greedy algorithm based on *matching pursuit* [Mallat and Zhang 1993] that refines the signal approximation with an iterative procedure. At each iteration, the current residual is decomposed by projecting the signal on each dictionary atom in $\mathbf{D}$, and selecting the atom that minimizes the residual. Once an atom is selected at each iteration, the signal is orthogonally projected to the span of the current selected atoms, and the residual is recomputed for the next iteration. The iteration stops when the number of non-zero coefficients satisfies the prescribed threshold, i.e., the value μ, or a specified maximum number of iterations is reached.

The primary reason we choose the sparse coding problem as (6.17) is that we have more flexibility in the sparsity of the coefficients. The sparsity level in the sparse coefficients has impact on the classification performance as demonstrated in Section 6.4.

In the training process of an SD, the spectral information in the hyperspectral image pixels is considered independently of each other. However, it is important to incorporate the spatial correlation among the pixels close to each other in order to improve the classification performance of hyperspectral image. This is due to the fact that neighbouring hyperspectral image pixels usually consist of similar materials. In our approach, we first trained a spectral dictionary without taking any spatial information into account. A contextual filter was then applied in the sparse coefficient domain to incorporate spatial information for the classification. Suppose $\alpha_{x,y}$ is a component in any coefficient vector $\boldsymbol{\alpha}$ corresponding to a dictionary atom, where x and y are the spatial coordinates. The contextual transform of $\alpha_{x,y}$, denoted by $c_{x,y}$, is computed as

$$c_{x,y} = \frac{1}{PQ} \sum_{p=-\lfloor \frac{P}{2} \rfloor}^{\lfloor \frac{P}{2} \rfloor} \sum_{q=-\lfloor \frac{Q}{2} \rfloor}^{\lfloor \frac{Q}{2} \rfloor} \omega_{x+p,y+q} \alpha_{x+p,y+q}, \tag{6.18}$$

where PQ is the predetermined filter (or window) size, and $\omega_{x+p,y+q}$ is the weight associated with the neighbour corresponding to the coordinates. In our experiment, we assigned the same weight of 1 to all the neighbours. After performing the contextual transform for all the sparse coefficient vectors, we obtain the *contextual sparse coefficient* (CSC) $\mathbf{c}_i$ corresponding to $\boldsymbol{\alpha}_i, i = 1, 2, \cdots, N$, respectively.

Compared to the original sparse coefficient vector $\boldsymbol{\alpha}$, the contextual transform yields smoothing effect within the local neighbourhood due to the nature of the applied filter. This smoothing effect can be interpreted as the influence of a particular dictionary atom has over a small neighbourhood of a hyperspectral image pixel in the original input domain, i.e., the domain before the sparse coding is applied. Recall that for $\boldsymbol{\alpha}_i \in \mathbb{R}^K$, where K is the number of atoms in the dictionary, each component in $\boldsymbol{\alpha}_i$ corresponds to an atom. Therefore, the contextual transform with a smoothing filter forces the hyperspectral image pixels in a small neighbourhood to adopt a similar sparsity pattern by having the same dictionary atoms to be activated in the sparse coefficient domain. However, the c will be less sparse than the $\boldsymbol{\alpha}$ in general, i.e. more nonzero coefficients in the average.

To evaluate the discriminant character of such contextual sparse coefficients for the classification of hyperspectral image, the CSCs are applied as feature vectors for training two classifiers, namely, CSCSVM and CSCNN.

In CSCSVM, the SVMs are trained on the training CSCs and their corresponding labels. We used the one-against-one strategy for multi-class classification using SVMs. Under such strategy, multiple SVMs are trained to model all possible pairwise classifications. Given a test sample, each SVM carries out a binary classification for one pair of classes, and the final decision is taken on the basis of "maximum voting" rule. The implementation LIBSVM in [Chang and Lin 2011] was used in our experiments.

The kNN classifier in CSCNN chooses the k nearest samples from all the training samples, and assigns a label to the test sample by a majority voting rule. The distance measure used in our experiment is the cosine similarity based spectral angle mapper (SAM),

$$\mathrm{dis}(\mathbf{x}, \mathbf{c}_i) = \mathrm{SAM}(\mathbf{x}, \mathbf{c}_i) = \arccos \frac{\mathbf{x}^T \mathbf{c}_i}{\|\mathbf{x}\|_2 \|\mathbf{c}_i\|_2}, \tag{6.19}$$

where $\mathbf{x}$ is a test sample, and $\mathbf{c}_i, i = 1, \ldots, L$ (L is the number of training samples).

6.4 Experiments

6.4.1 Data Set Description

Three data sets[1] are considered in our experiments: the Indian Pines data set acquired by the AVIRIS sensor, 200 bands with 145×145 pixels; the University of Pavia data set acquired by

[1]The data sets are freely available at:http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes.

Table 6.1: The Ground Truth Classes in AVIRIS Indian Pines and the Training and Test Sets for Each Class

Class		Samples	
No.	Name	Train	Test
1	Alfalfa	6	40
2	Corn-notill	144	1284
3	Corn-mintill	84	746
4	Corn	24	213
5	Grass-pasture	50	433
6	Grass-trees	75	655
7	Grass-pasture-mowed	3	25
8	Hay-windrowed	49	429
9	Oats	2	18
10	Soybean-notill	97	875
11	Soybean-mintill	247	2208
12	Soybean-clean	62	531
13	Wheat	22	183
14	Woods	130	1135
15	Buildings-Grass-Trees-Drives	38	348
16	Stone-Steel-Towers	10	83
	Total	1043	9206

Reflective Optics System Imaging Spectrometer (ROSIS-03) optical sensor, 103 bands with 610×340 pixels; and the AVIRIS Salinas data set, 204 bands with 512×217 pixels. Note that the number of bands indicated above is after the removal of some noisy channels.

The number of training samples chosen for Indian Pines data set was around 10% of the labelled samples from each class, and the remaining 90% was used as testing samples. The information about ground truth classes, and the number of training and testing samples are given in Table 6.1. For University of Pavia and Salinas data sets, around 5% of the labelled samples from each class was chosen for training, and the remaining as testing data. The relevant information regarding these two data sets are shown in Tables 6.2 and 6.3, respectively.

Table 6.2: The Ground Truth Classes in ROSIS University of Pavia and the Training and Test Sets for Each Class

Class		Samples	
No.	Name	Train	Test
1	Asphalt	274	6357
2	Meadows	270	18379
3	Gravel	196	1903
4	Trees	262	2802
5	Metal sheets	133	1212
6	Bare soil	266	4763
7	Bitumen	188	1142
8	Bricks	257	3425
9	Shadows	116	831
	Total	1962	40814

Table 6.3: The Ground Truth Classes in AVIRIS Salinas and the Training and Test Sets for Each Class

Class		Samples	
No.	Name	Train	Test
1	Broccoli-green-weeds_1	102	1907
2	Broccoli-green-weeds_2	186	2540
3	Fallow	98	1988
4	Fallow-rough-plow	76	1318
5	Fallow-smooth	134	2544
6	Stubble	198	3761
7	Celery	179	3400
8	Grapes-untrained	282	10989
9	Soil-Vinyard-develop	249	5954
10	Corn-senesced-weeds	164	3120
11	Lettuce-romaine-4wk	61	1007
12	Lettuce-romaine-5wk	104	1823
13	Lettuce-romaine-6wk	60	856
14	Lettuce-romaine-7wk	63	1007
15	Vinyard-untrained	257	7011
16	Vinyard-vertical-treils	116	1691
	Total	2329	51800

6.4.2 Experiment Design

In this section, the various classification strategies are compared. These include SVM with an RBF kernel (SVM), SVM applied to contextual data (CSVM) with an RBF kernel [Camps-Valls et al. 2006], joint sparsity model using SOMP [Chen et al. 2011a], CDL [Soltani-Farani et al. 2013], SCDL [Soltani-Farani et al. 2013], spectral dictionary learning for sparse coding (SDL) [Charles et al. 2011], and our approaches (CSCNN, CSCSVM). The classification performances were evaluated according to the accuracy for each class, the overall accuracy (OA) – the ratio of the number of accurately classified pixels to the total number of test pixels, the average accuracy (AA) – the average of the accuracies for each class, and the kappa (κ) statistic – the degree of agreement between different observers. These measures were computed using the confusion matrix. In particular, the κ is a measure of agreement or accuracy. The analysis is based on the difference between the actual agreement and the chance (or random) agreement. It indicates the extent to which the percentage correct values of a confusion matrix are due to true agreement versus chance agreement. For example, a κ value of 0.91 can be interpreted as the observed classification is 91% better than the one resulting from chance. All the classification results were obtained by taking the averages over 5 runs, and for each run, the training and testing samples were selected randomly.

The results for SOMP were obtained by setting the ℓ_p-norm for $p = \infty$, the sparsity level to 5, and the neighbourhood window size to 5×5. The setting of these parameters were guided by the work in [Chen et al. 2011a]. For SDL, the dictionary is trained by solving (6.15) with an ℓ_1 regularizer, and the sparse coding is done by (6.16). For the SVM with RBF kernel and CSVM with RBF kernel based methods, five-fold cross validations were carried out to select the parameters for RBF kernel within the search spaces of $\gamma \in \{2^{-10}, 2^{-9}, \ldots, 2^{3}\}$ and $C \in \{2^{-2}, 2^{-1}, \ldots, 2^{15}\}$. The CDL, SCDL, and CSCSVM results were produced by training the SVMs with linear kernels. The reason linear kernels could perform well is that due to the overcomplete dictionaries used in sparse representations, the sparse coefficients are usually in higher dimensions than the original pixel vectors. The parameter for linear kernels were selected within the search space of

Table 6.4: Classification Performance for Indian Pines Data Set using Various Classifiers

Class	SVM	CSVM	SOMP	CDL	SCDL	SDL	CSCNN[1]	CSCSVM[2]
1	60.00	92.00	81.50	85.50	94.00	32.50	89.03	**99.00**
2	75.81	96.45	88.08	89.70	94.67	65.70	91.82	**97.94**
3	71.39	95.74	82.33	86.27	94.34	52.31	92.54	**98.98**
4	56.71	92.96	72.30	85.63	95.02	33.71	96.17	**97.75**
5	90.48	96.12	94.73	95.10	96.91	87.07	95.75	**97.37**
6	93.95	97.98	96.70	97.37	**99.15**	93.86	95.17	98.72
7	73.60	93.60	88.80	88.00	**100.0**	57.60	63.21	97.60
8	95.80	99.77	98.83	99.39	**100.0**	96.92	99.91	99.95
9	38.89	70.00	62.22	43.33	87.78	17.78	56.67	**91.11**
10	70.35	93.37	90.15	85.05	95.20	57.83	89.93	**96.71**
11	81.65	97.19	94.69	92.63	97.36	73.15	94.34	**98.95**
12	74.01	95.25	80.83	81.51	89.72	61.88	94.73	**98.23**
13	96.39	98.58	98.36	97.92	99.13	95.41	**100.0**	99.34
14	93.81	98.93	98.45	97.80	**99.88**	92.70	97.27	99.10
15	55.98	96.95	80.40	88.79	96.84	53.28	93.20	**98.28**
16	93.25	90.36	**94.22**	92.29	93.25	90.36	80.38	92.77
OA	80.59	96.61	91.21	91.41	96.53	72.57	94.03	**98.41**
AA	76.38	94.08	87.66	87.89	95.83	66.38	89.38	**97.61**
κ	0.778	0.961	0.899	0.902	0.961	0.685	0.932	**0.982**

All the numerical results for the classification, excluding the ones for κ, are in percentage.
[1] Dictionary training parameter $\lambda = 0.00217$, sparse coding parameter $\mu = 10$.
[2] Dictionary training parameter $\lambda = 0.0117$, sparse coding parameter $\mu = 7$.

$C \in \{2^{-2}, 2^{-1}, \ldots, 2^{15}\}$ with five-fold cross validations.

The neighbourhood window size for methods CSVM, CDL, SCDL, CSCNN, and CSCSVM that involve contextual grouping was chosen by running cross validations among three options of 5×5, 7×7, and 9×9 pixels with the pixel in question is in the centre.

For approaches SDL, CDL, and SCDL, the parameters concerned in the dictionary training process were tuned as in [Charles et al. 2011; Soltani-Farani et al. 2013]. That is, the regularization parameter λ was chosen by running cross validations among the values $\{0.1, 1, 10, 100\}$. The dictionary size, i.e., the number of dictionary atoms, was selected from $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$, the fractions to the number of training samples for each class. For example, if the fraction is $\frac{1}{2}$, the dictionary size for the Indian Pines data set is at least 522, and for the Pavia University data set is at least 981. In our approaches, the dictionary size is set as $c \times r$, where c is the number of classes, and r is the number of atoms for each class. We choose r from the set $\{24, 32, 40\}$, and the regularization parameter λ from the range of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$. When a particular class has fewer training samples than the value of r, the number of dictionary atoms for that class is set to the number of training samples available. For all the sparse representation based approaches that need learning a dictionary, only the training samples were used to initialize or to learn the dictionaries, and subsequently, the testing samples were not seen at all during the dictionary training.

6.4.3 Experimental Results

6.4.3.1 Classification Results of Various Approaches

Table 6.4 shows the classification results of different approaches for Indian Pines data set. Both CSCNN and CSCSVM demonstrated competitive performance among the methods considered. In particular, CSCSVM produced similar or slightly better results compared to CSVM and SCDL, and consistent superior performance compared to the remaining approaches. Similar observations

Table 6.5: Classification Performance for University of Pavia Data Set using Various Classifiers

Class	SVM	CSVM	SOMP	CDL	SCDL	SDL	CSCNN	CSCSVM
1	83.84	96.34	70.59	91.85	96.12	88.92	98.05	**98.87**
2	87.02	98.87	93.14	96.54	99.02	84.57	**99.16**	98.33
3	75.49	93.08	94.67	86.73	96.91	76.00	93.09	**98.43**
4	96.55	98.79	96.65	97.15	**98.99**	95.83	96.34	98.44
5	99.62	99.97	**99.98**	99.97	99.93	99.74	99.10	99.95
6	84.35	98.78	90.74	95.48	**99.40**	87.67	83.24	99.29
7	90.56	99.19	99.37	96.97	99.40	82.54	98.13	**99.77**
8	85.68	94.12	93.24	82.66	96.22	69.68	84.36	**98.15**
9	98.58	98.80	86.11	98.46	**99.98**	100	99.84	98.75
OA	86.93	97.83	89.90	94.26	98.33	85.44	94.87	**98.62**
AA	89.08	97.55	91.61	93.98	98.44	87.22	94.59	**98.89**
κ	0.831	0.971	0.867	0.924	0.978	0.813	0.931	**0.982**

For both CSCNN and CSCSVM, dictionary training parameter $\lambda = 0.0117$, sparse coding parameter $\mu = 7$.

may also be made with the classification performances for University of Pavia and Salinas data sets, as shown in Tables 6.5 and 6.6, respectively. The values of parameters used to obtain the results in Tables 6.4, 6.5, and 6.6 are: i) the regularization parameter for dictionary training $\lambda = 0.0117$ for CSCSVM, and $\lambda = 0.00217$ for CSCNN; ii) the sizes of the dictionaries are restricted to the maximum of 32 atoms for each class; iii) the number of non-zero components in sparse coding $\mu = 7$.

Both SDL and CSCSVM take similar approaches in dictionary learning and sparse coding. The main difference is that SDL uses the sparse coefficients as the feature vectors for SVM, while CSCSVM incorporates the contextual information with the sparse coefficients before training the SVM classifiers. This leads to the significantly improved classification performance of CSCSVM over SDL for all three data sets used. Comparing among SOMP, CDL, SCDL, CSCSVM, and CSCNN, the difference is that SOMP, CDL and SCDL utilize the contextual information before the sparse coding stage or during the dictionary learning, while CSCSVM and CSCNN employ spectral information only for training the dictionary, and apply contextual information in the spectral sparse coefficients. The comparable classification performances we obtained for these methods are indicative that similar discriminative character can be achieved by adding the contextual information in the original hyperspectral image pixel domain or in the sparse coefficient domain. At this point, one may wonder what difference can these two approaches make. Let us examine the classification maps of different approaches, shown in Figure 6.2 (these are the classification maps for the first run for each method), especially the closely performing SCDL and CSCSVM. The classification errors for CSCSVM and CSCNN occur mostly along the neighbouring pixels from two different classes. However, the same claim can not be made for SCDL and other methods. This implies applying the contextual information in the (spectral) sparse coefficient domain could better enforce the neighbouring pixels to demonstrate similar features. Note that contextual grouping or joint sparsity strategies usually assume that the scenes for classification consist of homogeneous areas of the same materials or species. The classification errors among neighbouring pixels from different classes can be reduced by a better contextual grouping approach, such as segmentation.

Given the limited number of different classes to be identified in a scene, and the differing number of training samples, we aim at constructing the dictionaries by selecting the same or sim-

Table 6.6: Classification Performance for Salinas Data Set using Various Classifiers

Class	SVM	CSVM	SOMP	CDL	SCDL	SDL	CSCNN	CSCSVM
1	99.08	99.90	**100.0**	99.50	99.90	99.87	99.99	**100.0**
2	99.44	99.85	99.96	99.08	99.94	99.58	**100.0**	**100.0**
3	97.96	99.24	99.67	99.38	99.08	94.69	99.97	**100.0**
4	98.85	97.89	98.09	96.71	99.20	**99.30**	97.49	99.23
5	98.48	99.70	98.24	99.50	98.48	99.26	99.03	**99.84**
6	99.74	99.96	99.95	99.96	99.91	99.79	99.78	**100.0**
7	99.65	99.73	99.89	98.93	99.86	99.72	99.98	**100.0**
8	83.32	95.94	85.77	91.79	97.73	79.30	**99.03**	97.49
9	99.54	99.92	99.84	99.93	99.31	99.75	98.79	**100.0**
10	95.62	98.75	97.59	97.85	97.96	96.10	99.34	**99.92**
11	97.36	99.88	99.90	98.71	97.74	97.56	99.84	**100.0**
12	99.79	99.96	99.28	99.99	99.61	99.96	99.77	**100.0**
13	99.32	99.58	98.86	99.25	98.29	99.21	98.51	**100.0**
14	97.66	99.68	99.42	98.05	97.80	96.19	98.56	**99.94**
15	72.26	96.55	87.79	94.81	**98.31**	69.58	95.45	97.79
16	98.91	99.81	99.57	99.68	99.30	98.98	**100.0**	**100.0**
OA	91.95	98.43	94.93	97.05	98.78	90.75	98.78	**99.13**
AA	96.06	99.15	97.74	98.32	98.90	95.55	99.10	**99.64**
κ	0.910	0.983	0.944	0.967	0.986	0.897	0.986	**0.990**

For both CSCNN and CSCSVM, dictionary training parameter $\lambda = 0.0117$, sparse coding parameter $\mu = 7$.

ilar number of atoms from each class. This is not the case in [Chen et al. 2011a], where the dictionaries are constructed by using all the training samples. The advantage of our approach lies in a better control of the size of the dictionary, and satisfactory classification results could still be achieved using smaller dictionaries. Moreover, the imbalance among classes with considerable difference in the number of available training samples (e.g., the 'oat' and 'soybean_mintill') could be adjusted to some degree. While we prefer the dictionary to be overcomplete in sparse representations, a dictionary with smaller size means a smaller dimension in the resulting sparse coefficients, and subsequently, a reduced computational complexity.

6.4.3.2 Classification Performance on Varying Parameters for the Proposed Methods

In order to observe the effect of the regularization parameter λ on the classification performance, we ran experiments using various values for λ during the dictionary training. The classification performances using these dictionaries in CSCSVM and CSCNN are reported in Figures 6.3 and 6.4, where the results are for dictionary sizes of $K = 307$ and $K = 387$, respectively. Differences in the order of 10^{-3} are most common for all three criteria considered, and the maximum differences are limited by 10^{-2}. When the value of λ is fixed, the changes in the performances with respect to μ are similar for the OA, AA and κ, with some exceptions. For example, in Figure 6.3, when $\mu = 11$, for both dictionary sizes considered, the values for both OA and κ increased from those at $\mu = 7$, but this is not the case for AA. The reason is that although the performances for the majority of classes were consistent, they were not for a few of them (specifically, the class 'oat' for Indian Pines data set). It is also notable that for both CSCNN and CSCSVM, the performances are the same for $\lambda = 1.17, 1.17 \times 10^1$, irrespective of the dictionary sizes. Overall, the performance at $\lambda = 1.17 \times 10^{-2}$ for $K = 387$ showed more consistency compared to the others. Further, Figures 6.3 and 6.4 show that CSCSVM achieves approximately 3% higher OA than CSCNN does. However, the performance criteria AA for CSCNN is far worse than that

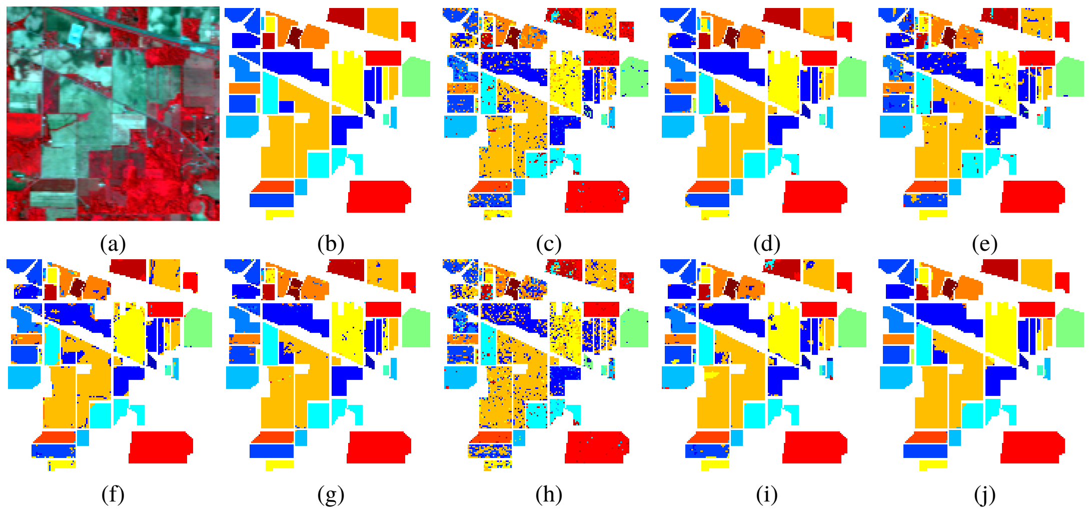


Figure 6.2: Indian Pines images: (a) False-colour image of original scene; (b) Ground truth. Classification maps: (c) SVM; (d) CSVM; (e) SOMP; (f) CDL; (g) SCDL; (h) SDL; (i) CSCNN; (j) CSCSVM. (The figure is best viewed in colour.)

of CSCSVM. This is primarily due to the very poor classification accuracies of CSCNN on two classes, namely, ‘Grass-pasture-mowed’ and ‘Oats’, which have fewer training samples.

As mentioned earlier, the discrepancy between OA and AA is mainly due to the poor class accuracies of a few classes. This is made clear in Figure 6.5, where we illustrate the class accuracies for 6 of the 16 labelled classes available in the Indian Pines data set (see Table 6.1 on Pg. 103 for the ground truth). Among the selected classes, ‘Grass-pasture-mowed’ and ‘Oats’ have relatively smaller number of labelled samples with 28 and 20, respectively. On the other hand, ‘Soybean-mintill‘ has the most number of labelled samples. The remaining three classes are ‘Alfalfa’, ‘Corn-notill’ and ‘Stone-Steel-Towers’. Among the selected classes, ‘Stone-Steel-Towers’ is the only non-vegetation or non-plant class. Figure 6.5 shows that the class accuracies for ‘Oats’ and ‘Stone-Steel-Towers’ are significantly worse than the others, and very sensitive to the density of the sparse coefficients. The class accuracy for ‘Grass-pasture-mowed’ is better than the two previous ones, and less sensitive to the differing values of μ. All other classes including the ones not shown here were classified with a similar accuracies above 97%, and robust to varying μs.

One may attribute the reason for poor classification accuracies for the class ‘Oats’ to the a smaller number of training samples available. Comparing it with the class ‘Grass-pasture-mowed’, both of them have similar number of training samples. However, the accuracies for the class ‘Grass-pasture-mowed’ are higher than those of the class ‘Oats‘. This means besides the lack of training samples, it is possible that there are some other factors which cause an inaccurate classification of the class ‘Oats’. One of such factors could be a strong spectral similarity between two or more types of land covers. These are within the same category or across different categories. For example, in the case of the class ‘Stone-Steel-Towers’, the spectral character of this class might be difficult to be distinguished from one of the vegetation classes in the Indian Pines data set to another. Further investigation needs to be done in order to gain clarification.

The comparison of results for CSCSVM using different sizes of dictionaries, together with using varying number of non-zero components, is presented in Figure 6.6. When we increase the dictionary size from 24 atoms for each class to 40 atoms, all the performance criteria considered yield slight improvements in the order of 10^{-3}. We also observe from both Figure 6.3 and Figure 6.6 that for a given dictionary, when the number of non-zero components μ increases, the OA and κ increase for the lower range of μ in general. However, the growth rate is higher for smaller μs than for larger ones. The growth range is approximately within the number of atoms per class. That is, if the dictionary size is limited to 32 atoms per class, the growth range is within approximately 32. Recall that the number of non-zero components indicates the density in the sparse coefficients. When the number of non-zero components increases, the reconstruction error decreases, as it is the optimization objective in our dictionary learning problem. However, the discriminative character of the sparse coefficients does not necessarily increase linearly with respect to the increasing density in the sparse coefficients. For example, in Figure 6.6, the minimum OA is 98.47% at $\mu = 3$, and the maximum OA is 98.89% at $\mu = 31$. There is an increase of only 0.42% for OA when the density is increased by 28. This is because, as more dictionary atoms are activated in the sparse coding, some of the activated dictionary atoms might introduce noise which reduces the ability of the sparse coefficients to distinguishes between different classes.

Table 6.7: The Running Time for Dictionary Training using Various λs.

λ	0.00117	0.00217	0.0117	0.117	1.17	11.7
Time (s)	151.9605	98.8643	35.2542	12.5843	6.7474	5.8157

A main advantage of online dictionary learning algorithm is its ability to converge rapidly while minimizing the expected cost. However, dictionary training is usually a time consuming process even if the algorithm is optimized. Table 6.7 shows the empirical time for training dictionaries for Indian Pines data set using different values for λ, while the number of iterations is limited to 500, and dictionary size is 32 atoms for each class. A general guideline we followed in our experiments is that the higher the value λ is, the faster the dictionary converges. Moreover, the classification performance in terms of OA increases in general, when the value of λ increases within a certain range.

6.5 Conclusion

Despite the advances in hyperspectral image classification techniques, challenges, such as high dimensionality and limited number of labelled samples, still exist. This chapter investigates hyperspectral image classification based on sparse representations. The main idea is to build discriminative feature vectors in the sparse coefficient domain for hyperspectral image classification through sparsity model. Under the proposed strategy, a hyperspectral image is modelled using a learned dictionary via sparse coding. Only the spectral information is considered during the dictionary learning stage. The spatial correlation is incorporated into the sparse coefficients through contextual neighbourhood grouping and local mean filter. The contextual sparse coefficients are then used to train two basic classifiers SVM and kNN. The effectiveness of such an approach is demonstrated by the experimental results on a number of hyperspectral images. In particular, the SVM with linear kernels perform very well because the sparse representation is able to transform the data into linearly separable space. This shows that other linear classifiers, such as linear discriminant analysis (LDA), can also perform well using our classification approach. Furthermore, the results also show that our approach offers a certain degree of flexibility in obtaining satisfactory classification performance through varying sizes of dictionaries and varying number of nonzero components in the sparse coefficients. This is advantageous in the case of limited number of labelled samples. As future work, we will investigate more accurate approaches for contextual grouping of neighbourhood pixels, and will apply sparse representations to semi-supervised classification of hyperspectral images.

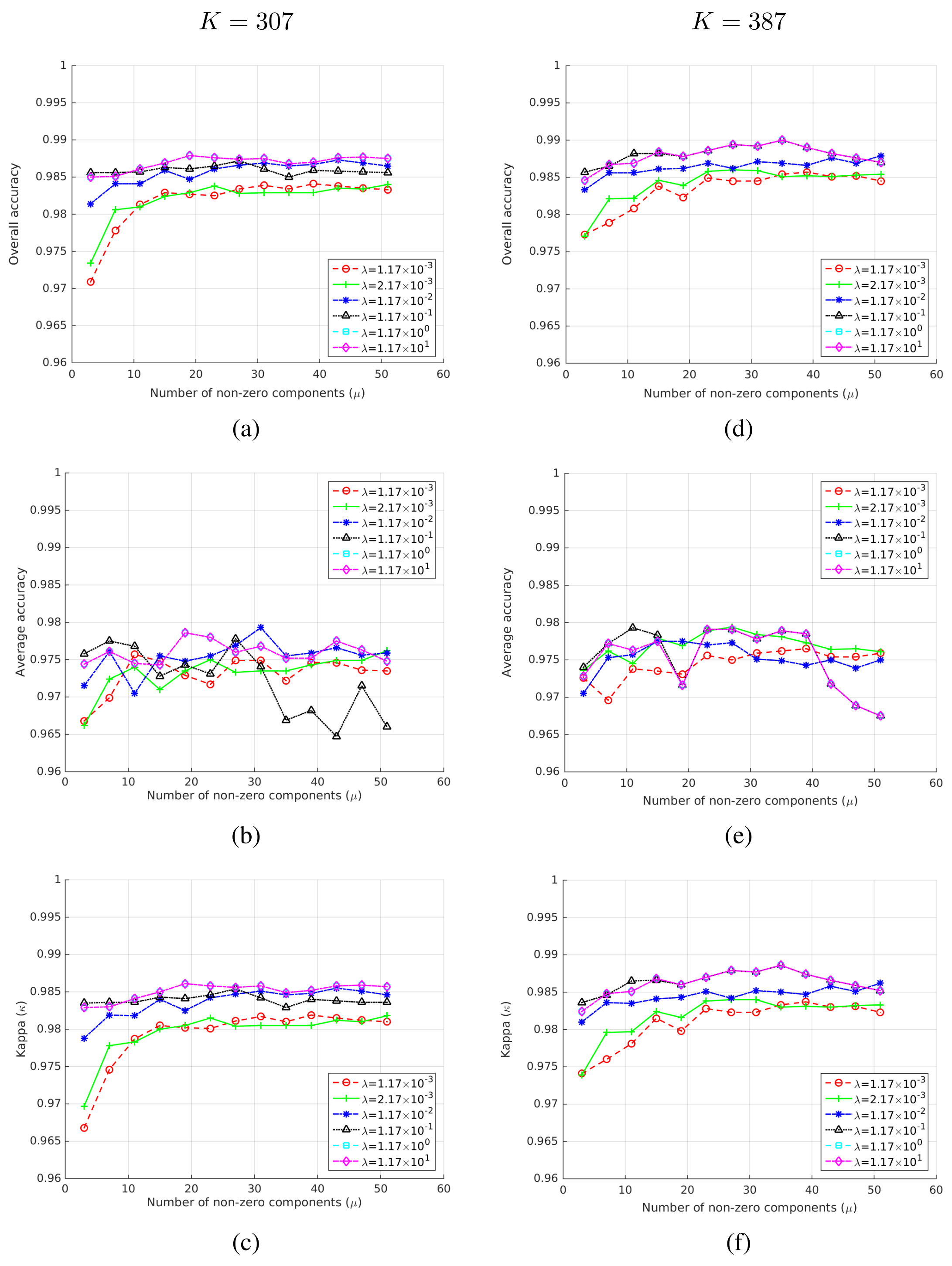


Figure 6.3: The classification performance of CSCSVM for Indian Pines data set using various regularization parameter λs. The dictionary size $K = 307$ for the first column, and $K = 387$ for the second column. (a), (d) The overall accuracy; (b), (e) The average accuracy; (c), (f) The κ.

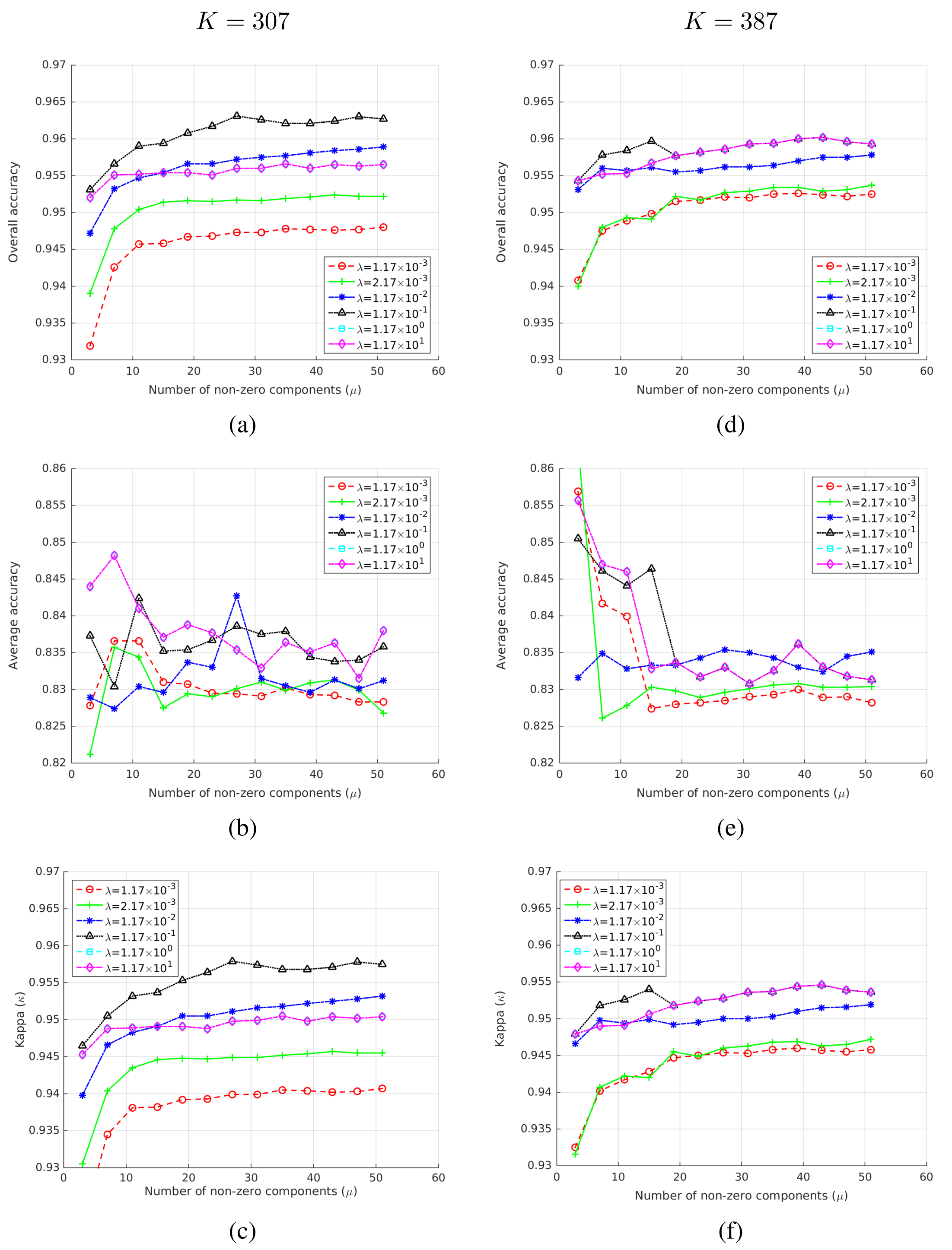


Figure 6.4: The classification performance of CSCNN for Indian Pines data set using various regularization parameter λs. The dictionary size $K = 307$ for the first column, and $K = 387$ for the second column. (a), (d) The overall accuracy; (b), (e) The average accuracy; (c), (f) The κ.

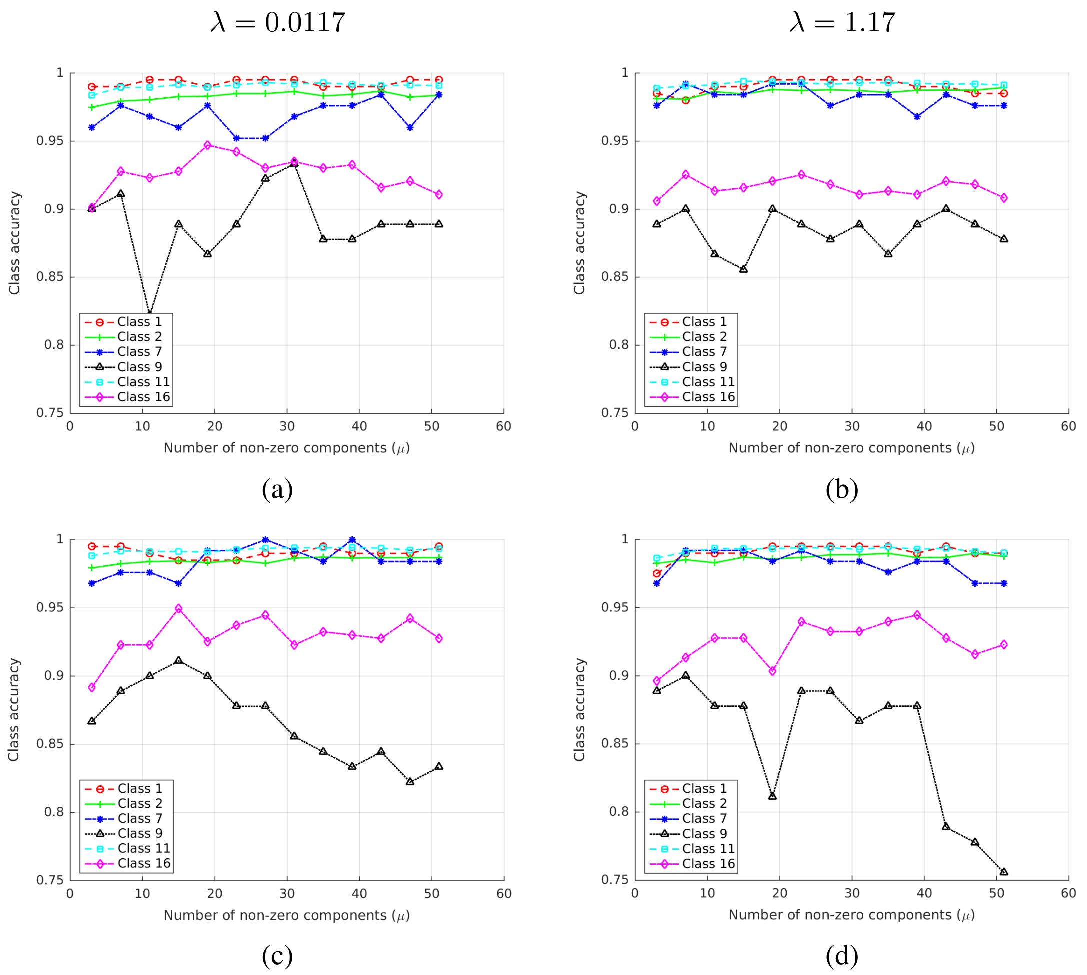
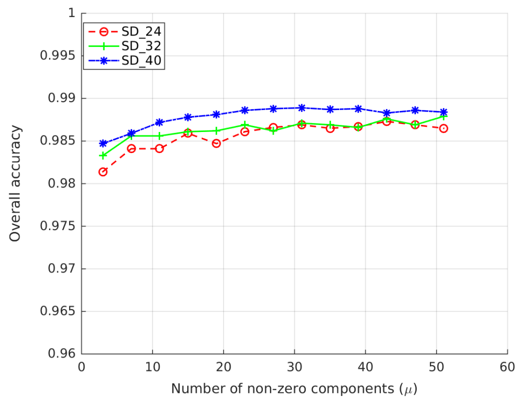
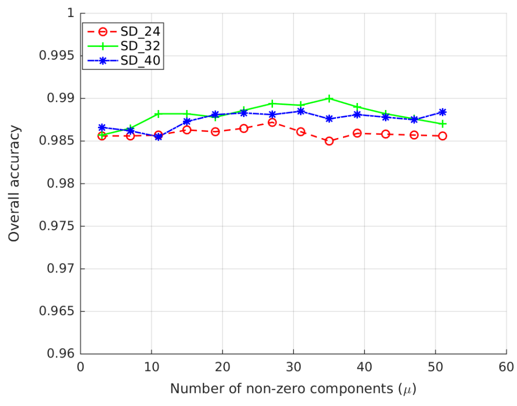


Figure 6.5: The selected class accuracies of CSCSVM for Indian Pines data set using different regularization parameters. The class number and name pairs are: (1, 'Alfalfa'); (2, 'Corn-notill'); (7, 'Grass-pasture-mowed'); (9, 'Oats'); (11, 'Soybean-mintill'); (16, 'Stone-Steel-Towers'). (a) $K = 307$, $\lambda = 1.17 \times 10^{-2}$; (b)$K = 307$, $\lambda = 1.17$. (c) $K = 387$, $\lambda = 1.17 \times 10^{-2}$; (d)$K = 387$, $\lambda = 1.17$.

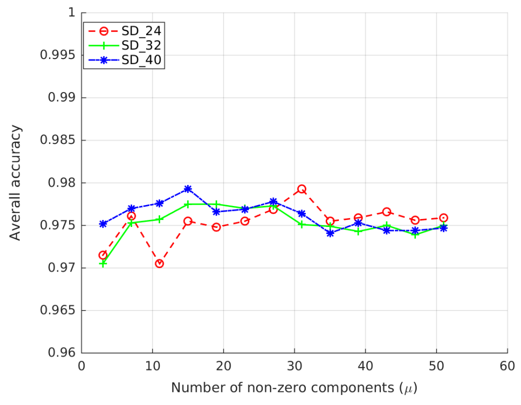
$\lambda = 0.0117$ $\lambda = 0.117$



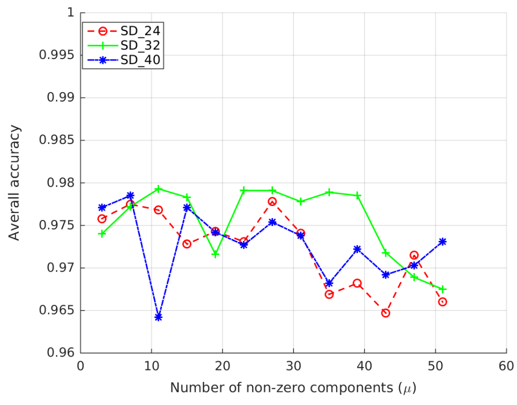
(a)



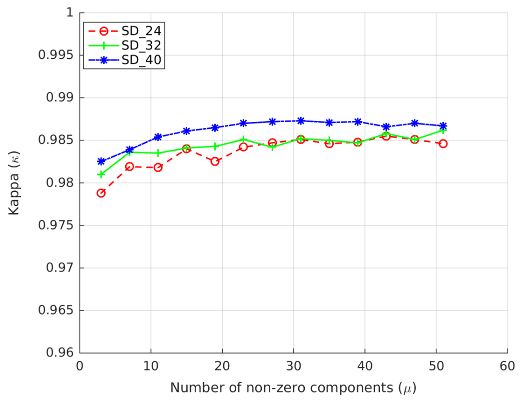
(d)



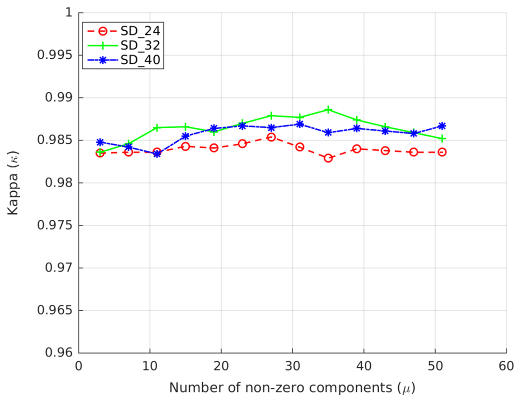
(b)



(e)



(c)



(f)

Figure 6.6: The classification performance of CSCSVM for Indian Pines data set using various dictionary sizes. SD_24, SD_32, SD_40 are dictionaries with 24, 32, and 40 atoms for each class, respectively. The regularization parameter $\lambda = 0.0117$ for the first column, and $\lambda = 0.117$ for the second column. (a), (d) The overall accuracy; (b), (e) The average accuracy; (c), (f) The κ.

Chapter 7
Conclusion

7.1 Summary

In this thesis we set out to explore incorporating sparse representation with hyperspectral image compression. The goal is to find that such an approach is effective, to extend the mechanism to hyperspectral image classification, and ultimately, to various hyperspectral image processing tasks. The key idea is that sparse representation should be able to discover the latent structures in hyperspectral images through dictionary learning and sparse coding. Surrounding this main theme, our work is carried out in two aspects.

The first aspect is developing a lossy hypespectral image compression framework that is based on sparse representation. Two approaches are taken for dictionary learning. The first one is learning a dictionary in the spectral domain of a hyperspectral image, and the second is in the multi-scale spectral domain. Given the learned dictionary, the sparse coefficients of a hyperspectral image are found by sparse coding. The compression scheme using spectral dictionaries is found to be very effective in terms of exploiting the spectral correlations in the hyperspectral images. It turns out that this approach can almost achieve similar performance as the JPEG 2000 multi-component coupled with DWT. The difference is that the former exploits the spectral redundancy only, while the latter exploits both spatial and spectral redundancies. The second approach explores learning a dictionary in the wavelet domain. This method is able to reduce the spectral redundancy, as well as the spatial one. However, both approaches suffer from the overheads of dictionary learning and encoding of the dictionary information. The overheads can be reduced by utilizing a pre-learned dictionary for training a new one. This method is effective if the base dictionary is chosen properly. In the case of hyperspectral images, there are often images available, obtained from a similar location, and thus a base dictionary can be learned from previously obtained data.

The second aspect is applying the sparse coefficients for hyperspectral image classification. The discriminative feature of the sparse coefficients is enhanced by incorporating contextual information. Simple classifiers such as SVMs and kNN trained on these contextual sparse coefficients lead to excellent classification performance. As the sparse coefficients in the lossy compression and classification are the same, it provides an alternate way of avoiding the dictionary encoding in the compression framework. However, this is only feasible if the complete reconstruction is not necessary.

7.2 Future Work

The methods presented in this thesis allow a few avenues for future research. They are described below either as improvements to the current work, or further questions.

- An imminent future work is to incorporate a way of effectively exploiting the spatial correlations with the current SRSD approach. A viable solution is to consider the contextual information in the learning of a spectral dictionary.
- The dictionary learning approach in the current lossy compression framework is primarily data compression oriented, and thus, there is a lack of mechanisms for information preserving. That is, reducing the data volume might sacrifice some vital information in hyperspectral images, such as the sub-pixel information in anomaly or target detection. Thus, it is desirable to have a dictionary learning approach that could minimize not only the reconstruction error, but also the loss of information. The related research question in this aspect becomes how to achieve a good balance in information preserving and data compression using sparse representation.
- An efficient coding of the sparse coefficients will improve the compression performance in the current work. The investigation of extending the bit plane coding to the sparse coefficients is a possible way to achieve this goal.
- The hyperspectral image classification approach in this thesis demonstrates the effectiveness of supervised classification based on sparse representation. A desired addition to the current work is a solution to the problem of unbalanced number of available training samples for each class at the dictionary training stage. In addition, owing to the lack of labelled data, unsupervised or semi-supervised classification methods are necessary. The extension of sparse representation to the semi-supervised or unsupervised hyperspectral image classification is another research area to pursue.
- Parallel processing is always an integral part in hyperspectral image processing, including compression. In the context of sparse representation based approaches, parallel processing could reduce the computational demand of dictionary learning and sparse coding. Accelerator based parallel computing for sparse representation is a future work that we intend to focus on.

7.3 Conclusion

Sparse representation, or sparse coding with an overcomplete dictionary, provides a strategy that resembles the one employed by the receptive field of simple cells in mammalian primary visual cortex [Olshausen and Field 1997]. Dictionary learning algorithms are able to discover the underlying structure of input signals or images by attempting to find the sparse coefficients for them. The resulting coefficients are not only sparse, they also possess a higher degree of statistical independence that leads to more efficient representations.

This thesis has thus focused on the ideas of applying sparse representation to hyperspectral image compression and classification. We have developed a lossy hyperspectral image compression framework that is based on sparse representation. We have also explored designing of hyperspectral image classifiers using dictionary learning and sparse coding. The main theme has been exploiting the rich spatial and spectral information in hyperspectral images for data compression and further data analysis. To this, the theory and methods of sparse representation turn out to be direct and effective. It is our hope that future work will generalize or expand the work in this thesis to the processing and analysis of broader types of signals and images.

Bibliography

Adams, M. D. and Kossentni, F. (2000). Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis. *IEEE Transactions on Image Processing*, 9(6):1010–1024.

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Sig. Proc.*, 54(11):4311 – 4322.

Akhtar, N., Shafait, F., and Mian, A. (2014). Sparse spatio-spectral representation for hyperspectral image super-resolution. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision - ECCV 2014. Lecture Notes in Computer Science*, volume 8695, pages 63–78. Springer, Cham.

Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn.*, 4(1):1–106.

Bachmann, C. M., Donato, T. F., Lamela, G. M., Rhea, W. J., Bettenhausen, M. H., Fusina, R. A., Bois, K. R. D., Porter, J. H., and Truitt, B. R. (2002). Automatic classification of land cover on smith island, VA, using HyMAP imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 40(10):2313–2330.

Bahrampour, S., Nasrabadi, M., Ray, A., and Jenkins, W. (2016). Multimodal task-driven dictionary learning for image classification. *IEEE Transactions on Image ProcessingI*, 25(1):24–38.

Baizert, P., Pickering, M., and Ryan, M. (2001). Compression of hyperspectral data by spatial/spectral discrete cosine transform. In *IGARSS 2001 - IEEE Geoscience and Remote Sensing Symposium*, volume 4, pages 1859–1861.

Beerten, J., Blanes, I., and Serra-Sagristà, J. (2015). A fully embedded two-stage coder for hyperspectral near-lossless compression. *IEEE Geoscience and Remote Sensing Letters*.

Benediktsson, J. A., Palmason, J. A., and Sveinsson, J. R. (2005). Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):480–491.

Benediktsson, J. A., Pesaresi, M., and Amason, K. (2003). Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):1940–1949.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, 2nd edition.

Bioucas-Dias, J. M. and Nascimento, J. (2012). Hyperspectral unmixing based on mixtures of dirichlet components. *IEEE Trans. Geosci. Remote Sensing*, 50(3):863–878.

Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., and Chanussot, J. (2012). Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379.

Blanes, I., Magli, E., and Serra-Sagrista, J. (2014). A tutorial on image compression for optical space imaging systems. *IEEE Geoscience and Remote Sensing Magazine*, 2(3):8–26.

Blanzieri, E. and Melgani, F. (2008). Nearest neighbor classification of remote sensing images with the maximal margin principle. *IEEE Transactions on Geoscience and Remote Sensing*, 46(6):1804–1811.

Boardman, J. W., Kruse, A., and Green, R. O. (1995). Mapping target signatures via partial unmixing of AVIRIS data. In *Proc. JPL, Airborne Earth Science Workshop*, pages 23–26.

Borengasser, M., Hungate, W. S. ., and Watkins, R. (2007). *Hyperspectral Remote Sensing Principles and Applications*. CRC Press.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems*, 20:161–168.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Broadwater, J., Banerjee, A., and Berlina, P. (2011). Kernel methods for unmixing hyperspectral imagery. In Prasad, L. B. S. and Chanussod, E. J., editors, *Optical Remote Sensing Advances in Signal Processing and Exploitation*, pages 247–269. New York: Springer.

Bryt, O. and Elad, M. (2008). Compression of facial images using K-SVD algorithm. *Journal of Visual Communication & Image Representation*, 19:270 – 282.

Camps-Valls, G., Gomez-Chova, L., Muñoz-Marí, J., Vila-Francís, J., and Calpe-Maravilla, J. (2006). Composite kernels for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 3(1):93–97.

Camps-Valls, G., Tuia, D., Gomez-Chova, L., Jiménez, S., and Malo, J. (2012). *Remote Sensing Image Processing*. Morgan & Claypool Publishers.

CCSDS (2012). The consultative committee for space data systems, lossless multispectral & hyperspectral image compression. https://public.ccsds.org/Pubs/123x0b1ec1.pdf. Accessed 2017-10-2.

CCSDS (2017a). The consultative committee for space data systems. https://public.ccsds.org/default.aspx. Accessed 2017-10-3.

CCSDS (2017b). The consultative committee for space data systems, image data compression, recommended standard, issue 2. https://public.ccsds.org/Publications/BlueBooks.aspx. Accessed 2017-10-3.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

Chang, C.-I. (2007a). Hyperspectral imaging systems. In Chang, C.-I., editor, *Hyperspectral Data Exploitation: Theory and Application*, pages 19–59. John Wiley & Sons, Inc.

Chang, C.-I. (2007b). Overview. In Chang, C.-I., editor, *Hyperspectral Data Exploitation Theory and Applications*, chapter 1. Johan Wiley & Sons, Inc.

Chang, C.-I. (2013). *Hyperspectral Data Processing: algorithm design and analysis*. John Wiley & Sons, Inc., Hoboken, NJ.

Charles, A., Olshausen, B., and Rozell, C. (2011). Learning sparse codes for hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):963 – 978.

Chen, G. and Qian, S. E. (2011). Denoising of hyperspectral imagery using principal component analysis and wavelet shrinkage. *IEEE Transactions on Geoscience and Remote Sensing*, 49(3):973–980.

Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129 – 159.

Chen, Y., Nasrabadi, N. M., and Tran, T. D. (2011a). Hyperspectral image classification using dictionary-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 49(10):3973–3985.

Chen, Y., Nasrabadi, N. M., and Tran, T. D. (2011b). Sparse representation for target detection in hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):629–640.

Christopoulos, C., Skodras, A., and Ebrahimi, T. (2000). The JPEG 2000 still image coding system: An overview. *IEEE Trans. on Consum. Electron.*, 46(4):1103–1127.

Clark, R. N. (1999). Spectroscopy of rocks and minerals, and principles of spectroscopy. In Rencz, A. N., editor, *Manual of Remote Sensing, Volume 3, Remote Sensing for Earth Sciences*, pages 3–58. John Wiley & Sons, Inc.

Corson, M. R., Korwan, D. R., Lucke, R. L., Snyder, W. A., and Davis, C. O. (2008). The hyperspectral imager for the coastal ocean (HICO) on the international space station. In *IGARSS 2008 - IEEE International Geoscience and Remote Sensing Symposium*, volume IV, pages 101–104.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27.

Craig, M. D. (1994). Minimum-volume transforms for remotely sensed data. *IEEE Trans. Geosci. Remote Sensing*, 32:542–552.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Commun. Pure Appl. Math.*, 41:909–996.

Daubechies, I. and Sweldens, W. (1998). Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, 4(3):247–269.

Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximation. *Constructive Approximation*, 13:57–98.

Do, M. N. and Vetterli, M. (2005). The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12):2091–2106.

Dobigeon, N., Moussaoui, S., Coulon, M., Tourneret, J. Y., and Hero, A. O. (2009). Joint bayesian endmember extraction and linear unmixing for hyperspectral imagery. *IEEE Trans. Sig. Proc.*, 57(11):4355–4368.

Donoho, D. L. (1999). Wedgelets: Nearly minimax estimation of edges. *The Annals of Statistics*, 27(3):859–897.

Donoho, D. L. and Johnstone, J. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Dragotti, P., Poggi, G., and Ragozini, A. (2000). Compression of multispectral images by three-dimensional SPIHT algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 38(1):416–428.

Du, Q. and Chang, C. I. (2004). Linear mixture analysis-based compression for hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 42(4):875 – 891.

Du, Q. and Fowler, E. (2007). Hyperspectral image compression using JPEG2000 and principle componenet analysis. *IEEE Geoscience and Remote Sensing Letters*, 4(2):201–205.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, 32(2):407–499.

Eismann, M. T., Stocker, A. D., and Nasrabadi, N. M. (2009). Automated hyperspectral cueing for civilian search and rescue. *Proceedings of the IEEE*, 97(6):1031–1055.

Elad, M. (2010). *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745.

Elad, M., Figueiredo, M., and Ma, Y. (2010). On the role of sparse and redundant representations in image processing. *Special Issue on Applications of Sparse Representation and Cpmressive Sensing, Proceesings of the IEEE*, 98(6):972 – 982.

Elmasry, G., Kamruzzaman, M., Sun, D.-W., and Allen, P. (2012). Principles and applications of hyperspectral imaging in quality evaluation of agro-food products: A review. *Critical Reviews in Food Science and Nutrition*, 52(11):999–1023.

Engan, K., Aase, S., and Husøy, J. (1999). Method of optimal directions for frame design. In *Proc. ICASSP '99 (Phoenix, USA)*, pages 2443–2446.

Engan, K., Skretting, K., and Husøy, J. (2007). Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation. *Digital Signal Processing*, 17(1):32 – 49.

Farrand, W. H., Singer, R. B., and Merényi, E. (1994). Retrieval of apparent surface reflectance from AVIRIS data: A comparison of empirical line, radiative transfer, and spectral mixture methods. *Remote Sensing of Environment*, 47(3):311–321.

Fauvel, M., Benediktsson, J. A., Chanussot, J., and Sveinsson, J. R. (2008). Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3804–3814.

Fauvel, M., Tarabalka, Y., Benediktsson, J. A., Chanussot, J., and Tilton, J. C. (2013). Advances in spectral — spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675.

Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am. A*, 4(12):2379–2394.

Fowler, J. (2000). Qccpack: an open-source software library for quantization, compression, and coding. In *Proc. SPIE*, volume 4115, pages 294–301.

Fowler, J. and Rucker, J. (2007). 3D wavelet-based compression of hyperspectral imagery. In Chang, C. I., editor, *Hyperspectral Data Exploitation: Theory and Applications*, pages 379–407. John Wiley & Sons, Inc.

Fu, W. J. (1998). Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416.

Gall, D. L. and Tabatabai, A. (1988). Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In *ICASSP-88, International Conference on Acoustics, Speech, and Signal Processing*, pages 761–764 vol.2.

Gao, L., Du, Q., Zhang, B., Yang, W., and Wu, Y. (2013). A comparative study on linear regression-based noise estimation for hyperspectral imagery. *IEEE Journal of Selected Topics* *in Applied Earth Observations and Remote Sensing*, 6(2):488–498.

García-Vilchez, F. and Serra-Sagristà, J. (2009). Extending the CCSDS recommendation for image data compression for remote sensing scenarios. *IEEE Trans. Geosci. Remote Sensing*, 47:3431–3445.

Geladi, P. L. M., Grahn, H. F., and Burger, J. E. (2007). Multivariate images, hyperspectral imaging: Background and equipment. In Grahn, H. F. and Geladi, P., editors, *Techniques and Applications of Hyperspectral Image Analysis*. John Wiley & Sons, Ltd.

Gersho, A. and Gray, R. M. (1991). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwell, MA, USA.

Goetz, A. F. H. (2009). Three decades of hyperspectral remote sensing of the earth : A personal view. *Remote Sensing of Environment*, 113:S5–S16.

Goetz, A. F. H., Vane, G., Solomon, J. E., and Rock, B. N. (1985). Imaging spectrometry for earth remote sensing. *Science*, 228(4704):1147–1153.

Goyal, V. (2001). Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21.

Green, R. O., Conel, J. E., Margolis, J. S., Carrere, V., Bruegge, C. J., Rast, M., and Hoover, G. (1990). In-flight validation and calibration of the spectral and radiometric characteristics of the airborne visible/infrared imaging spectrometer. In *Proc. SPIE 1298, Imaging Spectroscopy of the Terrestrial Environment*.

Green, R. O., Conel, J. E., and Roberts, D. A. (1993). Estimation of aerosol optical depth, pressure elevation, water vapor, and calculation of apparent surface reflectance from radiance measured by the airborne visible/infrared imaging spectrometer (AVIRIS) using a radiative transfer code. In *Proc. SPIE 1937, Imaging Spectrometry of the Terrestrial Environment*.

Green, R. O., Eastwood, M. L., Sarture, C. M., Chrien, T. G., Aronsson, M., Chippendale, B. J., Faust, J. A., Pavri, B. E., Chovit, C. J., Solis, M., Olah, M. R., and Williams, O. (1998). Imaging spectroscopy and the airborne visible / infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment*, 65(3):227–248.

Guanter, L., Kaufmann, H., Segl, K., Foerster, S., Rogass, C., Chabrillat, S., Kuester, T., Hollstein, A., Rossner, G., Chlebek, C., Straif, C., Fischer, S., Schrader, S., Storch, T., Heiden, U., Mueller, A., Bachmann, M., Mühle, H., Müller, R., Habermeyer, M., Ohndorf, A., Hill, J., Buddenbaum, H., Hostert, P., van der Linden, S., Leitão, P., Rabe, A., Doerffer, R., Krasemann, H., Xi, H., Mauser, W., Hank, T., Locherer, M., Rast, M., Staenz, K., and Sang, B. (2015). The EnMAP spaceborne imaging spectroscopy mission for earth observation. *Remote Sens.*, 7(7):8830–8857.

Guha, T. and Ward, R. (2012). Learning sparse representations for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1576–1588.

Haboudane, D., Miller, J. R., Pattey, E., Zarco-Tejada, P. J., and Strachan, I. B. (2004). Hyperspectral vegetation indices and novel algorithms for predicting green LAI of crop canopies: Modeling and validation in the context of precision agriculture. *Remote Sensing of Environment*, 90(3):337–352.

Harvey, A. R., Lawlor, J., McNaught, A. I., Williams, J. W., and Fletcher-Holmes, D. W. (2002). Hyperspectral imaging for the detection of retinal disease. In *Proceedings of the SPIE*, volume 4816, pages 325–335.

Horev, I., Bryt, O., and Rubinstein, R. (2012). Adaptive image compression using sparse dictionaries. In *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 592–595.

Hsu, C. W. and Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63.

Hughes, J. M., Rockmore, D. N., and Wang, Y. (2013). Bayesian learning of sparse multiscale image representations. *IEEE Trans. Image Processing*, 22(12):4972–4983.

Huo, C., Zhang, R., Yin, D., Wu, Q., and Xu, D. (2012). Hyperspectral data compression using sparse representation. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2012 4th Workshop on*, pages 1– 4.

Iordache, M. D., Bioucas-Dias, J. M., and Plaza, A. (2011). Sparse unmixing of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 49(6):2014–2039.

Iordache, M. D., Bioucas-Dias, J. M., and Plaza, A. (2014). Collaborative sparse regression for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 52(1):341–354.

Jelinek, F. (1968). *Probablistic Information Theory*. McGraw Hill.

Jenatton, R., Mairal, J., Obozinski, G., and Bach, F. (2010). Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, pages 487–494.

Jimenez, L. O. and Landgrebe, D. A. (1998). Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(1):39–54.

Kang, J., Gabbouj, M., and Kuo, C. (2013). Sparse/DCT (S/DCT) two-layerd representation of prediction residuals for video coding. *IEEE Transactions on Image Processing*, 22(7):2711–2722.

Kaukoranta, T., Franti, P., and Nevalainen, O. (1999). Reduced comparison search for the exact GLA. In *Proceeding of Data Compression Conference, IEEE Coupter Society*.

Kelly, E. J. (1986). An adaptive detection algorithm. *IEEE Transactions on Aerospace and Electronic Systems*, AES-22(2):115–127.

Keshava, N. and Mustard, J. F. (2002). Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57.

Kim, B., Xiong, Z., and Pearlman, W. (2000). Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT). *IEEE Transactions on Circuits and Systems For Video Technology*, 10(8):1374–1387.

Kruse, F. A., Boardman, J. W., and Huntington, J. F. (2003). Comparison of airborne hyperspectral data and EO-1 Hyperion for mineral mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 41(6):1388–1400.

Kwan, C., Ayhan, B., Chen, G., Wang, J., Ji, B., and Chang, C.-I. (2006). A novel approach for spectral unmixing, classification, and concentration estimation of chemical and biological agents. *IEEE Transactions on Geoscience and Remote Sensing*, 44(2):409–419.

Landgrebe, D. A. (1999). Information extraction principles and methods for multispectral and hyperspectral image data. In Chen, C., editor, *Information Processing for Remote Sensing*, pages 3–38. Worls Scientific Publishing Company.

Lee, H., Younan, N., and King, R. (2002). Hyperspectral image cube compression combining JPEG2000 and spectral decorrelation. In *IEEE Geoscience and Remote Sensing Symposium (IGARSS'02)*, volume 6, pages 3317–3319.

Li, C.-G., Lin, Z., and Guo, J. (2013). Bases sorting: Generalizing the concept of frequency for over-complete dictionaries. *Neurocomputing*, 115(Supplement C):192 – 200.

Li, S., Yin, H., and Fang, L. (2012). Group-sparse representation with dictionary learning for medical image denoising and fusion. *Biomedical Engineering, IEEE Transactions on*, 59(12):3450–3459.

Li, W. and Du, Q. (2014). Gabor-filtering-based nearest regularized subspace for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(4):1012–1022.

Li, W. and Du, Q. (2015). Adaptive sparse representation for hyperspectral image classification. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4955–4958.

Li, W., Tramel, E. W., Prasad, S., and Fowler, J. E. (2014). Nearest regularized subspace for hyperspectral classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(1):477–489.

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95.

Lloyed, S. P. (1982). Least squares quantization in PCM. *IEEE Trans. Inf. Theor.*, 28(2):127–135.

Lu, T., Li, S., Fang, L., Ma, Y., and Benediktsson, J. A. (2016). Spectral – spatial adaptive sparse representation for hyperspectral image denoising. *IEEE Transactions on Geoscience and Remote Sensing*, 54(1):373–385.

Ma, W. K., Bioucas-Dias, J. M., Chan, T. H., Gillis, N., Gader, P., Plaza, A. J., Ambikapathi, A., and Chi, C. Y. (2014). A signal processing perspective on hyperspectral unmixing: Insights from remote sensing. *IEEE Signal Processing Magazine*, 31(1):67–81.

Magli, E., Olmo, G., and Quacchio, E. (2004). Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC. *IEEE Geoscience and Remote Sensing Letters*, 1(1):21–25.

Mahmood, A., Robin, A., and Sears, M. (2017). Modified residual method for the estimation of noise in hyperspectral images. *IEEE Trans. Geosci. Remote Sensing*, 55(3):1451–1460.

Mairal, J., Bach, F., and Ponce, J. (2014). Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009). Online dictionary learning for sparse coding. In *26th International Conference on Machine Learning, Montreal, Canada*.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60.

Mairal, J., Elad, M., and Sapiro, G. (2008a). Sparse representation for color image restoration. *Image Processing, IEEE Trans. On*, 17(1):53–69.

Mairal, J., Sapiro, G., and Elad, M. (2008b). Learning multiscale sparse representations for image and video restoration. *Multiscale Modeling & Simulation*, 7(1):214–241.

Mairal, J. and Yu, B. (2012). Complexity analysis of the Lasso regularization path. In Langford, J. and Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning*

(ICML-12), pages 353–360, New York, NY, USA. ACM.

Mallat, S. (2008). *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition.

Mallat, S. and Zhang, Z. (1993). Matching pursuit with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415.

Manolakis, D., Marden, D., and Shaw, G. A. (2003). Hyperspectral image processing for automatic target detection applications. *Lincoln Laboratory Journal*, 14(1):79–116.

Meer, F. D. V. D. and de Jong, S. M., editors (2001). *IMAGING SPECTROMETRY: Basic Principles and Prospective Applications (Remote sensing and digital image processing, Vol 4)*. Dordrecht etc.: Kluwer Academic.

Melgani, F. and Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790.

Mercier, G. and Lenon, M. (2003). Support vector machines for hyperspectral image classification with spectral-based kernels. In *2003 IEEE InternationalGeoscience and Remote Sensing Symposium, IGARSS 2003.*, volume 1, pages 288–290.

Mitani, Y. and Hamamoto, Y. (2006). A local mean-based nonparametric classifier. *Pattern Recogn. Lett.*, 27(10):1151–1159.

Motta, G., Rizzo, F., and Storer, J. A., editors (2009). *Hyperspectral Data Compression*. Springer-Verlag.

Moussaoui, S., Hauksdottir, H., Schmidt, F., Jutten, C., Chanussot, J., Brie, D., Doute, S., , and Benediktsson, J. A. (2008). On the decomposition of data hyperspectral data by ica and bayesian positive source separation. *Neurocomputing*, 71:2194–2208.

NASA (2018). Remote sensors. `https://earthdata.nasa.gov/user-resources/remote-sensors`. 2018-01-12.

NASA JPL (2015). AVIRIS. http://aviris.jpl.nasa.gov/aviris/index.html. 2015-04-08.

Nasrabadi, N. M. (2014). Hyperspectral target detection : An overview of current and future challenges. *IEEE Signal Processing Magazine*, 31(1):34–44.

Olshausen, B. A. and Field, D. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311 – 3325.

Osborne, M. R., Presnell, B., and Turlach, B. A. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 40–44.

Pearlman, W., Islam, A., Nagaraj, N., and Said, A. (2004). Efficient, low-complexity image coding with a set-partitioning embedded block coder. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(11):1219–1235.

Penna, B., Tillo, T., Magli, E., and Olmo, G. (2007). Transform coding techniques for lossy hyperspectral data compression. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(5):1408–1421.

Pickering, M. and Ryan, M. (2001). Efficient spatial-spectral compression of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 39(7):1536–1539.

Pickering, M. and Ryan, M. (2006). An architecture for the compression of hyperspectral imagery.

In Motta, G., Rizzo, F., and Storer, J. A., editors, *Hyperspectral Data Compression*, pages 1–34. Springer.

Plaza, A., Benediktsson, J. A., Boardman, J. W., Brazile, J., Bruzzone, L., Camps-valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., Marconcini, M., Tilton, J. C., and Trianni, G. (2009a). Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113:S110–S122.

Plaza, J., Hendrix, E. M. T., García, I., Martín, G., and Plaza, A. (2012). On endmember identification in hyperspectral images without pure pixels: A comparison of algorithms. *Journal of Mathematical Imaging and Vision*, 42(2-3):163–175.

Plaza, J., Plaza, A., Perez, R., and Martinez, P. (2009b). On the use of small training sets for neural network-based characterization of mixed pixels in remotely sensed hyperspectral images. *Pattern Recognition*, 42(11):3032–3045.

Qian, S. (2004). Hyperspectral data compression using a fast vector quantization algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8).

Qian, Y., Ye, M., and Zhou, J. (2013). Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features. *IEEE Transactions on Geoscience and Remote Sensing*, 51(4):2276–2291.

Rabbani, M. and Joshi, R. (2002). An overview of the JPEG 2000 still image compression standard. *Signal Processing: IMage Compression*, 17(1):3–48.

Ravn, C., Skibsted, E., and Bro, R. (2008). Near-infrared chemical imaging (nir-ci) on pharmaceutical solid dosage forms-comparing common calibration approaches. *Journal of Pharmaceutical and Biomedical Analysis*, 48(3):554–561.

Reed, I. S. and Yu, X. (1990). Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution. *IEEE Transactions on Acoustics Speech and Signal Processing*, 38(10):1760–1770.

Rizzo, F., Storer, J., and Carpentieri, B. (2005). Overlap and channel errors in adaptive vector quantization for image coding. *Information Sciences*, 171(1-3).

Roger, R. and Cavenor, M. (1996). Lossless compression of AVIRIS images. *IEEE Transactions on Image Processing*, 5(5):713–719.

Rubinstein, R., Zibulevsky, M., and Elad, M. (2008). Efficient implemetation of the K-SVD algorithm and the Batch-OMP method. Technical Report CS-2008-08, Technion – Israel Institute of Technology, Haifa, Israel.

Said, A. (2003). Arithmetic coding. In Sayood, K., editor, *Lossless compression handbook*, pages 101–152. Academic Press.

Said, A. and Pearlman, W. (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems For Video Technology*, 6(3):243–250.

Sallee, P. and Olshausen, B. A. (2003). Learning sparse multiscale image representations. *Adv. Neural Inf. Process. Syst.*, 15:1327–1334.

Sayood, K. (2005). *Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Schaepman, M. E. (2009). Imaging spectrometers. In Warner, T. A., Nellis, M. D., and Foody, G. M., editors, *The SAGE Handbook of Remote Sensing*, pages 166–178. SAGE Publication

Ltd.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Shao, G., Wu, Y., A, Y., Liu, X., and Guo, T. (2014). Fingerprint compression based on sparse representation. *IEEE Transactions on Image Processing*, 23(2):489 – 501.

Shapiro, J. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462.

Shaw, G. A. and Burke, H. K. (2003). Spectral imaging for remote sensing. *Lincoln Laboratory Journal*, 14(1):3–28.

SKA (2015). Square kilometre array. https://www.skatelescope.org/. 2018-01-10.

SKA SA (2015). The SKA project. http://www.ska.ac.za/about/project.php. 2015-06-03.

Skretting, K. and Engan, K. (2010). Recursive least squares dictionary learning algorithm. *IEEE Transactions on Signal Processing*, 58(4):2121–2130.

Skretting, K. and Engan, K. (2011). Image compression using learned dictionaries by RLS-DLA and compared with K-SVD. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1517–1520.

Soltani-Farani, A., Rabiee, H. R., and Hosseini, S. A. (2013). Spatial-aware dictionary learning for hyperspectral image classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(1):527–541.

Sun, X., Nasrabadi, N. M., and Tran, T. D. (2015). Task-driven dictionary learning for hyperspectral image classification with structured sparsity constraints. *IEEE Transactions on Geoscience and Remote Sensing*, 53(8):4457–4471.

Sun, X., Qu, Q., Nasrabadi, N. M., and Tran, T. D. (2014). Structured priors for sparse-representation-based hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 11(7):1235–1239.

Tang, X., Cho, S., and Pearlman, W. (2003a). 3D set partitioning coding methods in hyperspectral image compression. In *The International Conference on Image Processing, In Proceedings of*, volume 2, pages 239–242, Barcelona, Spain.

Tang, X. and Pearlman, W. (2006). Three-dimensional wavelet-based compression of hyperspectral images. In Motta, G., Rizzo, F., and Storer, J. A., editors, *Hyperspectral Data Compression*, chapter 10, pages 273–308. Springer.

Tang, X., Pearlman, W. A., and Modestino, J. W. (2003b). Hyperspectral image compression using three-dimensional wavelet coding. In *Proc. SPIE Volume 5022*, pages 5022 – 5022 – 11.

Taubman, D. (2000). High performance scalable image compression with EBCOT. *Image Processing, IEEE Trans. On*, 9(7):1158–1170.

Taubman, D. S. and Marcellin, M. W. (2002). JPEG 2000: standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological).*, 58(1):267–288.

Tikhonov, A. and Arsenin, V. (1977). *Solutions of ill-posed problems*. Johan Wiley & Sons, Inc.

Tropp, J. A., Gilbert, A. C., and Strauss, M. J. (2006). Algorithms for simultaneous sparse approximation: Part i: Greedy pursuit. *Signal Process.*, 86(3):572–588.

Ülkü, I. and Töreyin, B. (2014). Lossy compression of hyperspectral images using online learning based sparse coding. In *Computational Intelligence for Multimedia Understanding (IWCIM), 2014 International Workshop on*, pages 1–5.

Ülkü, I. and Töreyin, B. (2015). Sparse representations for online-learning-based hyperspectral image compression. *Applied Optics*, 54(29):8625–8631.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

Vetterli, M. and Kovacevic, J. (1995). *Wavelet and Subband Coding*. Prentice Hall.

Wang, H. and Celik, T. (2016). Sparse representation based lossy hyperspectral data compression. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2761–2764.

Wang, H. and Celik, T. (2017). Sparse representation-based hyperspectral data processing: Lossy compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(5):2036–2045.

Wang, Y., Rucker, J., and Fowler, J. (2004). Three-dimensional tarp coding for the compression of hyperspectral images. *Geoscience and Remote Sensing Letters, IEEE*, 1(2):136–140.

Wei, Q., Bioucas-Dias, J., Dobigeon, N., and Tourneret, J. Y. (2015). Hyperspectral and multispectral image fusion based on a sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 53(7):3658–3668.

Winter, M. E. (1999). N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. In *Proc. SPIE Vol 3753, Imaging Spectrometry V*, pages 266–275.

Wu, Q., Zhang, R., and Wang, F. (2014). Hyperspectral data compression using double sparsity model. In *2014 Workshop on Hyperspectral Image and Signal Processing (WHISPERS)*.

Wu, X. (1997). Lossless compression of continuous-tone images via context selection, quantization, and modeling. *IEEE Transactions on Image Processing*, 6(5):656–664.

Wu, X. and Memon, N. (1997). Context-based, adaptive, lossless image coding. *IEEE Transactions on Communications*, 45(4):437–444.

Wu, X. and Memon, N. (2000). Contex-based lossless interband compression — extending CALIC. *IEEE Transactions on Image Processing*, 9(6):994–1001.

Xu, Y., Wu, Z., Li, J., Plaza, A., and Wei, Z. (2016). Anomaly detection in hyperspectral images based on low-rank and sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 54(4):1990–2000.

Yokoya, N., Yairi, T., and Iwasaki, A. (2012). Coupled nonnegative matrix factorization unmixing for hyperspectral and multispectral data fusion. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):528–537.

Zare, A., Bolton, J., Gader, P., and Schatten, M. (2008). Vegetation mapping for landmine detection using long-wave hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 46(1):172–178.

Zhao, Y. Q. and Yang, J. (2015). Hyperspectral image denoising via sparse representation and low-rank constraint. *IEEE Transactions on Geoscience and Remote Sensing*, 53(1):296–308.

Zhou, M., Chen, H., Paisley, J., Ren, L., Li, L., Xing, Z., Dunson, D., Sapiro, G., and Carin, L. (2012). Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images. *IEEE Transactions on Image Processing*, 21(1):130–144.

Zou, J., Li, W., and Du, Q. (2015). Sparse representation-based nearest neighbor classifiers for hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 12(12):2418–2422.