HYPERSPECTRAL IMAGE COMPRESSION
USING A TWO-STAGE KLT

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Electrical Engineering

by
Seton S. Schroeder

May 2009

© 2009
Seton S. Schroeder
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Hyperspectral Image Compression using a Two-stage KLT

AUTHOR: Seton S. Schroeder

DATE SUBMITTED: May 1, 2009

COMMITTEE CHAIR: John A. Saghri, Associate Professor

COMMITTEE MEMBER: Fred W. DePiero, Professor

COMMITTEE MEMBER: Jane Zhang, Assistant Professor

ABSTRACT

Hyperspectral Image Compression using a Two-stage KLT

Seton S. Schroeder

A computationally efficient, lossy, bandwidth compression scheme for hyperspectral imagery is presented. In certain cases, the direct application of the standard KLT algorithm over an entire data set is not practical. The data may be too large for efficient processing by one encoder and so portions of the entire data set may necessarily be sampled by different encoders. The algorithm proposed handles such a limitation by leveraging the standard JPEG 2000 technology. The component decorrelation of the JPEG 2000 (extension 2) is replaced with a two-stage Karhunen-Loeve Transformation (KLT) operation resulting in a reduction in the computational complexity. At the first stage the spectral data is partitioned into sets of adjacent spectral images, which are decorrelated independently of one another. At the second-stage inter-partition correlation is exploited by combining principle component images and performing a second set of KLT operations. It is shown that reconstructed image quality is improved by minimizing partition size at the first stage. The method is evaluated using the criteria of root-mean-square error as well as the performance of a terrain-classification algorithm. A relationship between partition size and minimal inter-partition decorrelation is also proposed and may act as a guide in the tuning of compression settings.

TABLE OF CONTENTS

List of Tables	vi
List of Figures	vii
Chapter 1 Introduction	1
Chapter 2 Background	3
2.1 Multispectral Images	3
2.2 KLT in Compression and Transmission of Multispectral Images.....	4
Chapter 3 Recent Data-compression Approaches	7
3.1 Bidirectional Interband Prediction Method	7
3.2 Distributed Karhunen-Loeve Transform	9
3.2.1 <i>Local KLT Constraint</i>	11
3.2.2 <i>Distributed KLT Algorithm</i>	15
3.3 Sub-optimal KLT	15
Chapter 4 Thesis Methods	18
4.1 Spectral Compression Technique	18
4.1.1 <i>Protocol</i>	19
4.1.2 <i>KLT Algorithm</i>	20
4.2 Spacial Compression Technique	22
4.3 Reconstruction	24
4.4 Classification	24
4.5 Quantitative Assessment of Reconstruction	25
4.5.1 <i>Root-mean-squared Error</i>	26
4.5.2 <i>Confusion Matrix</i>	26
4.6 PC Contribution to Second-stage KLT	27
Chapter 5 Results	29
Chapter 6 Conclusion and Future Work	37
Bibliography	47
Appendix	48
The Karhunen-Loeve Transform	48
<i>Concepts</i>	48
<i>Derivation</i>	51
Matlab Code	54

LIST OF TABLES

Table 5-1 8 File/group Confusion Matrix 34

Table 5-2 16 File/group Confusion Matrix 34

Table 5-3 20 File/group Confusion Matrix 35

LIST OF FIGURES

Figure 2-1 The Electromagnetic Spectrum	4
Figure 2-2 Spectral Image Compression and Reconstruction	6
Figure 3-1 Marginal KLT	10
Figure 3-2 Local KLT	13
Figure 3-3 Sub-optimal KLT	17
Figure 4-1 Two-stage KLT Operation	20
Figure 5-1 Spectral-grouping Effect on Image Degradation	29
Figure 5-2 Ground Truth Thematic Map	30
Figure 5-3 8 File/group Thematic Map	31
Figure 5-4 16 File/group Thematic Map	32
Figure 5-5 20 File/group Thematic Map	33
Figure 5-6 Second-stage Decorrelation Analysis	36
Figure 6-1 Partitioning and Encoding in JP3D	44
Figure 6-2 JP3D scanning Pattern	44
Figure A-1 KLT Conceptual Example	49
Figure A-2 Effect of Spectral Axis Rotation on Data Variance.....	52

Chapter 1 Introduction

Imaging technology has made it possible to extract a great deal of information about the content of the Earth's surface and atmosphere. One of the most advanced systems, the hyperspectral imager, has allowed the capture of electromagnetic waves over the visible and infrared regions, with high spectral resolution. The motivation for the study of hyperspectral image processing lies in the number of useful applications the field has. In agriculture, it aids in the identification of soil composition. It is used in environmental studies to determine the degree of land and air pollution, as well as forest monitoring. Hyperspectral imaging also has uses for the military in surveillance. Many of these applications require gathering information of expansive regions. Therefore, aircraft and satellites are often the platforms used to capture such large scene. However the bandwidth required to transmit the information to a ground station on Earth is currently not sufficient for the image data in its raw form. Therefore, compression techniques that attempt to further optimize the quality and usability of transmitted data continue to be investigated and used.

The purpose of this project was, firstly, to review a few of the most current multispectral image compression techniques. The second aim was to propose a computationally simpler alternative that can handle the restrictive scenario where the spectral images have been partitioned and compressed at different terminals. A quantitative goal was to compress 144, 256x256 pixel spectral images at 0.25 bpp such that the reconstructed images exhibited a root-mean-squared error of less than 1%. Looking at variations in image reconstruction quality due to changes in spatial compression ratios, however, was not an aim of this project. A second goal of this thesis

was to receive a total classification accuracy of greater than 95% when applying the reconstructed image to a 6-class, minimum Euclidean distance, general hill climb, unsupervised classification scheme. Iterations were stopped after a 1.8% change in pixel classification was achieved. This produced less than 10 iterations in every case.

Chapter 2 Background

The data processing performed in this project made use of images of reflected electromagnetic wave information, covering a range of frequencies. The first step involves a typical procedure of spectrally decorrelating data and removing spectral redundancies. Once the data is spectrally compressed, the images are converted to a format that removes spacial redundancies. Finally, the images are decompressed and the reconstructed image data are compared, pixel-by-pixel, to the original spectral image data, using a root mean squared error calculation.

2.1 Multispectral Images

One technique for gathering information about the content of atmosphere or land is through the data provided by multispectral images. Imaging systems have made it possible to capture the radiant energy that is reflected and emitted from the air and land cover. This energy is carried on the various wavelengths of the electromagnetic spectrum. The spectrum can be divided into 7 bands. Ranging from lowest wavelength to highest wavelength, these bands are: Radio waves, Microwaves, Infrared, Visible, Ultra Violet, X-rays, and Gamma rays (Figure 2-1). Human eyes can only detect energy delivered from an object at wavelengths that fall within the Visible Light spectrum, which ranges from 0.4 μm (the color violet) to 0.7 μm (the color red). Special sensors, however, allow information to be captured at wavelengths outside this range. Hyperspectral imagers specifically detect the Visible, Near Infrared, Mid Infrared, and Thermal regions. The images captured by such technology often then go through a compression scheme such as the Karhunen-Loeve Transform, before they can be transmitted.

Figure 2-1 The Electromagnetic Spectrum. Spectral images contain information carried by electromagnetic waves categorized by their wavelengths [1].

2.2 KLT in Compression and Transmission of Multispectral Images

The Karhunen-Loeve Transformation (KLT) operation is the foundation for the spectral compression method used in this project. It is a standard technique used to compress and separate a scene's multiple spectral images into 'principal component' images of uncorrelated data. To do such, linear algebra techniques are utilized. See the Appendix for a conceptual description and mathematical derivation of the process.

The process of converting spectral images of a scene into its principle components using the KLT only compresses information that varies with spectral frequency. However, before such information can be transmitted over limited bandwidths, it must be spatially compressed as well. Joint Photographic Experts Group (JPEG) is an organization that developed a 2-D image compression standard to do just that, JPEG 2000 [2].

When images are sent in a compressed JPEG 2000 format from a satellite to Earth they are then decompressed. The process of decompressing the image data involves first converting the data back to its raw data format via the inverse JPEG 2000 operation. This puts the data into a form that can allow for spectral reconstruction of the original images. An inverse KLT operation is performed to bring the principle component images back to

their spectral-image form. If all the image data is used in the compression, transmission, and reconstruction process then, in theory, no information should be lost. In practice, this would require no round-off and/or quantization errors, in the computational process. Such perfect reconstruction is known as lossless transmission. If only a subset of the principle component images are transmitted, the process is lossy, and the reconstructed image will not contain all the original image information. Both KLT and JPEG 2000 were used in this project to spectrally and spacially compress and decompress the image data in a lossy fashion (Figure 2-2). But before further describing the methods used in this project, a review of some other recent data-compression approaches will be analyzed to set a basis for the evaluation of this thesis' approach.

Figure 2-2 Spectral Image Compression and Reconstruction. The spectral images are compressed and reconstructed using KLT and JPEG operations.

Chapter 3 Recent Data-compression Approaches

3.1 Bidirectional Interband Prediction Method

Ideally, it is best to use all the pixel data of one's spectral images for a particular scene, however transmitting all the data is not feasible due to downlink channel capacity. Therefore, one approach is to assume a linear relationship between the intensities (due to reflection and/or emitted energy) of particular bands, in order to predict other pixel values. If one were to have a limited set of spectral images, one could estimate the parameters that describe the linearity between each two-band associations in a set. For example, if one were to observe the intensities of the same pixel from two spectral images of a ground scene, the relationship between these intensities could be written as such:

$$I_1(p_1) = aI_2(p_1) + b \quad (1),$$

where I_1 and I_2 are the intensities of a pixel, p_1 , for two arbitrary bands [3]. The parameter, a , describes the rate at which I_1 changes as I_2 changes for the given feature. The parameter, b , can typically be described as the average difference between the intensity of the two bands for the given feature. To make computation easier, one could determine a set of parameters for equally-sized block partitions of a whole scene. The larger the size of the blocks, the lower the overhead associated with the transmitted prediction parameters. The tradeoff, however, is a larger prediction error - especially with busier scenes.

This linear model is straight forward, however what is not, is the determination a reference band, from which predictions of one of the other bands will be made. In other words, one would have to decide on which spectral image's intensities to take, I_2 , from

which to make a prediction of the spectral illumination, I_l , of a pixel at another wavelength. An approach to this is to simply take the band that yields the minimum sum of prediction error variances when a best-fit linear relationship is made with all the other bands, of all blocks, for all pixels. Simulations have demonstrated that using a middle wavelength as a reference in either the visible region or infrared region provides a better prediction for other wavelengths [3].

However, there are downsides to this linear prediction method. The coding noise accumulates significantly at lower bit rates, causing coding distortion in the anchor band, reducing the correlation accuracy with the prediction band. Also, using only one band as a reference is not always an effective enough constraint for predicting the intensity levels of another band. Therefore, a method that uses two reference bands to predict a single prediction band was developed, which solves both of the issues just described [3].

The modified linear equation that makes use of two reference bands is similar to the single-reference band equation, but with an additional scaling term:

$$I_1(p_1) = aI_2(p_1) + bI_3(p_1) + c \quad (2),$$

where the prediction band intensity is a sum of weighted contributions from two reference bands, in addition to a constant offset, c . In this new scenario each prediction band for each block is associated with two particular reference bands. Depending on the location of the prediction band, one of the anchors is a band that falls in the middle of the Visible or Infrared regions. The second anchor band falls on the other side of the prediction band, so that the prediction band is flanked. This second band is determined by starting at the lowest available band, if the prediction band is higher, and moving up one band at a time band. If the prediction band is lower than the second band, then the

iterations of second band choices starts at the highest possible band. For each secondary reference band chosen, a root mean squared error is determined. When the error reaches a local minima, iterations cease and the current reference bands at that iteration are used.

The bidirectional prediction scheme is very effective at providing better-quality reconstructed images while adding little additional parameter data as overhead. The use of two reference bands minimizes further the mean square error values between the original and reconstructed images without much more overhead. However, this method requires that, for every band prediction made, a whole set of linear equations be determined. Furthermore, each band is processed separately. This can be very impractical for a large number of spectral bands. The Visible band and Infrared band references that optimize the reconstruction can vary as well, depending on the number of bands used. Finally, the process of finding the second reference band, at best, provides a local minima error. However, a method based on heuristics can often do better (e.g. using the median visible and infrared bands as anchors and using sample pixels in the calculation of a maximum covariance matrix for the selection of the second reference band).

3.2 Distributed Karhunen-Loeve Transform

Applying the Karhunen-Loeve Transform to distinguish correlating bands, although just as good as a bidirectional interband prediction method, can carry much more computation and overhead [3]. This is particularly true when one attempts to apply a single KLT operation to the entire set of a very-large number of bands, describing a whole scene. When applied in this manner, the KLT is known specifically as a *direct*, *full*, or *joint* KLT. This method may also not even be possible if different band regions are fed into separate, non-communicating terminals, before being collected into a fusion

center. The terminals perform compression and encoding only on the sub-spectrum of images that they receive. They are transmitted to their respective receivers and then processed together. A simple alternative, in this scenario, would be to apply a separate KLT operation to each set of spectral images for each terminal. This local approach to the KLT strategy is known as a *marginal* KLT (Figure 3-1). The downside to this method, however, is that it completely ignores the correlation between bands detected by separate terminals. The

Figure 3-1 Marginal KLT. One approach to a KLT operation on large number of spectral images is to perform it on groups of spectral images, x_i , separately. The transformed images are passed through individual encoders and spacially compress at their own bit rates, T_i [5].

operation only makes use of local statistics. Given the constraints of a partitioned processing method for a set of spectral images, an optimal KLT approach would lie somewhere between the ideal *joint* KLT and the oversimplified *marginal* KLT technique. Such an approach, which provides the information of reconstructed spectral images to those spectral images that have yet to be processed, is termed the *distributed* KLT [3].

A *distributed* KLT algorithm that minimizes the approximation error or distortion rate of compressed and reconstructed images can be derived if one looks at a particular constraining scenario. The scenario that will be investigated is one where all but one of the terminals has a fixed transform matrix and the goal is to optimize the transformation matrix of the remaining terminal given the eigenvectors of the others, yielding the optimal *local* KLT [3].

3.2.1 Local KLT Constraint

As with a *marginal* KLT approach, the *local* KLT scenario looks at an individual terminal and performs a KLT. The difference is the conditionality of the parameters used in the KLT. The transformation matrix of the *marginal* KLT and eigenvalues are affected only by local statistics. In a *local* KLT operation, the eigenvalue and eigenvector values of the isolated terminal are determined from the results of the transformation parameters of the other terminals. The parameters of the transformation at the isolated terminal are chosen such that the total mean squared error after reconstruction, is minimized. The method for minimizing this error is called linear approximation.

A simple solution for determining the best linear approximations for every terminal, in the *marginal* KLT case, does not seem to exist in general [5]. Therefore, the *local* KLT scenario is a next-best approach. When it comes to the spacial compression of multiple terminals in the *marginal* KLT scenario, the rates of compression, which are related to the local eigenvalues, are independent of each other. This independency does not make for fully optimized compression rates either. Unfortunately, the optimal compression rates for a multiple-terminal scenario is mostly unknown to date, except in the case of the two-terminal case [5]. Therefore, in the *local* KLT scenario one can divide

the spectral images into two partitions. One partition simply contains the spectral images of the terminal whose linear approximation we are trying to optimize. The second partition consists of the spectral images from all the other terminals together. The terminals of this second partition are given user-defined compression rates from which the isolated terminal's compression rate will be determined. However, this all depends on the first step of creating two spectral partitions, which will now be described in more detail.

Once the spectral images are partitioned, a new covariance matrix can be created, which segments the covariance matrix of the full set of spectral images. The new covariance matrix can be seen best in its matrix form:

$$C_{x'} = \begin{bmatrix} C_{x1} & C_{x12} \\ C_{x21} & C_{x2} \end{bmatrix} \quad (3),$$

where C_{x1} is simply the local covariance matrix of the single-terminal partition. Likewise, C_{x12} is the set of covariances between every spectral value from the single-terminal partition and every spectral value from the multi-terminal partition. The covariance matrices are defined in the same manner. Furthermore, the multiple-terminal partition's linear approximation can be expressed as:

$$y2 = P_2 x2 + z2 \quad (4),$$

where x and y are vectors that denote a pixel's original spectral values and principle component values, respectively [5]. P_2 is the matrix of eigenvectors, arranged in rows, of the covariance matrix for the multi-terminal partition. The vector, $z2$, is the Gaussian random noise associated with the chosen linear approximation component images' pixel. The goal is to optimize the transformation matrix of the isolated terminal such that the distortion of the reconstructed spectral images, $\hat{x}$, is minimized (Figure 3-2).

Figure 3-2 Local KLT. Another approach to performing a KLT operation to a large number of spectral images is to partition the spectral images into two groups, x_1 and x_2 . The transformation parameters for one partition is fixed and used to optimize the other [5].

In order to derive a new set of optimal eigenvectors for the *local* KLT, a new covariance matrix is constructed. This matrix is associated with the values of the partitioned covariance matrix, C_x . We start by considering a matrix A , defined as:

$$A = [C_{x21} \ C_{x2}P_2'] \quad -1 \quad (5) .$$

A is a matrix with number of rows equal to the number of spectral bands originally grouped into the multi-terminal partition. Its number of columns equals the number of original spectral bands in the single-terminal partition plus the number of eigenvectors in the multi-terminal partition. Let A_I consist of the first set of columns equal to the number of original spectral bands in the single-terminal partition, M . We can then define the new covariance matrix for our linear approximation as:

$$C_{w'} = [C_{x1} - C_{x12}P_2'(P_2C_{x2}P_2' + P_z)^{-1}P_2C_{x12}][I_M \ A_I'] \quad (6) .$$

This equation can be written in its simpler form as:

$$C_{w'} = Q_w'D_w'Q_w' \quad (7) .$$

One will notice the similarity of this equation to that defining the relationship between the covariance matrix of spectral values of an image and its associate eigenvectors, described under the KLT derivation found in the Appendix. Recall,

$$C_X = EDE' \quad (8) ,$$

where C_X was the covariance matrix, E was the eigenvector matrix (written as columns), and D was a diagonal matrix describing the eigenvalues. In fact, in our new *local* KLT algorithm, the eigenvectors are the columns of Q_w and the eigenvalues are the diagonal values of D_w . We relate the original single-terminal partition's transformation matrix to Q_w and A_I as follows:

$$P_I = [Q_w^{(M)}] \begin{bmatrix} I_M \\ A_1 \end{bmatrix} \quad (9) ,$$

where $Q_w^{(M)}$ refers to the first M columns of Q_w and I_M is an M dimensional identity matrix.

To determine, in practice, the most appropriate transformation parameters to use in the *local* KLT scenario, one must have access to y_2 (or $P_2x_2 + z_2$). A constraint is that the best linear approximation is dependent on the values used for y_2 . The scheme for optimizing spacial compression, for the *local* KLT case, is similar to the method for spectral compression. One chooses an arbitrary fixed rate for our multi-terminal partition, and use this to determine an optimal rate for the single-terminal partition [5]. There does not seem to be any simple, direct solution to this *local* KLT approach, however, achieving local minima root-mean-squared errors have been demonstrated. Furthermore, the method has been proven to be more effective than a *marginal* KLT approach [5].

3.2.2 Distributed KLT Algorithm

The *distributed* KLT approach is an improvement on the *local* KLT constraint. It is similar in the sense that the linear approximation of one partition is optimized, while the KLT operation on the rest of the spectral data is held fixed [5]. However, the *distributed* KLT separates the terminals into more than two partitions. One partition acts as the single-terminal partition, as described in the *local* KLT scenario, and the transformation matrices of all the other partitions are not changed [5]. The *distributed* KLT method is performed under this situation, but using only a subset of pixels from the scene image. After optimization of the single-terminal partition's transformation matrix, a single pixel is added to the sample and a new 'single-terminal partition' is chosen before a *local* KLT operation is performed again. Every time a new pixel is added to complete the scene, a different partition's linear approximation is optimized. As the number of pixels used in the *local* KLT operation increases, the linear approximations of each partition will improve. The chances of reaching a local minima is dependent on the random choice sequence of which terminal to optimize. However, the iterations can be performed multiple times to the scene image until a local minima is found. Optimal spatial compression rates for each partition can be computed in a similar fashion.

3.3 Sub-optimal KLT

Another simpler approach to addressing the correlations between groupings of spectral bands is to apply the *sub-optimal* KLT [8]. In this method, the original spectral images are divided into groups of the same number of images. The number of files is arbitrary but allows for an easy computation of the KLT for each group. A subset of the principle component images associated with the highest eigenvalues, are then combined

into sets of new groups, and a second stage of KLT operations is performed. The unused first-stage principle components are sent directly to the encoder for transmission.

Once a second-stage set of principle components are generated, the principle component images of non-combined spectral images are combined. As before, images of larger eigenvalues are paired and passed through yet another KLT operation. The same is done with the lower eigenvalue images. The process continues up to an arbitrary point. This process can be performed by breaking up groups into pairs of 8 images each (Figure 3-3). The division of the KLT groups into two-level sub-blocks of three KLT operations, is performed because it has been shown that such an architecture yields insignificant degradation in the coding gain for a large number of spectral images (0.3 % for $N = 64$). However, this assumes that the signals used reflect a Markov process [8]. Still, such an approach allows for more thorough removal of spectral redundancy across the entire spectrum of images, and therefore greater compression with minimal effect on coding gain.

Figure 3-3. Sub-optimal KLT. This method interleaves principle component images at multiple stages to make use of the correlation between spectral information in groups along the entire spectra of images [8].

Chapter 4 Thesis Methods

This thesis project involved a data compression scheme similar to the *sub-optimal* KLT approach. However, it addresses the interband correlation issues highlighted in the development of the *distributed* KLT algorithm. The bidirectional interband prediction approached was not incorporated, but helped to support the notion that the grouping of spectral image patterns shared by multiple spectral images can lead to better reconstruction results upon decompression.

The spectral image processing, in the case of this project, involved images taken of a region of San Francisco. Raw hyperspectral images of the scene, taken by a HIRIS (Hyperspectral Infrared Imaging System) instrument onboard an airplane, were utilized [9]. Each image provided the reflected illumination from the city for a very particular, narrow, electromagnetic bandwidth. These multispectral images were spectrally compressed using a very particular computationally-effective KLT algorithm written in Fortran. This was followed by spacial compression using the JPEG 2000 standard. The images were then decompression and reconstructed so that the resulting and original spectral images could be quantitatively compared. Statistical calculations and a feature classification process were performed using Matlab.

4.1 Spectral Compression Technique

The 144, 256 x 256 pixel, spectral images were AVIRIS overhead the city [9]. The compression algorithm used in this project involved a two-stage KLT operation. There are many computational algorithms that can achieve what the KLT operation tries to accomplish. In this paper, one of the simpler, but effective, algorithms was applied.

4.1.1 Protocol

The protocol for using the KLT operation, firstly, consisted of separating all the files into groups of spectrally adjacent images. Each group contained the same number of images. An exception to this is if the number of images chosen for each group was not divisible by 144, in which case one group would contain fewer images. Secondly, the KLT operation, described in Appendix, was performed separately on each group of the images, one group at a time. This process of converting all the spectral images to their principle component images was performed three times. Each time the number of images used in a group was different. Results were observed for: 20, 16, and 8 files per group.

Once the KLT operation was applied, inter-group correlation had to be addressed. An approach was therefore taken in which the groups of resulting principle component images were interleaved and regrouped so that all the first principle component images were put into one group, all the second principle component images were put into another group, and so forth, down to four principle components deep. At this point, the KLT operation was performed once again, however this time, on each group of principle component images, one group at a time (Figure 4-1).

Figure 4-1. Two-stage KLT Operation. The first-stage KLT operation is performed on the ‘m’ groups of ‘n’ spectral images each, $X((i-1)n+1) - X(in)$, where $i = 1, 2, 3 \dots m$. This results in ‘m’ sets of PC images. The PC images are then regrouped by level (1-4 in this example) and each of these groups, of ‘m’ PC images each, goes through a second KLT operation. The result is sets of second-stage PC images, Y, where the number of sets is equal to the depth of first-stage PC images that were regrouped for second stage transformation.

4.1.2 KLT Algorithm

The objective of the KLT is clear if one understands the operation conceptually and linear algebra proofs being utilized. However, executing the operation, computationally, is not so straightforward. Perfect results cannot be achieved, therefore a number of algorithms have been designed to get ‘good enough’ approximations.

As the KLT derivation in the Appendix indicates, the covariance matrix of our spectral values, X , can be expressed as:

$$C_X = EDE^T \quad (1),$$

where one tries to find the appropriate eigenvector values, E , that will diagonalize D .

While real non-symmetric matrices can be diagonalized, the transformation matrix is not necessarily real [7]. However, a real similarity transformation can get close to doing the job. It can reduce the matrix to many two-by-two blocks along the diagonal and leave the other elements 0. The strategy is to iteratively flank the diagonal matrix, D , with a series of transforms until the dot product of these transforms, on either side, bring D close enough to a diagonal matrix. The inner product of the transforms will then yield the eigenvectors. The transform right multiplying with D is the inverse of the transform left multiplying with D . The resulting matrix is a rotated version of the original D . This result is then flanked again by another transform and its inverse in the same manner and a new rotated D matrix is computed. This process continues until the final result, C_X is a diagonal matrix, within computer precision [7].

There are basically two general approaches to iteratively diagonalizing the matrix, D [7]. Most modern eigensystem algorithms use both in combination for better results. The first approach is to design transformation matrices to zero out a particular off-diagonal element or row/column of D . However, a finite sequence of simple transformation matrices is not generally enough to completely diagonalize a matrix to within machine precision. Therefore, one can iterate the finite sequence of transforms until the off-diagonal values are negligible. Otherwise one can make use of a second approach, the *factorization methods*. In this approach the covariance matrix can be split

into the inner product of a left and right factor [7]. The left factor is the transformation matrix and will satisfy the equation:

$$F_R F_L = F_L^{-1} D F_L \quad (2) ,$$

The matrix diagonalization approach used in this thesis is known as the Jacobi Transformation algorithm. As described above, the simple sequence of transforms is each a rotation designed to zero-out one of the off diagonals. Successive transformations undo previously set zeros, but overall the off-diagonal elements get closer to zero. The Jacobi method is a simple method compared to many others and is more appropriate for matrices not much larger than 10. The computational processing time becomes significantly larger after this.

4.2 Spatial Compression Technique

The result of all the KLT operations was a number of second-stage KLT principle component images and a left-over set of first-stage KLT principle component images that did not get used in the second-stage process. Each of these images was compressed spacially into a form usable for transmission. The compression was a JPEG 2000 standard, which involved converting the principle component images, in a ‘.raw’ format to a ‘.jp2’ format. To do this, each principle component image was loaded into JASC Paint Shop Pro. In the conversion, the number of bytes used to store the contents of the image (file size) is reduced. The lower the file size converted to, the greater the loss of information from the reconstructed images. However, the tradeoff to converting to a small file size is having a larger capacity for transmission other sets of image data. With limited transmission bandwidth, increasing the compression with minimal loss is important.

This thesis is not interested so much in the effect of the degree of compression of an image. The concern was what effect the choice in file number per KLT group, during the first-stage processing, had on image degradation after reconstructed. We therefore used a fixed reference compression rate for this paper. This was done by choosing a reasonable effective coding bit rate, R , of 0.25 bits per pixel.

R is essentially the number of bits used to code a pixel (in this case 8 bits) divided by a compression factor, CR . The compression factor is the total number of pixels in a set of files compressed together divided by the desired number of bytes used to compress these files [6]. The pixels in a set, in our case, are the total number of pixels in all the files within a set of principle component images that underwent a particular KLT operation. Therefore second-stage KLT files were placed in a different set than the utilized first-stage KLT files.

The total number of bytes allocated for the compression was also determined in a particular way. It was not only made up of the sum of the compressed file sizes for every principle component image in a set, but also an overhead file providing all the parameter information used in the particular KLT process. The overhead file had to be transmitted too, because it was necessary for the reconstruction of the images. The left over available bits were distributed among the principle component images in a set.

The distribution of bits was not even, however. It turns out that the degree of compression that can be performed on each file of a set, and maintain a fairly consistent degree of degradation, can be set proportional to the square root of its eigenvalue (or standard deviation) [6]. Therefore, the compressed-file sizes chosen for each principle component was determined under two conditions. Firstly, a file size was dictated by the

images relative standard deviation. Secondly, the sum of the bytes used for the file sizes equaled the total number of bytes available to allow for an effective coding bit rate of 0.25 bpp.

4.3 Reconstruction

The reconstruction process began by spacially decompressing the '.jp2'-converted images back into their '.raw' form. The following steps were essentially the reverse of the spectral compression protocol used. An Inverse KLT (IKLT) was performed on the set of decompressed second-stage KLT principle component images. Again, each set was derived from the grouping of all first, second, third, and fourth principle component images from the first-stage KLT. Once these principle components were reconstructed, they were reordered with the leftover decompressed first-stage KLT principle component images. At this point, a final IKLT was performed on each of the groups of principle component images to yield reconstructed spectral component images.

4.4 Classification

Once spectral images of a scene are obtained, whether directly or through a transmission and reconstruction scheme, the images must be used to provide useful information. The spectral signature of locations in a scene can be used to identify the types of ground cover. There are many methods for such classification, each with its own advantages and disadvantages.

An unsupervised classification approach was applied to the original and reconstructed spectral images. The scene of San Francisco was resolved into 6 distinct classes. To begin the classification process, 6 initial spatial locations were chosen by referencing particular spatial coordinates for 6 distinct features in previous performed

classification on the same spectral images [9]. These same ‘centroids’, each marked by a unique set of reflection intensities, were used as the initial starting points for the classification of all the spectral image sets obtained. All other pixels in the scene were assigned to the class associated with a centroid. This assignment was based on a minimum Euclidean distance criterion [10]. Pixels were assigned to the class of the centroid that they were the minimum, spectral-vector distance from. For the next classification iteration, a new centroid was determined for each class by taking the average of the intensities of each pixel in a class. The pixels were then reclassified in the same manner. The process of reclassifying pixels continued until the sum of squared distances from a centroid to its associated class pixels changed relatively little at the next iteration (see Appendix for code). A root-mean-squared-error (rmse) criterion was used such that when an iteration caused less than a 1.8% change in pixel classification, the process would stop (described in section 4.5.1). In all cases, this occurred in less than 10 iterations. The result of this ‘general hill climbing’ approach was a ‘thematic map’ distinguishing different classes of objects in the scene by different colors.

4.5 Quantitative Assessment of Reconstruction

There have been a number of methods proposed for achieving a suitable measure of image degradation of a decompressed image. A popular theoretical measure is to simply calculate the total root-mean-squared error between the original and reconstructed spectral images. However, for the practical application of ground-cover identification, it is also important to compare error in the classification process via confusion matrices. In this thesis, both were investigated.

4.5.1 Root-mean-squared Error

To measure the degree of degradation of the reconstructed images, each resulting spectral image was compared to its respective original spectral image, pixel by pixel. This was performed for all three types of groupings of spectral images: 20, 16, and 8 files per group. The effectiveness of each of the three types of grouping was compared by looking at their overall percent root-mean-squared error (rmse), with respect to their corresponding original spectral images. The calculations were done using code written in Matlab to speed up the processing (see Appendix). Firstly, the sum of the squared differences between each original and reconstructed pixel pair, for all spectral images of a particular grouping, was determined. The cumulative summing of squared differences was performed over the entire spectrum. This yielded the *total error square*, which was divided by the total number of pixel pairs (the number of pixels in an image, 256x256, times the number of images used, 144). This last calculation gave the *mean square error*. Taking the square root of this value produces the *root-mean-square error (rmse)*. Finally, by dividing the *rmse* by 255 (the largest value that a pixel can be) the error becomes scaled. Multiplying the result by 100 gives the *percent rmse*. We looked at the *percent rmse* for each of the three file size groupings and compared them.

4.5.2 Confusion Matrix

A confusion matrix reveals the percentage of false positives and false negatives in the classification process described earlier. The pixel classification received from the thematic map, obtained using the original spectral images, was treated as the true classification of the scene. The pixel classifications for each reconstruction scenario was compared to each pixel or the true classification. The percentage of correctly identified

pixels in a class (relative to the reconstructed classification) is placed in the right most column in the confusion matrix (Tables 5-1, 5-2, 5-3). These values are the ‘User Accuracy.’ The bottom row of the matrix contains the percentage of correctly classified pixels (relative to the true classification). These values are the ‘Producer Accuracy.’ Finally, the total percentage of correctly identified pixels can be found in the lower right corner of the matrix.

4.6 PC Contribution to Second-stage KLT

In an effort to investigate how much of the first-stage principle component image information is decorrelated in the second-stage KLT operation, the characteristics of the second-stage transformation were looked at. The value of each of the elements in an eigenvector indicates the relative amount of information that each of the first-stage principle component images, used in the transformation, contributes to its single corresponding principle component image received at the second stage. After the first-stage KLT, all of the PCs of the same level were combined (i.e. PC 1 from each group, where 1 is the level number and i value in Figure 4-1). Again, each PC image represented the transformation of a particular grouping of spectral images. An entire set of eigenvectors results for a single equi-level PC, second-stage transformation.

One noted phenomena was that, given a particular equi-level PC transformation, each and every eigenvector, of a single transformation, could all contain a 0 (accurate to 2 significant digits, in this thesis) as their value at a particular component index. This indicated that the first-stage PC image associated with the index value did nothing to contribute to the second-stage decorrelation. In other words, to begin with, information from this first-stage PC image was not correlated with the other first-stage PC images in

its KLT group. This is equivalent to not including the uncorrelated first stage PC image in the calculation of the second stage transformation matrix. Under such scenarios the percentage of contributing first-stage PC images in a transformation group was compared at each PC level (i value in Figure 4-1) and across the grouping scenarios of 8, 16, and 20 files/group. Percentages were calculated by dividing the number of eigenvalue indices that did not contain the value 0 across all eigenvectors by the total possible number of eigenvalue components.

Chapter 5 Results

When the *percent rmse* values between original and reconstructed images were determined, a general trend between file number per spectral grouping and image degradation was observed. At a coding bit rate of 0.25 bpp, as the number of files per grouping was decreased, the overall *percent rmse* decreased (Figure 5-1).

Figure 5-1 Reducing the number of spectral images used per group in 1st stage KLT operations had a general effect of reducing the degradation of the reconstructed image.

These quantitative results were consistent with the qualitative results as well. By inspection of the thematic maps obtained from the classification process, one can see a trend in image resolution and accuracy. The thematic map constructed from the original

set of spectral images showed very distinct features of the scene of San Francisco (Figure 5-2). Comparing this to the thematic map generated from the reconstructed spectral

Figure 5-2 Ground Truth Thematic Map. The thematic map constructed from the original spectral images shows distinct features of San Francisco.

images of the 8 file/group compression scheme showed some minor changes in the shape of the class features (Figure 5-3). However, major trends in feature shape were not lost.

The 16 file/group thematic map had much more obvious deviations from the original

Figure 5-3 8 File/group Thematic Map. The thematic map constructed from the 8 file/group spectral images shows minor losses in feature shapes compared to the original.

thematic map (Figure 5-4). Much of the resolution of the features marked in yellow and teal had degraded. In addition to poor resolution, one could see loss in the major contours of the features for a 20 file/group scenario (Figure 5-5).

Figure 5-4 16 File/group Thematic Map. The thematic map constructed from the 16 file/group spectral images shows some evident resolution loss of features compared to the original.

Figure 5-5 20 File/group Thematic Map. The thematic map constructed from the 20 file/group spectral images shows some evident contour loss of features compared to the original.

Looking more closely at the classification results via the associated confusion matrix, showed a minor trend in User and Producer Accuracy. A more evident trend was found in the overall accuracy. The likelihood of getting classification false positives and false negatives appeared to increase a bit in general as more files/group are used in the compression process (Tables 5-1, 5-2, 5-3). But the number of correctly identified pixels did not seem to necessarily increase for a given class as the number of files/group

decreases. However, in general, the number of total correctly identified pixels seemed more likely to increase with the decrease in file/group number.

Class	1	2	3	4	5	6	<i>Row Sum</i>	<i>User Accuracy (%)</i>
1	6026	0	0	0	23	116	6165	97.75
2		20480	24	23	581	0	21108	97.02
3	0	20	3885	89	0	0	3994	97.27
4	0	844	13	10176	0	0	11033	92.23
5	74	70	0	0	9776	12	9932	98.43
6	116	0	0	0	5	13183	13304	99.09
<i>Col. Sum</i>	6216	21414	3922	10288	10385	13311	65536	
<i>Producer Accuracy (%)</i>	96.94	95.64	99.06	97.99	94.14	99.04		96.93

Table 5-1 8 File/group Confusion Matrix. The confusion matrix for a thematic map comparison of an 8 file/group compression to the original spectral images. One can see the degree of false positives and false negatives for each class.

Class	1	2	3	4	5	6	<i>Row Sum</i>	<i>User Accuracy (%)</i>
1	6117	0	0	0	224	44	6385	95.80
2	0	19555	2	2610	0	0	22167	88.22
3	0	150	3468	36	0	0	3654	94.91
4	0	0	452	7642	0	0	8094	94.42
5	0	1709	0	0	10138	0	11847	85.57
6	99	0	0	0	23	13267	13389	99.09
<i>Col. Sum</i>	6216	21414	3922	10288	10385	13311	65536	
<i>Producer Accuracy (%)</i>	98.41	91.32	88.42	73.59	97.62	99.67		91.84

Table 5-2 16 File/group Confusion Matrix. The confusion matrix for a thematic map comparison of a 16 file/group compression to the original spectral images. One can see the degree of false positives and false negatives for each class.

Class	1	2	3	4	5	6	<i>Row Sum</i>	<i>User Accuracy (%)</i>
1	5923	0	0	0	284	545	6752	87.72
2	0	18580	171	1235	1110	0	21096	88.07
3	0	60	3673	102	0	0	3835	95.78
4	0	1521	78	8950	0	0	10549	84.84
5	152	1253	0	1	8967	78	10451	85.80
6	141	0	0	0	24	12688	12853	98.72
<i>Col. Sum</i>	6216	21414	3922	10288	10385	13311	65536	
<i>Producer Accuracy (%)</i>	95.29	86.77	93.65	86.18	86.35	95.32		89.69

Table 5-3 20 File/group Confusion Matrix. The confusion matrix for a thematic map comparison of a 20 file/group compression to the original spectral images. One can see the degree of false positives and false negatives for each class.

The results suggested that one can optimize image reconstruction and feature identification through a change in file number per group. Additionally, second-stage KLT results suggested an optimization in number and choice of first-stage PC images for inter-group decorrelation. Second-stage KLT was performed on the collection of the first 4 PC images of the first-stage. Inspection of the eigenvector components of the second-stage KLT showed situations in which no significant PC image decorrelation occurred for particular PC groupings after the first stage. The percentage of PC images that were used in the second-staged KLT were noted for each compression scheme. Results showed that, in every case of choice in file number per group, fewer first-stage PC images contributed to second-stage decorrelation (Figure 5-6). Furthermore, over a 4-PC-image-deep second-stage transformation, the percentage of first-stage PC image contribution dropped off faster the fewer the number of spectral images that were used per group. The actual PC images that did not contribute were also identifiable from the set of eigenvectors by noting the index value of the eigenvectors, which is associated with PC level.

Figure 5-6 Second-stage Decorrelation Analysis. The percentage of combined equi-level PC images, from the first-stage KLT, that contribute in the second stage KLTs, decreases with the depth of PC level used. Furthermore, the fewer spectral images used per group, the faster the drop-off in contributing first-stage PC images with PC depth.

Chapter 6 Conclusion and Future Work

The proposed 2-stage KLT method addresses a number of practical limitations in transmitting a large collection of spectral images. For instance, in some cases when a large spectrum of hyperspectral images must be captured at once, multiple independent terminals must be used. Furthermore, the processing may also have to be done through separate terminals. Therefore, the compression technique has to be designed to perform many *local* transformations without misrepresenting the areas of spectral correlation. At the same time, with many images to be processed, it is more practical to rely on a method that can treat the spectral images as groups of correlated images rather than individually. However, the grouping should be as methodical as possible, to maximize the accuracy of the reconstructed image, while still keeping the protocol simple.

The compression approach used in this project acknowledges the limitations of processing a large set of images simultaneously. While there are other simpler techniques that perform the transformation, like the *bidirectional interband prediction* method, they are not always so straightforward and can require too much specialized treatment to each band [3]. This is not always worthwhile when dealing with a large number of bands. The application of the *local* KLT would therefore, more appropriate for a study like this one. However, even though the *local* KLT finds an optimal transformation matrix for a single spectral partition, it requires a single transformation operation of all the rest of the spectral images at once. This operation can be computationally inefficient, especially if the number of other spectral images is very large. The two-stage KLT gets around this by partitioning the spectral images into more than two partitions, making the transformation operations less computationally demanding. The *local* KLT may actually produce better

reconstruction than the two-stage KLT because it focuses on finding optimum transformation parameters to produce minimum error. But this computational approach seems to be a much larger price to pay than the proposed method. The *local* KLT does not do much to elevate the biggest criticism of the KLT algorithm – the computational burden. The *distributed* KLT makes even more dramatic computational efforts into further optimizing reconstruction [5]. This can be highly impractical when the most critical factor is addressing interband correlation, which the two-stage KLT does.

Recombining groups of independent principle component images at many KLT stages would prove very effective and would ensure benefits from all correlations throughout the spectra [8]. The computation is also much more straightforward than the *distributed* KLT. However, if the number of files per group is small and/or the number of groups is large, the processing power necessary to address the less significant correlations at the later stages, may not make up for the benefits. This is precisely what the *sub-optimal* KLT fails to address. A two-stage KLT, may be enough to capture the significant spectral correlations. Like the *sub-optimal* KLT, the two-stage method used an arbitrary number of files per spectral grouping. However, unlike the *sub-optimal* KLT, this paper further provides some rational on the side effects and tradeoffs with a choice in file number per group. Such information can be used to optimize file number per group choice.

In general, it appears that decreasing the number of spectral images used for the first-stage local KLTs yields overall less image degradation and classification inaccuracies. This also means more KLT operations at the first stage. However, to get full decorrelation between groups, fewer PC images, and possibly fewer PC depth levels, are

required at the second-stage. The *sub-optimal* KLT that was described uses a uniform PC depth at every KLT stage. However, depending on the contribution distribution of PC images to later KLT stages, the contribution of a fewer number of images or PC levels may be necessary to yield comparable decorrelation results. In other cases, the number of PC images and levels needed to completely decorrelate images at later stages, may be insufficient with an arbitrary *suboptimal* KLT file per group number. One can provide this rational to optimize the depth of decorrelated second-stage PCs on the data generated from this paper, while using the scenario yielding the best reconstruction and classification.

For the experiment described in this thesis, one would want to use 8 files/group, but will require at least 4 levels deep in second-stage PC decorrelation to get at least roughly 10% first-stage PC contribution to the second stage. The reason for this is that at the fourth PC level already roughly only 10% of the first-stage PC images contribute (Figure 5-6). In fact, from the inspection of the eigenvectors one can actually select the particular PC images needed for the second KLT. Again, this is all determined by looking at the values of the eigenvectors of the second-stage transformation. 10% at the fourth PC level means that one tenth of the collected PC 4 images from the first-stage KLT were correlated with the rest of the PC 4 images. In other words, across all the eigenvectors for the particular second-stage transformation, one-tenth of the eigenvector components contained non-zero values. Therefore, one can get the same results by simply removing the PC 4 images whose corresponding eigenvector indices are 0, and then perform the transformation again. In this way computational efforts can be saved.

The results of my experiments indicated at the smallest file/group number, 8, only needed PC 1 through PC 4 to almost completely decorrelate the images. In fact, was this choice in file/group number and PC depth of second-stage compression that allowed the quantitative goals, described in the introduction, to be met. Again, one quantitative goal was to compress the 144, 256x256 pixel spectral images at 0.25 bpp such that the reconstructed images exhibited a root-mean-squared error of less than 1%. With 8 files/group a root-mean-squared error of 0.29 %. The second quantitative goal was to receive a total classification accuracy of greater than 95%. A 6-class, unsupervised minimum Euclidean distance, general hill climb classification scheme was considered. Iterations were set to stop after less than a 1.8% change in pixel classification was achieved. Again, this second goal was achieved by using 8 files/group to yield a total classification accuracy of 97.93%. Therefore all the quantitative goals of this thesis were met.

However, this does not suggest that, in every compression scenario, optimal reconstruction will occur with 8 files/group and a PC depth of 4. Results may vary with choice in image, the resolution of the spectral imager used, as well as changes in test parameters (i.e. a use of 0.25 bpp and 1.8% pixel change in the general hill climb method). My results have merely suggested that minimizing file number per group seems to yield more accurate reconstruction results. My results also indicate that the PC depth used at second-stage KLT is a function of the file number per group used and that this depth is determinable by inspection.

To explain the apparent relationship between file/group number and depth of second-stage PC image decorrelation, a hypothesis is proposed. It appears that the fewer

the number of images in a group used at the first stage, the more accurate the linear approximation of the KLT becomes. Assuming a linear relationship between spectral dynamics is a bold statement to make, but with fewer data samples simpler models, such as a linear relationship, may be fairly accurate. This could explain the overall lowest rmse results with 8 files/group.

However, one can expect that within a very narrow spectral range, few dynamics are present. But over a larger range, such as a spectral group containing 20 files, many major spectral dynamic relationships may be present. Therefore the probability of any two spectral groups of 20 images sharing the same dynamics at a given PC level is higher than say groupings of 8 spectral images. This could explain why with the higher files/group number much more second-stage decorrelation at lower PC depths was taking place. Again, at a PC level of 4 almost all images were uncorrelated when using 8 files/group. This was probably because not much dynamic information could be found at this level over such a small spectral range. However, with 16 files/group and 20 files/group, much more decorrelation was possible beyond a PC depth of 4 at the second stage. Exploiting this apparent relationship between file/group number and required PC level decorrelation at the second stage of KLT operations is what makes this method so advantageous.

Although, the two-stage KLT method, introduced here, has some advantages over existing compression techniques, it does have some of its own limitations as well. The major drawback on this compression methodology stems from the fact that at the second stage KLT only PC images from the same depth level are combined. This falsely makes the assumption that all the dynamics captured at one PC level can only be contained in

the same PC level for other spectral group ranges. The *distributed-KLT* method takes this into account by including the contribution of every set of transformed images to every other set of transformed spectral images.

In addition to a lack of correlation optimization, the described compression scheme does not propose an optimal file per group number to be used. The number of files chosen per group was arbitrary. It simply depends on the number of files one can afford to process at each stage and how accurate a reconstruction one wishes to get. For the most part, it was noticed that reducing the number of files used for the first-stage KLT acted to improve the reconstruction results overall. However, as was observed, this is not true at the local spectral range level or the local class assignment level. In conclusion, the 2-stage KLT method is useful when a *direct* KLT on all spectral images is either impractical or not possible. It caters itself to scenarios where the number of KLT operations must also be minimized but where exploiting interband correlation is important. One can use the trade offs, shown, among number of KLT operations, number of files undergoing a KLT operation, and desired reconstructed image accuracy, to determine one's ideal layout for his/her 2-stage KLT approach. As was shown, it is possible to achieve less than 1% rmse and greater than 95% total classification accuracy.

Future work could look deeper into the spectral grouping preferences in particular spectral ranges. This might develop into a methodology that could be used in determining an optimal set of file-number groups to be used for any set of spectral images. One could further write an efficient algorithm that compares *mean squared errors* at the image level instead of the group level, in order to further reduce the total *percent rmse*. It is also worth trying a different KLT algorithm. An algorithm that makes use of the *factorization*

method may prove to be more accurate at determining transformation parameters [7]. An investigation of the spacial compression algorithm could also be done. In particular, one could compare the use of JPEG2000 in this paper to that of simply applying the recently introduced JP3D, the 3D extension of JPEG2000.

JP3D is the new standard for spacial compression of 3D data. It has particularly useful applications in such fields as medical imaging, where storage of large volumetric data sets is necessary. The JP3D specifications essentially do not introduce new coding concepts but simply extend those of JPEG2000 parts 1 & 2 (embedded block coding by optimized truncation (EBCOT) based on the principles of layered zero coding (LZC)) to a 3rd dimension [11]. The JP3D codec first partitions the volumetric data set (typically a stack of 2D images) into tiles of volumetric blocks that are independently encoded as separate grayscale datasets. All the tile sample coefficients are then filtered via a discrete wavelet transform, using the Mallat decomposition pattern [11]. From here, the codec partitions the resulting sub-bands into dyadically-sized code-blocks, independently encoded by the EBCOT coder (Figure 6-1). The code-block coefficients are processed

Figure 6-1 Partitioning and Encoding in JP3D. After a 3D data set is subdivided, each subdivision is broken down into code-blocks, which are encoded independently [11].

four adjacent bits at a time within a bit plane, before moving down to the next layer

(Figure 6-2). The bit plane pattern is identical to that used in JPEG2000 part 1 [11].

Figure 6-2 JP3D Scanning Pattern. 3D datasets are processed along bit planes and from one slice to the next [11].

The JP3D compression scheme could also be applied to spectral images used in remote sensing, where the 3rd axis could simply be replaced by the spectral information of a 2D scene. The algorithm does have a conceivable application advantages over the 2-stage KLT. Because JP3D uses a wavelet transform approach in all three dimensions, as opposed to a linear transform, such as the KLT, it is more suitable for transmitting data a few lines at a time [11]. Our approach does not have this kind of scan-based-mode flexibility over the spectral dimension. It requires the collection of statistics over the whole image and selects spectral ranges before transmission can take place.

However, as a hyperspectral imaging compression algorithm, JP3D seems to have some computational disadvantages and could possibly be undesirable as a compression model compared to the proposed approach. JP3D's process of partitioning, sub-partitioning, and then filtering at the tile sample coefficient level, can be very involved [11]. If spectral linearity can be assumed, computation can be greatly, and sufficiently, simplified. This simplification is the aim of the 2-stage KLT approach. However, adequate, full spectrum, yet intelligent, decorrelation is also the focus of the proposed approach. JP3D makes efforts to perform decorrelation within and between every bit plane [11]. However, the information provided by the 2-stage KLT justifies the needlessness of attempting decorrelation between select spectral planes from different spectral sets. Therefore, unlike JP3D, the 2-stage KLT avoids unnecessary efforts to further compress an image. It should also be pointed out that, just in terms of transform models, it has been shown that using a KLT in the spectral dimension, followed by a JPEG 2000 discrete wavelet transform (DWT) in the spacial dimension, yields superior rate distortion performance than a JPEG 2000-based DWT in all 3 dimensions [12]. This

suggests that the 2-stage KLT, which makes use of KLT and the JPEG 2000 DWT, may be, in general, a more desirable model. It has also been proposed that application of JP3D may be ineffective at fully addressing issues specific to hyperspectral imagery [13].

BIBLIOGRAPHY

- [1] Saghri, John. "Principle Component Analysis: Lecture Materials 10." California Polytechnic State University, San Luis Obispo. Spring 2008.
- [2] Penna, B. "Transform Coding Techniques for Lossy Hyperspectral Data Compression." IEEE Transaction on GeoScience and Remote Sensing Vol. 34, NO. 5, May 2007.
- [3] Ashok, Rao. "Multispectral Data compression Using Bidirectional Interband Prediction." IEEE Transactions of Geoscience and Remote Sensing Vol. 34, NO. 2, March 1996.
- [4] Shlens, Jonathon. "A Tutorial on Principle Component Analysis." Systems Neurobiology Laboratory, Stalk Institute of Biological Studies. Version 2, Dec. 2005.
- [5] Gastpar, Michail. "The Distributed Karhunen-Loeve Transform." IEEE Transactions on Inforamation Theory Vol. 52, No. 12, Dec. 2006.
- [6] Saghri, John. "Laboratory Experiment 5: Bandwidth Compression of Multispectral Imagery Data." California Polytechnic State University, San Luis Obispo. June 2008.
- [7] Press, William. Numerical Recipes: The Art of Scientific Computing. New Rochelle: Cambridge Press, 1988.
- [8] Wongsawat, Yodchanan. "Integer Sub-optimal Karhunen-Loeve Transform for Multi-channel Lossless EEG Compression." University of Texas at Arlington. 2004.
- [9] Saghri, J. , Tescher, A. , Planinac, A. "Band Compression of Hyperspectral Imagery Data using a Simplified KLT/JPEG 2000 Approach." Proceedings of SPIE, vol. 558, no. 68, August 6, 2004.
- [10] Saghri, John. "Classification: Lecture Materials 5." California Polytechnic State University, San Luis Obispo. Spring 2008.
- [11] Bruylants, Tim. "Volumetric Image Compression with JPEG 2000." SPIE. 2007.
- [12] Du, Q. "Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis." IEEE Geoscience and Remote Sensing Letters 4(2), 201-205, April 2007.
- [13] Pickering, M. "An Architecture for the Compression of Hyperspectral Imagery." In G. Motta, F. Rizzo, and J.A. Storer (Eds.). Hyperspectral Image Compression. Chapter 1, pp. 1-34. New York: Springer, 2006.

APPENDIX

The Karhunen-Loeve Transformation

Concept

Suppose we have a picture of a scene and the picture is separated into its spectral images. Each spectral image contains pixel values indicating the intensities of the ground-cover reflections of particular a spectral band (i.e. the Visible Red wavelength content in the scene). Each spectral image can also be thought of as images taken of an event by a single camera from a particular reference angle in space. Let's use an example to explain this [4].

Assume the scene, or event, that we are observing is a ball on a spring, moving back and forth along one dimension in space. We observed the position of the ball in space over time using cameras taken from various angles, relative to the direction of motion of the ball (Figure A-1). The assumption that the KLT method makes is that the

Figure A-1 KLT Conceptual Example. Spectral images of a scene can be thought of as taking pictures of an event, such as a moving ball on a spring, where each image is taken from a different perspective [4].

observations taken by the camera, in this case, position relative to the 2D plane of the camera lens, is a movement with displacement in one linear dimension, but within a multiple dimensional space. Each camera can only detect the components of the x , y , z directions of movement of the ball in space, relative to the positioning of its 2D lens. Further, each camera has the chance that it can detect data (movement) along any combination of the fixed, reference dimensions x , y , and z . A camera's lens could perchance be positioned so that it is perpendicular to the x axis. It therefore would not detect movement of the ball, moving in the x direction. If the ball is moving along only one dimension then we need at least one camera to fully observe it, but facing a plane along the direction of movement.

Now suppose the scene contains three simultaneous events, where each event is a ball moving along one of the three orthogonal directions. If we want to fully describe the events of three balls moving in mutually orthogonal directions then we need at least three cameras, all of which are orthogonal to each other and provide unrelated data. One, however, does not know how many dimensions we are able to detect from each camera, as this is dependent on the direction from which the camera is observing the events. However, if the mapped data of a camera moves with high correlation to the mapped data of another camera, then both cameras are most likely detecting the same event of movement along one dimension. So if two cameras are being used to capture the same event and produce separate images, one can simply combine the two images into one and get rid of any redundant movement information. This is exactly what the KLT operation does. It takes highly correlated spectral image information and combines it, effectively reducing the number of images needed to describe an event. But if the scene contains multiple independent events, which are changing along different, orthogonal dimensions, one would need to separate all the compressed camera image data into new images that each capture a different, uncorrelated event. In the case of three orthogonal-moving balls, we would combine and partition the data collected by multiple cameras into three principle component images, each describing a different ball's, one-dimensional motion.

Now that we have a better idea of how the KLT operation works, we can tie our example into the application of ground-scene image compression. Think of a single camera recording in time, from our example, as being homologous to a single spectral image pixel value. A single camera captures many positions of multiple balls in the scene. Likewise, a single spectral image band captures many pixel intensities of multiple

reflection events from ground features. Therefore, choosing a particular point in time whose camera image data you want to look at is analogous to choosing a particular pixel whose spectral value you want to look at.

Derivation

The first step in converting a series of spectral images into a set of principle component images, via the KLT method, is to determine how related spectral images are. This is done by finding the covariance between every combination of spectral values for all the pixels in the image [4]. We begin by defining a matrix, $\mathbf{X}$. Every column in this matrix consists of the reflection intensities of each spectral image for a particular pixel. There are n rows for each of the n pixels in the scene image. We can define the covariance matrix as:

$$\mathbf{C}_X = (1/(n-1))\mathbf{X}\mathbf{X}^T .$$

Our goal is to convert every spectral image into an equal set of principle component images. Essentially, we are recombining the spectral information into images of uncorrelated data, each of which describes a different, independent spectral trend in the scene. However, not every principle component image is capturing a very significant trend. The new derived principle component images will be defined as the matrix, $\mathbf{Y}$, structured in the same way as $\mathbf{X}$. We assume that any spectral image pixel value may be linearly related to any other spectral image value, such that the spectral measurements are orthogonal to one another. If we think of the spectral values as measurements along orthogonal axes of trends in the scene, we can rotate the axes such that a set of linearly correlated spectral data will produce the largest variance along its new axis (Figure A-2).

Figure A-2 Effect of Spectral Axis Rotation on Data Variance. If we assume that the measurements of the spectral bands are coordinates of a linear relationship between sets of bands, then the spectral band axes can be rotated. The closer one rotates an axis toward a data trend, the larger the variance along the axis becomes. Likewise, the accuracy of the measurement of the trend will increase, therefore the signal-to-noise ratio will increase [4].

The movement of data of a particular trend is expected to move along this particular new axis. Therefore, the further one's axis of measurement is rotated from this principle axis, the more inaccurate (more noisy) the measurement of the trend's signal along the axis becomes. The set of values along the new, rotated axes are the set of values provided in each new principle component image. This axis rotation process and subsequent transformation of pixel values can be expressed as:

$$\mathbf{Y} = \mathbf{P}\mathbf{X},$$

where the rows of $\mathbf{P}$ are the *principle components* of $\mathbf{X}$ that map the pixel spectral values to a new set of values.

If we were to have our new principle component images be uncorrelated with each other, and each provide an independent set of trends in the scene, then the

covariances between the data, mapped to the principle component axes, should be zero. At the same time, the new data mapped to each particular principle component should contain a non-zero variance. Therefore, the goal of the KLT operation is to provide a covariance matrix for $\mathbf{Y}$ whose off-diagonal values are zero and whose diagonal values are non-zero. In theory, this can be done and we show it through some basic principles in linear algebra.

Firstly, as with $\mathbf{X}$, we can define the covariance matrix for $\mathbf{Y}$ as:

$$\mathbf{C}_Y = (1/(n-1))\mathbf{Y}\mathbf{Y}^T .$$

By the identity provided earlier, we can rewrite this equation and come to the following conclusion:

$$\begin{aligned}\mathbf{C}_Y &= (1/(n-1))(\mathbf{P}\mathbf{X})(\mathbf{P}\mathbf{X})^T \\ &= (1/(n-1))\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T \\ &= (1/(n-1))\mathbf{P}(\mathbf{X}\mathbf{X}^T)\mathbf{P}^T \\ &= (1/(n-1))\mathbf{P}\mathbf{C}_X\mathbf{P}^T .\end{aligned}$$

Because $\mathbf{C}_X$ is a symmetric matrix, it can be diagonalized by an orthogonal matrix of its eigenvectors:

$$\mathbf{C}_X = \mathbf{E}\mathbf{D}\mathbf{E}^T ,$$

where $\mathbf{E}$ is the matrix of eigenvectors arranged by columns and $\mathbf{D}$ is a diagonal matrix.

We also note that the inverse of the orthogonal matrix, $\mathbf{P}$, is its transpose. With these observations in mind, we can further rewrite the equation for $\mathbf{C}_Y$:

$$\begin{aligned}\mathbf{C}_Y &= (1/(n-1))\mathbf{P}\mathbf{C}_X\mathbf{P}^T \\ &= (1/(n-1))\mathbf{P}(\mathbf{P}^T\mathbf{D}\mathbf{P})\mathbf{P}^T \\ &= (1/(n-1))(\mathbf{P}\mathbf{P}^T)\mathbf{D}(\mathbf{P}\mathbf{P}^T)\end{aligned}$$

$$= (1/(n-1))(\mathbf{P}\mathbf{P}^{-1})\mathbf{D}(\mathbf{P}\mathbf{P}^{-1})$$

$$\mathbf{C}_Y = (1/(n-1))\mathbf{D}.$$

It is clear from this reduction that in order to reach our goal of diagonalizing $\mathbf{C}_Y$ we must find the appropriate choice of $\mathbf{P}$. As one will noticed from the way one comes up with this final equation, the rows of $\mathbf{P}$ will be the eigenvectors. The associated eigenvalues will be the diagonal values of $\mathbf{C}_Y$.

For every eigenvector and associated eigenvalue we have a respective principle component image. The larger the eigenvalue in a set, the more information content that its respective principle component image contains of the scene. However, 95% of the information content of all the spectral data resides in the first two or three spectral component images [1]. Therefore, after such a compression method, often only the first three principle component images are needed to reconstruct the original spectral images without significant information loss.

Matlab Code

% Calculation of percentage root mean squared errors for each of the
% spectral images (simplified for debugging)

```
o1=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M97.raw',256,256);
o2=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M98.raw',256,256);
o3=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M99.raw',256,256);
o4=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M100.raw',256,256);
o5=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M101.raw',256,256);
o6=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M102.raw',256,256);
o7=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M103.raw',256,256);
o8=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M104.raw',256,256);
o9=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M121.raw',256,256);
o10=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M122.raw',256,256
);
o11=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M123.raw',256,256
);
o12=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M124.raw',256,256
);
o13=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M125.raw',256,256
);
o14=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M126.raw',256,256
);
o15=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M127.raw',256,256
);
o16=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M128.raw',256,256
);
%o17=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M17.raw',256,25
6);
%o18=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M18.raw',256,25
6);
%o19=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M19.raw',256,25
6);
%o20=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M20.raw',256,25
6);

r1=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M97R.raw',256,256);
r2=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M98R.raw',256,256);
r3=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M99R.raw',256,256);
r4=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M100R.raw',256,256
);
r5=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M101R.raw',256,256
);
r6=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M102R.raw',256,256
);
```

```

r7=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M103R.raw',256,256
);
r8=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M104R.raw',256,256
);
r9=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M105R.raw',256,256
);
r10=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M106R.raw',256,25
6);
r11=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M107R.raw',256,25
6);
r12=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M108R.raw',256,25
6);
r13=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M109R.raw',256,25
6);
r14=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M110R.raw',256,25
6);
r15=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M127R.raw',256,25
6);
r16=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M128R.raw',256,25
6);
%r17=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M17R.raw',256,2
56);
%r18=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M18R.raw',256,2
56);
%r19=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M19R.raw',256,2
56);
%r20=ReadRawImage('C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M20R.raw',256,2
56);

rmse1_matrx=cat(3,o1,r1,zeros(256,256));
rmse2_matrx=cat(3,o2,r2,zeros(256,256));
rmse3_matrx=cat(3,o3,r3,zeros(256,256));
rmse4_matrx=cat(3,o4,r4,zeros(256,256));
rmse5_matrx=cat(3,o5,r5,zeros(256,256));
rmse6_matrx=cat(3,o6,r6,zeros(256,256));
rmse7_matrx=cat(3,o7,r7,zeros(256,256));
rmse8_matrx=cat(3,o8,r8,zeros(256,256));
rmse9_matrx=cat(3,o9,r9,zeros(256,256));
rmse10_matrx=cat(3,o10,r10,zeros(256,256));
rmse11_matrx=cat(3,o11,r11,zeros(256,256));
rmse12_matrx=cat(3,o12,r12,zeros(256,256));
rmse13_matrx=cat(3,o13,r13,zeros(256,256));
rmse14_matrx=cat(3,o14,r14,zeros(256,256));
rmse15_matrx=cat(3,o15,r15,zeros(256,256));
rmse16_matrx=cat(3,o16,r16,zeros(256,256));
%rmse17_matrx=cat(3,o17,r17,zeros(256,256));

```

```

%rmse18_matrx=cat(3,o18,r18,zeros(256,256));
%rmse19_matrx=cat(3,o19,r19,zeros(256,256));
%rmse20_matrx=cat(3,o20,r20,zeros(256,256));

for i=1:256
    for j=1:256
        rmse1_matrx(i,j,3) = (rmse1_matrx(i,j,2) - rmse1_matrx(i,j,1))^2;
        rmse2_matrx(i,j,3) = (rmse2_matrx(i,j,2) - rmse2_matrx(i,j,1))^2;
        rmse3_matrx(i,j,3) = (rmse3_matrx(i,j,2) - rmse3_matrx(i,j,1))^2;
        rmse4_matrx(i,j,3) = (rmse4_matrx(i,j,2) - rmse4_matrx(i,j,1))^2;
        rmse5_matrx(i,j,3) = (rmse5_matrx(i,j,2) - rmse5_matrx(i,j,1))^2;
        rmse6_matrx(i,j,3) = (rmse6_matrx(i,j,2) - rmse6_matrx(i,j,1))^2;
        rmse7_matrx(i,j,3) = (rmse7_matrx(i,j,2) - rmse7_matrx(i,j,1))^2;
        rmse8_matrx(i,j,3) = (rmse8_matrx(i,j,2) - rmse8_matrx(i,j,1))^2;
        rmse9_matrx(i,j,3) = (rmse9_matrx(i,j,2) - rmse9_matrx(i,j,1))^2;
        rmse10_matrx(i,j,3) = (rmse10_matrx(i,j,2) - rmse10_matrx(i,j,1))^2;
        rmse11_matrx(i,j,3) = (rmse11_matrx(i,j,2) - rmse11_matrx(i,j,1))^2;
        rmse12_matrx(i,j,3) = (rmse12_matrx(i,j,2) - rmse12_matrx(i,j,1))^2;
        rmse13_matrx(i,j,3) = (rmse13_matrx(i,j,2) - rmse13_matrx(i,j,1))^2;
        rmse14_matrx(i,j,3) = (rmse14_matrx(i,j,2) - rmse14_matrx(i,j,1))^2;
        rmse15_matrx(i,j,3) = (rmse15_matrx(i,j,2) - rmse15_matrx(i,j,1))^2;
        rmse16_matrx(i,j,3) = (rmse16_matrx(i,j,2) - rmse16_matrx(i,j,1))^2;
        %rmse17_matrx(i,j,3) = (rmse17_matrx(i,j,2) - rmse17_matrx(i,j,1))^2;
        %rmse18_matrx(i,j,3) = (rmse18_matrx(i,j,2) - rmse18_matrx(i,j,1))^2;
        %rmse19_matrx(i,j,3) = (rmse19_matrx(i,j,2) - rmse19_matrx(i,j,1))^2;
        %rmse20_matrx(i,j,3) = (rmse20_matrx(i,j,2) - rmse20_matrx(i,j,1))^2;
    end;
end;

sumse1 = sum(sum(rmse1_matrx(:, :, 3)));
sumse2 = sum(sum(rmse2_matrx(:, :, 3)));
sumse3 = sum(sum(rmse3_matrx(:, :, 3)));
sumse4 = sum(sum(rmse4_matrx(:, :, 3)));
sumse5 = sum(sum(rmse5_matrx(:, :, 3)));
sumse6 = sum(sum(rmse6_matrx(:, :, 3)));
sumse7 = sum(sum(rmse7_matrx(:, :, 3)));
sumse8 = sum(sum(rmse8_matrx(:, :, 3)));
sumse9 = sum(sum(rmse9_matrx(:, :, 3)));
sumse10 = sum(sum(rmse10_matrx(:, :, 3)));
sumse11 = sum(sum(rmse11_matrx(:, :, 3)));
sumse12 = sum(sum(rmse12_matrx(:, :, 3)));
sumse13 = sum(sum(rmse13_matrx(:, :, 3)));
sumse14 = sum(sum(rmse14_matrx(:, :, 3)));
sumse15 = sum(sum(rmse15_matrx(:, :, 3)));
sumse16 = sum(sum(rmse16_matrx(:, :, 3)));
%sumse17 = sum(sum(rmse17_matrx(:, :, 3)));

```



```

%sumse18 = sum(sum(rmse18_matrx(:,3)));
%sumse19 = sum(sum(rmse19_matrx(:,3)));
%sumse20 = sum(sum(rmse20_matrx(:,3)));

totsumse = sumse1+sumse2+sumse3+sumse4...
    +sumse5+sumse6+sumse7+sumse8 ...
    +sumse9+sumse10+sumse11+sumse12+sumse13+sumse14+sumse15+sumse16; % ...
    %+sumse17+sumse18+sumse19+sumse20;
totsumse
%tmse = totsumse/(256*256*16);
%trmse = sqrt(tmse);
%prmse = (100/255)*trmse

% Calculation of PC compressed file sizes

%stot = 114688; %CR = 16;

%oh = 4620;
%cvar1 = 1039.48;
%cvar2 = 162.40;
%cvar3 = 34.64;
%cvar4 = 4.46;
%cvar5 = 2.61;
%cvar6 = 1.73;
%cvar7 = 1.01;

%x1 = (stot-
    oh)/(1+sqrt(cvar2/cvar1)+sqrt(cvar3/cvar1)+sqrt(cvar4/cvar1)+sqrt(cvar5/cvar1)+
    sqrt(cvar6/cvar1)+sqrt(cvar7/cvar1))
%x2 = (stot-
    oh)/(1+sqrt(cvar1/cvar2)+sqrt(cvar3/cvar2)+sqrt(cvar4/cvar2)+sqrt(cvar5/cvar2)+
    sqrt(cvar6/cvar2)+sqrt(cvar7/cvar2))
%x3 = (stot-
    oh)/(1+sqrt(cvar1/cvar3)+sqrt(cvar2/cvar3)+sqrt(cvar4/cvar3)+sqrt(cvar5/cvar3)+
    sqrt(cvar6/cvar3)+sqrt(cvar7/cvar3))
%x4 = (stot-
    oh)/(1+sqrt(cvar1/cvar4)+sqrt(cvar2/cvar4)+sqrt(cvar3/cvar4)+sqrt(cvar5/cvar4)+
    sqrt(cvar6/cvar4)+sqrt(cvar7/cvar4))
%x5 = (stot-
    oh)/(1+sqrt(cvar1/cvar5)+sqrt(cvar2/cvar5)+sqrt(cvar3/cvar5)+sqrt(cvar4/cvar5)+
    sqrt(cvar6/cvar5)+sqrt(cvar7/cvar5))
%x6 = (stot-
    oh)/(1+sqrt(cvar1/cvar6)+sqrt(cvar2/cvar6)+sqrt(cvar3/cvar6)+sqrt(cvar4/cvar6)+
    sqrt(cvar5/cvar6)+sqrt(cvar7/cvar6))
%x7 =

```

```

% (stot-
oh)/(1+sqrt(cvar1/cvar7)+sqrt(cvar2/cvar7)+sqrt(cvar3/cvar7)+sqrt(cvar4/cvar7)+
sqrt(cvar5/cvar7)+sqrt(cvar6/cvar1))

```

```

function x = ReadRawImage(filename,width,height)
% Read a RAW DATA format image written by IPLAB.
% The raw image data is written as a series of unsigned
% eight-bit integers in row first order.

```

```

% Open file
fid = fopen(filename);

% Read in the data.
x = fread(fid, [width,height], 'uint8');

% Close file
fclose(fid);

```

```

function x = ReadRawImage2(filename, width, height)
% Read a RAW DATA format image written by IPLAB.
% The raw image data is written as a series of unsigned
% eight-bit integers in row first order.

```

```

% Open file
id = fopen(filename,'r');

% Read in the data.
t = fread(id, [width,height], 'uint8');

%x=uint8(zeros(height,width));

x = double(t);

%for q=1:height
%   for w=1:width
%       x(q,w)=double(t(q,w));
%   end
%end

% Close file
fclose(id);

```

```

% This function takes in spectral images and classifies pixels into one of 6
% classes via an unsupervised minimum Euclidean distance and
% General Hill Climb approach. The result is a thematic map.
function themmap = isodata_mindist2(files,rws,clmns,M)

```

```

M(:,:,files+1)=zeros(rws,clmns);

```

```

cntr1=[];
cntr2=[];
cntr3=[];
cntr4=[];
cntr5=[];
cntr6=[];

```

```

for f=1:files
    cntr1=cat(2,cntr1,M(195,45,f));
    cntr2=cat(2,cntr2,M(185,167,f));
    cntr3=cat(2,cntr3,M(2,95,f));
    cntr4=cat(2,cntr4,M(205,20,f));
    cntr5=cat(2,cntr5,M(60,175,f));
    cntr6=cat(2,cntr6,M(180,25,f));
end

```

```

x=1;
%count = 262144;
%while count>=5243 | x==10
limit = 180000000; % ~((1.8*(255/100))^2)*(256^2)*144;
%limit=100000000;
sse=limit+1;
ssev=[];

```

```

while ((sse >= limit) && (x <= 20))
%for x=1:2
    count = 0;
    new_cntr1 = zeros(1,files);
    new_cntr2 = zeros(1,files);
    new_cntr3 = zeros(1,files);
    new_cntr4 = zeros(1,files);
    new_cntr5 = zeros(1,files);
    new_cntr6 = zeros(1,files);

    new_cntr1_count=0;
    new_cntr2_count=0;
    new_cntr3_count=0;
    new_cntr4_count=0;
    new_cntr5_count=0;

```

```

new_cntr6_count=0;

pixelvlueto_cntr1=0;
pixelvlueto_cntr2=0;
pixelvlueto_cntr3=0;
pixelvlueto_cntr4=0;
pixelvlueto_cntr5=0;
pixelvlueto_cntr6=0;

sse = 0;

for i=1:rws
    for j=1:clmns
        temp=M(i,j,files+1);
        for fcnt=1:files
            pixelvlueto_cntr1 = pixelvlueto_cntr1+(M(i,j,fcnt) - cntr1(fcnt))^2;
            pixelvlueto_cntr2 = pixelvlueto_cntr2+(M(i,j,fcnt) - cntr2(fcnt))^2;
            pixelvlueto_cntr3 = pixelvlueto_cntr3+(M(i,j,fcnt) - cntr3(fcnt))^2;
            pixelvlueto_cntr4 = pixelvlueto_cntr4+(M(i,j,fcnt) - cntr4(fcnt))^2;
            pixelvlueto_cntr5 = pixelvlueto_cntr5+(M(i,j,fcnt) - cntr5(fcnt))^2;
            pixelvlueto_cntr6 = pixelvlueto_cntr6+(M(i,j,fcnt) - cntr6(fcnt))^2;
        end
        pixelvlueto_cntr1 = sqrt(pixelvlueto_cntr1);
        pixelvlueto_cntr2 = sqrt(pixelvlueto_cntr2);
        pixelvlueto_cntr3 = sqrt(pixelvlueto_cntr3);
        pixelvlueto_cntr4 = sqrt(pixelvlueto_cntr4);
        pixelvlueto_cntr5 = sqrt(pixelvlueto_cntr5);
        pixelvlueto_cntr6 = sqrt(pixelvlueto_cntr6);

        pixelvlueto_distcs =
[pixelvlueto_cntr1,pixelvlueto_cntr2,pixelvlueto_cntr3,pixelvlueto_cntr4,pixelvlueto_
to_cntr5,pixelvlueto_cntr6];
        [r,c]= min(pixelvlueto_distcs);
        M(i,j,files+1) = c;

        % test=M(i,j,4);
        if temp ~= M(i,j,files+1)
            count = count + 1;
        end

        switch c
            case 1
                for fcnt1=1:files
                    new_cntr1(fcnt1)=new_cntr1(fcnt1)+M(i,j,fcnt1);
                end
            end
        end
    end
end

```

```

        new_cntr1_count=new_cntr1_count+1;

        sse = sse + pixelvlue_to_cntr1^2;
    case 2
        for fcnt2=1:files
            new_cntr2(fcnt2)=new_cntr2(fcnt2)+M(i,j,fcnt2);
        end
        new_cntr2_count=new_cntr2_count+1;

        sse = sse + pixelvlue_to_cntr2^2;
    case 3
        for fcnt3=1:files
            new_cntr3(fcnt3)=new_cntr3(fcnt3)+M(i,j,fcnt3);
        end
        new_cntr3_count=new_cntr3_count+1;

        sse = sse + pixelvlue_to_cntr3^2;
    case 4
        for fcnt4=1:files
            new_cntr4(fcnt4)=new_cntr4(fcnt4)+M(i,j,fcnt4);
        end
        new_cntr4_count=new_cntr4_count+1;

        sse = sse + pixelvlue_to_cntr4^2;
    case 5
        for fcnt5=1:files
            new_cntr5(fcnt5)=new_cntr5(fcnt5)+M(i,j,fcnt5);
        end
        new_cntr5_count=new_cntr5_count+1;

        sse = sse + pixelvlue_to_cntr5^2;
    case 6
        for fcnt6=1:files
            new_cntr6(fcnt6)=new_cntr6(fcnt6)+M(i,j,fcnt6);
        end
        new_cntr6_count=new_cntr6_count+1;

        sse = sse + pixelvlue_to_cntr6^2;
    end

end

end

ssev=cat(2,ssev,sse);
if sse <= limit;
    break;

```

```

end

if new_cntr1_count ~= 0
    for flc1=1:files
        cntr1(flc1)=new_cntr1(flc1)/new_cntr1_count;
    end
    %clear cenroids to calculate new centroids
end
if new_cntr2_count ~= 0
    for flc2=1:files
        cntr2(flc2)=new_cntr2(flc2)/new_cntr2_count;
    end
end
if new_cntr3_count ~= 0
    for flc3=1:files
        cntr3(flc3)=new_cntr3(flc3)/new_cntr3_count;
    end
end
if new_cntr4_count ~= 0
    for flc4=1:files
        cntr4(flc4)=new_cntr4(flc4)/new_cntr4_count;
    end
end
if new_cntr5_count ~= 0
    for flc5=1:files
        cntr5(flc5)=new_cntr5(flc5)/new_cntr5_count;
    end
end
if new_cntr6_count ~= 0
    for flc6=1:files
        cntr6(flc6)=new_cntr6(flc6)/new_cntr6_count;
    end
end

x=x+1;

end

themmap=M(:, :, files+1);

```

```

% This code reads in the pixel values of two thematic maps and produces
% a confusion matrix

files=144;
rws=256;
clmns=256;

M=zeros(rws,clmns);
%Mb=zeros(rws,clmns);
flindx=[1:files];
for f1=1:files
    if flindx(f1) < 10
        file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+flindx(f1)));
        file=cat(2,file,'.raw');
        M(:,f1)=ReadRawImage2(file,clmns,rws);
    elseif flindx(f1) >= 10 && flindx(f1) < 100
        % if (f1 <= filemax)

file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(flinindx(f1)/10
)));
        file=cat(2,file,char(48+mod(flinindx(f1),10)));
        file=cat(2,file,'.raw');
        M(:,f1)=ReadRawImage2(file,clmns,rws);
    % else
    %
file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(f1*2/10)));
    % file=cat(2,file,char(48+mod((f1*2),10)));
    % file=cat(2,file,'.raw');
    % Mb(:,:(f1*2)-filemax))=ReadRawImage2(file,clmns,rws);
    % end
    else
    % if (f1 <= filemax)

file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(flinindx(f1)/10
0)));
        file=cat(2,file,char(48+floor((flindx(f1)-((floor(flinindx(f1)/100))*100))/10)));
        file=cat(2,file,char(48+mod(flinindx(f1),10)));
        file=cat(2,file,'.raw');
        M(:,f1)=ReadRawImage2(file,clmns,rws);
    % else
    %
file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(f1*2/100)));
    % file=cat(2,file,floor((f1*2-floor(f1*2/100)*100)/10));
    % file=cat(2,file,char(48+mod((f1*2),10)));
    % file=cat(2,file,'.raw');
    % Mb(:,:(f1*2)-filemax))=ReadRawImage2(file,clmns,rws);

```

```

%     end
    end
end

o_thmp=isodata_mindist2(files,rws,clmns,M);
%%

%files=144;
%rws=256;
%clmns=256;

MR=zeros(rws,clmns);
%Mb=zeros(rws,clmns);
%flsusc=files;
flindx=[1:files];
for f1=1:files
    if flindx(f1) < 10
        file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+flindx(f1)));
        file=cat(2,file,'R.raw');
        MR(:,f1)=ReadRawImage2(file,clmns,rws);
    elseif flindx(f1) >= 10 && flindx(f1) < 100
%        if (f1 <= filemax)

file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(flinindx(f1)/10)
));
        file=cat(2,file,char(48+mod(flinindx(f1),10)));
        file=cat(2,file,'R.raw');
        MR(:,f1)=ReadRawImage2(file,clmns,rws);
%    else
%
file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(f1*2/10)));
%        file=cat(2,file,char(48+mod((f1*2),10)));
%        file=cat(2,file,'.raw');
%        Mb(:,((f1*2)-filemax))=ReadRawImage2(file,clmns,rws);
%    end
    else
%        if (f1 <= filemax)

file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(flinindx(f1)/10
0)));
        file=cat(2,file,char(48+floor((flindx(f1)-((floor(flinindx(f1)/100))*100))/10)));
        file=cat(2,file,char(48+mod(flinindx(f1),10)));
        file=cat(2,file,'R.raw');
        MR(:,f1)=ReadRawImage2(file,clmns,rws);
%    else

```



```

%
file=cat(2,'C:\Users\Seton\Desktop\MS_Thesis\PCAcodes\M',char(48+floor(f1*2/100)));
%     file=cat(2,file,floor((f1*2-floor(f1*2/100)*100)/10));
%     file=cat(2,file,char(48+mod((f1*2),10)));
%     file=cat(2,file,'.raw');
%     Mb(:,:,((f1*2)-filemax))=ReadRawImage2(file,clmns,rws);
%     end
    end
end

r_thmp=isodata_mindist2(files,rws,clmns,MR);
%%

classes=6;

conf_matrix = zeros(classes,classes);
column_sum = zeros(1,classes);
row_sum = zeros(1,classes);
overall_accuracy=0;
err_of_inclusion=0;
err_of_exclusion=0;
diagonal_sum=0;
total=0;
temp=0;
erexclud=zeros(1,classes);
erinclud=zeros(1,classes);
overall_acc=0;

for i=1:rws
    for j=1:clmns
        conf_matrix(r_thmp(i,j),o_thmp(i,j))=conf_matrix(r_thmp(i,j),o_thmp(i,j))+1;
    end
end

for i=1:classes
    column_sum(i) = sum(conf_matrix(:,i));
    row_sum(i) = sum(conf_matrix(i,:));
    diagonal_sum=diagonal_sum+conf_matrix(i,i);
    erexclud(i)=conf_matrix(i,i)/column_sum(i);
    erinclud(i)=conf_matrix(i,i)/row_sum(i);
end;

total=sum(column_sum)
erexclud
erinclud

```

```
overall_acc=diagonal_sum/total  
conf_matrix
```

```
figure(1);  
mesh(o_thmp);  
figure(2);  
mesh(r_thmp);
```