AN INVESTIGATION OF DATA COMPRESSION TECHNIQUES FOR HYPERSPECTRAL CORE IMAGER DATA

Kerry-Anne Cawse[*] , Steven Damelin[†] , Louis du Plessis[‡] , Richard McIntyre[§] , Michael Mitchley[¶] and Michael Sears[‖]

Abstract

We investigate algorithms for tractable analysis of real hyperspectral image data from core samples provided by AngloGold Ashanti. In particular, we investigate feature extraction, non-linear dimension reduction using diffusion maps and wavelet approximation methods on our data.

1 Introduction

Until quite recently it was uncommon to have large amounts of hyperspectral data to store and process. Most data sets – albeit large in terms of numbers of bands – contained relatively low numbers of pixels. For example, a normal strip of HyMap data would contain about 2 million pixels [8]. Certainly, this is plenty of data when the survey area consists of multiple strips and over

[*]School of Computational and Applied Mathematics, University of the Witwatersrand, Johannesburg,Private Bag 3, Wits 2050, South Africa. *e-mail: acawse@telkomsa.net*

[†]The Unit for Advances in Mathematics and its Applications, Department of Mathematical Sciences, Georgia Southern University, P.O. Box 8093, Statesboro, GA 30460-8093, U.S.A. *e-mail: damelin@georgiasouthern.edu*

[‡]School of Computer Science and School of Computational and Applied Mathematics, University of the Witwatersrand, Johannesburg, Private Bag 3, Wits 2050, South Africa. *e-mail: laduplessis@gmail.com*

[§]School of Computational and Applied Mathematics, University of the Witwatersrand, Johannesburg, Private Bag 3, Wits 2050, South Africa. *e-mail: mcintyrer@gmail.com*

[¶]School of Computer Science, University of the Witwatersrand, Johannesburg, Private Bag 3, Wits 2050, South Africa. *e-mail: mitchleym@cs.wits.ac.za*

[‖]Computer Science, University of the Witwatersrand, Johannesburg, Private Bag 3, Wits 2050, South Africa. *e-mail: msears@icon.co.za*

100 spectral bands are involved. In the case of the AngloGold Ashanti Hyperspectral Core Imager (HCI), however, one metre of core generates 250 000 pixels and some 400 bands. This is the same order of magnitude, but the instrument scans about 5 metres per hour, generating about 2Gb of data in an hour. With core running to thousands of metres, it is easy to visualise unmanageably large data sets being acquired quite rapidly.

On the satellite front, ASTER covers large area scenes, but the multispectral nature of the data and the three independent spectrometers keep the data sets relatively small. Hyperion produced true hyperspectral data, but again the scenes were relatively small areas and sparsely acquired. However, the advent of the EnMap satellite system at the end of the decade – a truly operational hyperspectral satellite system – will change the availability of hyperspectral data sets and their size. Presumably the raw data will be stored by the satellite principles. This makes compression possibilities still more attractive for the users.

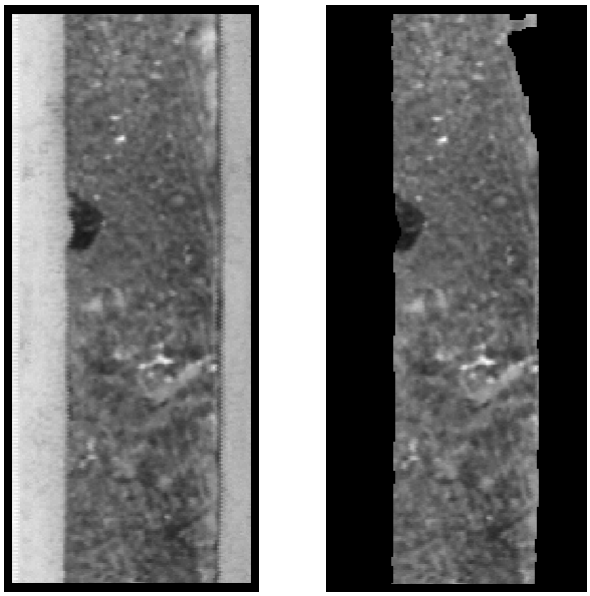

Figure 1: A view of the core sample data at a particular frequency on the left and the masked image at a particular frequency on the right.

These sensor developments motivate the need for hyperspectral data compression, and there are two potential aspects to address. The first of these is storage. In the case of the HCI, it is natural to store the raw data, but a variety of intermediate and final processing products produce multiple

copies of the data of the same size. There is thus a need for compression to allow reasonably cost effective storage of these data sets. The second, perhaps more compelling, motivation is the need to process ever larger hyperspectral data sets. In the case of the HCI, processing different parts of the core independently leads to compatibility problems with the results. It would thus be desirable to have much smaller data sets that can be processed simultaneously over any coherent region of interest in the core. It seems likely that similar issues will be relevant with the commercial advent of EnMap data sets. In the case of airborne or satellite data, one would expect an atmospheric correction to be done before any sort of compression. Otherwise, inevitably most of the compressed information will explain the atmospheric spectrum.

Because of the immediate interest and availability of data sets, the MISG working group focussed on data from the HCI. A small test data set was available, courtesy of the Geosciences Resource Group of Anglo Technical Division. While the strongest motivation is to obtain compressed data that will still yield satisfactory processed products, we did not have time to test processing software on our compressed outputs. Further work on the project will allow tests using several processing techniques and quantitative and qualitative comparisons with the results obtained on uncompressed data.

The left of Figure 1 shows a monochrome image of the core sample at a particular frequency. The Hyperspectral Core Imager (HCI) scans each point of the core at 400 different frequencies. Note that the raw data needs processing to remove, for example, the core tray shown at the right and left edges. The cropped data is shown on the right of Figure 1.

In view of the large size of even this data set, for convenience, we used a still smaller sample of the core, as shown in Figure 2. Figure 3 is a three-dimensional view of what the data looks like – there are r frequencies, and for each frequency, there is a map of dimension $p \times q$. Using the r frequencies a complete spectrum can be drawn for every pixel in the image.

Brainstorming around the problem suggested at least three approaches worthy of pursuit. All needed to exploit the intrinsic redundancy inherent in hyperspectral data sets.

The first method is to concentrate on the features in the spectra. Most processing techniques identify endmembers in the data and then unmix against an appropriate set of these. Both the selection of, and match with, the endmembers is essentially an automated process of feature matching, so this approach seems reasonable. Difficulty arises in the definition of what constitutes a feature, and how to efficiently do the coding. The details of the approach we adopted and some indications of its success are presented

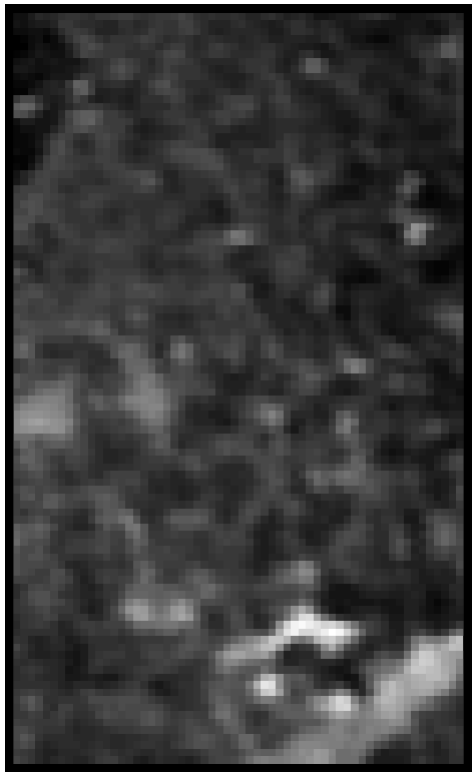

Figure 2: The reduced sample used for testing.

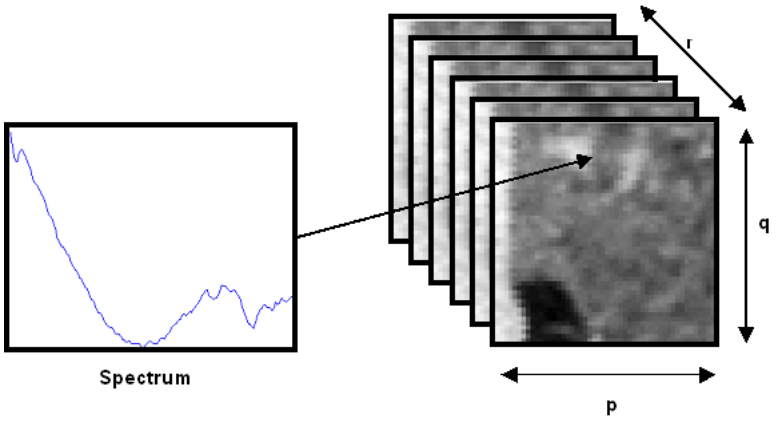

Figure 3: The hyperspectral data cube.

in Section 2 below.

The second approach is to mathematically exploit the redundancy directly using diffusion metrics. Here the dimensionality is reduced while preserving local distance. This is exactly what one wants to do when clustering mineral types within HCI data, and is the key idea behind the processing method currently used by GRG which employs Self Organising Maps (SOM). Intrinsically, this approach is thus very attractive and offers considerable potential. The major issue here is a computational one: large matrices need to be processed to obtain eigenvalues and eigenvectors. This is an interesting issue in its own right, and worthy of research. In Section 3 below the approach is described and some results presented. Since only standard PCs were available at the workshop, it was impossible to process even the available test data set completely.

The final approach is to use one of the standard wavelet coding procedures for each spectrum. This offers a standard compression approach by coding the data by the components of the basis wavelets used, and thus reduces the dimensionality to the size of the basis chosen. The encouraging nature of this approach is presented in Section 4 below.

Finally, we were very fortunate to attract an enthusiastic and interested group to work on this project. Certainly we would never have made the sort of progress outlined below, if it hadn't been for their commitment and hard work.

2 Feature extraction

As in all feature extraction procedures, the issue is deciding what constitutes a feature and what is noise. However, the context is helpful. We are not trying to identify relevant features for an application, but merely trying to identify the key features which describe a particular spectrum.

The approach adopted was to look at the slope of the spectrum as a function of the bands, and to record the spectral value and the band at which it occurred, if the slope of the curve changed significantly at that point. Of course, there is a change of slope at (almost) every band since the spectrometer is discretising the spectrum in any case. So we slightly smoothed the data and then selected every turning point. These turning points were then clustered to show the major turning points. These were around 20 points from the original data set of around 500. This is illustrated in Figure 4. A useful aspect of this procedure is that it automatically smoothes out noise in the data by ignoring features arising from small changes resulting from

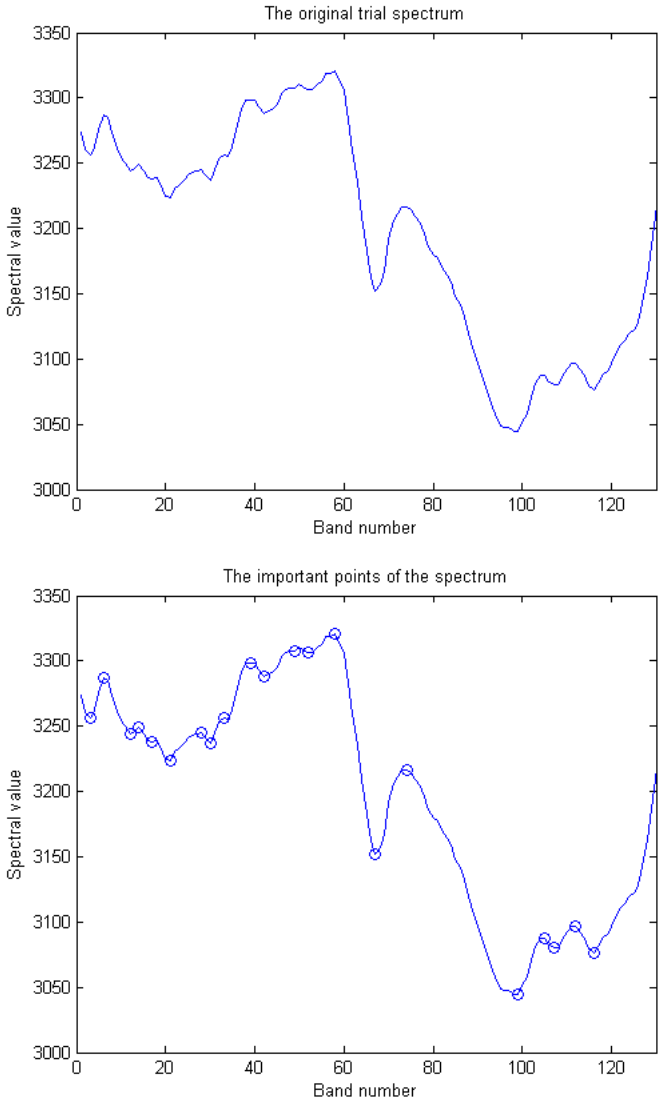

Figure 4: The original trial spectrum features selected by the algorithm

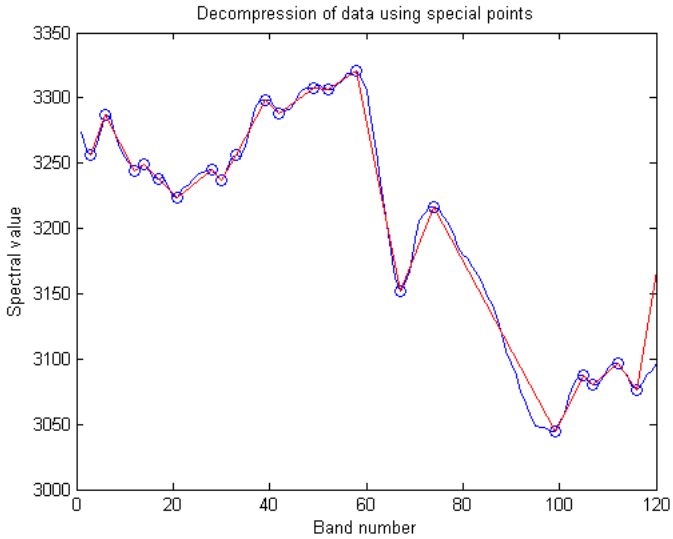

Figure 5: Shape-following curve shown against feature points and original spectrum.

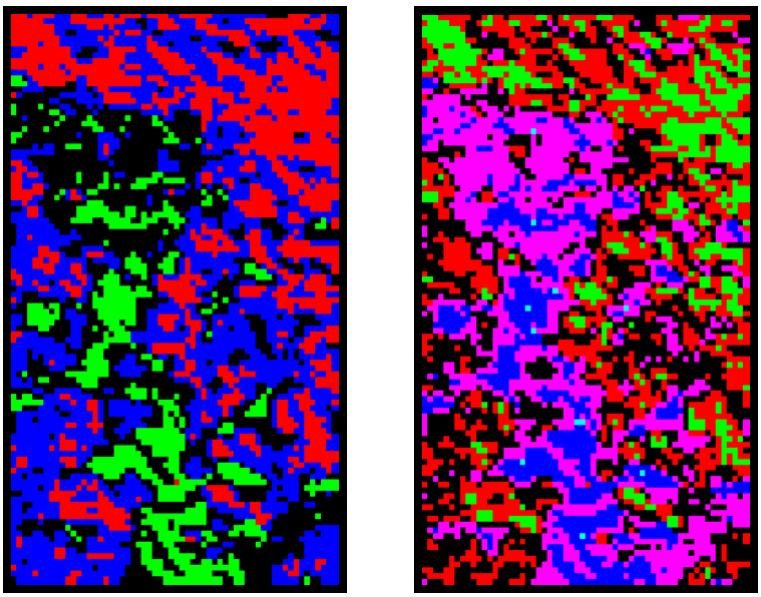

Figure 6: The image on the left shows 4 clusters extracted with K-means, and the image on the right shows 6 clusters.

system noise. A less attractive feature is that the compressed data are hard to work with because different spectra will have their 20 points selected at different bands. This problem can be dealt with at a coding level.

When it comes to "decompressing" the data, it is easy to fit a curve through the stored data points. Clearly we want to reconstruct the relatively smooth nature of the data without the effects of noise, and also preserve the shape features which were the motivation for the procedure in the first place. This was done by fitting a shape-following curve, as shown in Figure 5.

When it comes to processing data after this procedure, the nature of the information extraction problem needs to be considered. A feature extraction type requirement might allow processing directly from the extracted features. In general, however, we would probably need to use the data reconstructed as above. An indication of how this could work is shown in Figure 6. On the left of Figure 6, K-means has been used with four clusters to classify the reconstructed data, while on the right of Figure 6, K-means has been used with six clusters for classification. These are crude classification techniques, but spatial coherence of the regions selected is indicated. The next step in this approach would be to process the reconstructed data using more sophisticated techniques such as those already developed for the processing of "raw" HCI data, or by using SOM for example.

3 Diffusion maps

The idea behind using diffusion metrics is to reduce dimensionality of the data while preserving local distance between points. This is a particularly interesting approach, because this is exactly the way in which the different clusters need to be represented for the Hyperspectral Core Imager (HCI). The approach is to represent the data optimally – in some sense – in a lower dimensional space than the intractable 400 dimensional space in which the data are collected.

3.1 Methodology

We start with a rectangular array of $p \times q$ data points, where each data point is a spectrum of r values. This is reshaped to get a vector of $N = pq$ data points, $\Omega = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. On Ω, we define a weight function, $w(\mathbf{x}, \mathbf{y})$ such that $w(\mathbf{x}, \mathbf{y}) = w(\mathbf{y}, \mathbf{x})$ and $w(\mathbf{x}, \mathbf{y}) \geq 0$. The affinity matrix, $\mathbf{W}$, can then be calculated, where the i, j^{th} entry of $\mathbf{W}$ is defined to be $w(\mathbf{x}_i, \mathbf{x}_j)$. From the constraints imposed on the weight function, it is obvious to see that $\mathbf{W}$ will be a symmetric semi-positive definite matrix.

The next step is to define a Markov random walk on the data set. The i, j^{th} element of the transition matrix for the Markov process, $\mathbf{P}$, is given by

$$p(\mathbf{x}_i, \mathbf{x}_j) = \frac{w(\mathbf{x}_i, \mathbf{x}_j)}{d(\mathbf{x}_i)} \tag{1}$$

where

$$d(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in \mathbf{X}} w(\mathbf{x}_i, \mathbf{x}_j) \ . \tag{2}$$

The probability of moving from one data point to another in one time step is given by $p(\mathbf{x}, \mathbf{y})$. The time can be increased to find the probability of moving from $\mathbf{x}$ to $\mathbf{y}$ in t time steps, $p_t(\mathbf{x}, \mathbf{y})$. This probability is given by the corresponding entry of P^t [11].

It is reasonable to expect that as time tends toward infinity, the system will converge to a steady state. As long as every state is reachable from every other state, every column of $\mathbf{P}^t$ will converge to the same probability as time tends toward infinity [9]. That is

$$\lim_{t \to \infty} p_t(\mathbf{x}, \mathbf{y}) = \phi_0(\mathbf{y}) \ , \tag{3}$$

where $\phi_0(\mathbf{y})$ is given by

$$\phi_0(\mathbf{y}) = \frac{d(\mathbf{y})}{\sum_{z \in \Omega} d(\mathbf{z})} \ . \tag{4}$$

Another way to look at the Markov process is in terms of random walkers. If a random walker starts at $\mathbf{x}$ and moves to other data points, based on the values of the transition matrix, as time tends toward infinity, it will be trapped for long times in certain subsets of points, with only rare transitions between them [6]. These subsets are the clusters we are looking for.

We want to define a metric that preserves local distance between data points. If the probability distributions of the two points are close, then the points will be close as well. Using this observation, we define the diffusion distance

$$D_t^2(\mathbf{x}, \mathbf{y}) = \sum_{z \in \Omega} \frac{(p_t(\mathbf{x}, \mathbf{z}) - p_t(\mathbf{y}, \mathbf{z}))^2}{\phi_0(\mathbf{z})} \ . \tag{5}$$

The diffusion distance reflects the connectivity of the data. Points that are close together in the original data set will have a small diffusion distance. The diffusion distance compares all possible paths from $\mathbf{x}$ to $\mathbf{y}$, making it

very robust to noise [4]. Another way to look at the diffusion distance is as a comparison between two random walks, starting at $\mathbf{x}$ and $\mathbf{y}$, respectively [6].

We can cluster together subsets of data points with a small diffusion distance between them. As the time parameter, t, in the diffusion distance is increased, the clusters defined by the diffusion distance becomes coarser [11].

It is shown in [4] that $\mathbf{P}$ has a sequence of N eigenvalues and eigenvectors. Let $|\lambda_0| \geq |\lambda_1| \geq \cdots \geq |\lambda_{N-1}|$ be the eigenvalues and $\psi_0, \psi_1, \ldots, \psi_{N-1}$ the corresponding set of right eigenvectors of $\mathbf{P}$, such that $\mathbf{P}\psi_i = \lambda_i \psi_i$. It can be verified that $\lambda_1 = 1$ and $\psi_0 = \mathbf{1}$ [9]. This follows from the property that the sum of every row in $\mathbf{P}$ is equal to 1.

In [9] it is shown that the diffusion distance can be written as

$$D_t^2(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{N-1} \lambda_j^{2t} (\psi_j(\mathbf{x}) - \psi_j(\mathbf{y}))^2 \,. \tag{6}$$

Note that ψ_0 is not considered, since it is a constant vector and has no effect on the sum. Because of the decay of the eigenvalues, $D_t^2(\mathbf{x}, \mathbf{y})$ can be approximated to the desired accuracy by only considering the first $q(t)$ eigenvalues. The accuracy is obtained by setting $q(t)$ equal to the largest index such that $|\lambda_{q(t)}|^t < \delta |\lambda_1|^t$ where δ defines the accuracy:

$$D_t^2(\mathbf{x}, \mathbf{y}) \simeq \sum_{j=1}^{q(t)} \lambda_j^{2t} (\psi_j(\mathbf{x}) - \psi_j(\mathbf{y}))^2 \,. \tag{7}$$

If we define the function $\Psi_t : \mathbb{R}^N \to \mathbb{R}^{q(t)}$ where

$$\Psi_t(\mathbf{x}) = \begin{pmatrix} \lambda_1^t \psi_1(\mathbf{x}) \\ \lambda_2^t \psi_2(\mathbf{x}) \\ \vdots \\ \lambda_{q(t)}^t \psi_{q(t)}(\mathbf{x}) \end{pmatrix} \tag{8}$$

then

$$D_t^2(\mathbf{x}, \mathbf{y}) \simeq \sum_{j=1}^{q(t)} \lambda_j^{2t} (\psi_j(\mathbf{x}) - \psi_j(\mathbf{y}))^2 = \|\Psi_t(\mathbf{x}) - \Psi_t(\mathbf{y})\|^2 \,, \tag{9}$$

where $\|\cdot\|$ denotes the Euclidean norm. The function Ψ_t gives a mapping of the data points from the original space to a $q(t)$-dimensional Euclidean space. This gives a low dimensional representation of the data [10].

3.2 Results

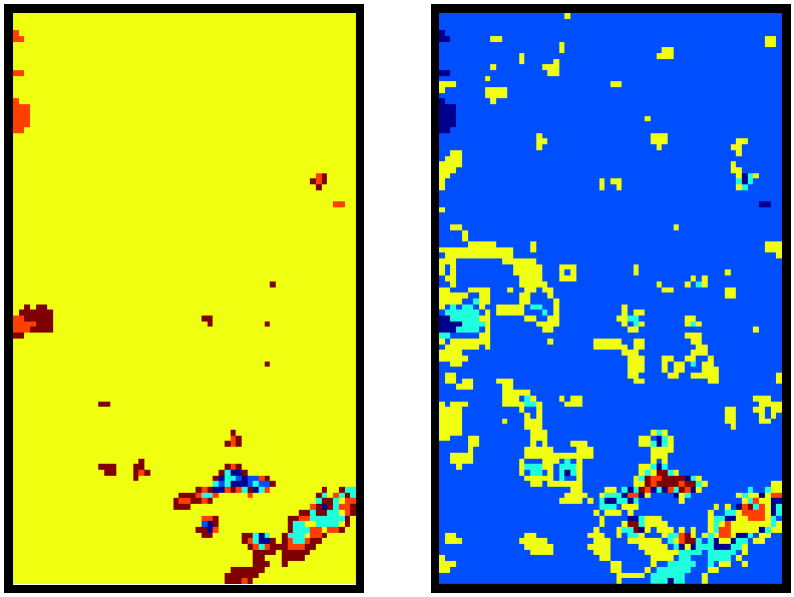

Figure 7: Results of applying the process of diffusion maps to the test data. The diffusion map was applied with a Gaussian kernel of width $\sigma = 10^3$, and reduced to 20 dimensions. The results were clustered using K-means. The left image shows the result of using 6 clusters and the right 10 clusters.

In all experiments we used the Gaussian kernel with width σ as a weight function:

$$w_\sigma(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right) \tag{10}$$

where $\|\cdot\|$ again denotes the Euclidean norm. We chose this kernel as it satisfies the requirements, and gives a high value for spectra that are close together and a low value for very different spectra. This is ideal since the weights will be mapped to probabilities. A high value means a high probability for the random walker to cross between the two data points, which in turn implies that the two points should be in the same cluster.

In [11] it is mentioned that there is currently no good theory for choosing σ. In general σ gives the rate at which similarity between two different spectra deteriorates. Choice of σ depends on the specific application, and is

a trade-off between sparseness of $\mathbf{P}$, for a small σ, and whether the kernel gives a true measure of affinity between the points.

Working code for the process of diffusion maps could not be implemented at the conference. However, Matlab code was written afterward. The results of applying the procedure on the data set is given in Figure 7. The diffusion map was used with a kernel width, $\sigma = 10^3$. All values less than 10^{-3} in the initial affinity matrix were set to 0. The dimension was reduced to 20 and the results clustered using K-means, first with 6 clusters, shown on the left, and then with 10 clusters, on the right. There is an obvious spatial correlation between bands of the unprocessed data and the results.

Already at *MISG 2008* it became apparent that the most costly part of the process is the construction of the affinity matrix. However, this matrix is symmetric, and all the diagonal entries are equal to one. This means that we only need to calculate $(N^2 - N)/2$ entries of the matrix. Calculating this matrix could be very easily parallelized however, since any two entries are completely independent of each other. The transition matrix, $\mathbf{P}$, can be obtained from $\mathbf{W}$ by dividing every row element-wise with $d(\mathbf{x}_i)$, where i corresponds to the row number. This could also be done in parallel. The resulting matrix will hopefully be sparse, and since we only need to find the first couple of eigenvectors and eigenvalues, this should not pose too much of a problem.

4 Wavelet approximation

Two applications of wavelets were considered at the MISG. The first application, quickly discarded, was as a method of smoothing the data for feature extraction. This was found to interfere with the mechanism of extraction, however, due to the imposition of the features of the wavelet basis.

The second application was to use wavelet approximation in its own right. Approximation theory tells us that we can approximate any piecewise continuous function using a family of orthogonal basis functions. The idea is to map the spectral data of each pixel onto its approximating coefficients. That is, for each pixel of the core, we store only the coefficients of the approximating functions. This maps the pixels into a much lower dimensional space, provided the order of approximation is chosen appropriately.

Wavelets have been widely applied to the analysis of hyperspectral data acquired from remote sensing [2, 7], as well as in the compression of hyperspectral data [5]. The method used will be presented in this section, together with the results obtained. These results will be compared to those acquired

using Chebyshev functional approximation of the hyperspectral data.

4.1 Methodology

For any basis of orthogonal functions $\phi_n(x)$, we can find the coefficients a_n of the projection of any function $f(x)$ onto that basis using the inner product, defined as [3]

$$a_n = \frac{\int_a^b w(x)\phi_n(x)f(x)dx}{\int_a^b w(x)\phi_n(x)\phi_n(x)dx} \ , \tag{11}$$

where the interval $[a,\ b]$ is the support of $\phi_n(x)$ and $w(x)$ is the weight function of the family of functions. For the Haar wavelets, this weight function is simply 1, and the support is the interval [0, 1]. The daughter functions of the mother wavelet $\psi(x)$ are generated using

$$\psi_{j,k}(x) = 2^{0.5j}\psi(2^j x - k) \tag{12}$$

for order j and $0 \leq k \leq 2^j - 1$. Since j and k take on discrete integer values, the Haar wavelet is known as a discrete wavelet. Since wavelets have a square-integral of 1, the coefficient of the projection of $f(x)$ onto the daughter wavelet $\psi_{j,k}(x)$ is given by [1]

$$a_{j,k} = 2^{0.5j}\int_0^1 \psi(2^j x - k)f(x)dx \ . \tag{13}$$

For the MISG, it was assumed that the number of bands in the data set are a power of two. This condition can be artificially reached by adding zero entries to the spectral data of each pixel. This does not affect the order of complexity, as we are, at most, nearly doubling the signal length. This expansion by a constant factor does not change the asymptotic growth rate. Since the data is discrete, the coefficient integral can be rewritten as

$$a_{j,k} = 2^{0.5j}\sum_{i=0}^{n}\left(f_i\int_{x_i}^{x_{i+1}}\psi(2^j x - k)dx\right) \ ,$$

where $x_0 = 0$ and $x_{n+1} = 1$, and f_i the ith element of the data vector. By the assumption that the number of bands (and thus the number of elements in the data vector) is a power of two, this can be further simplified to

$$a_{j,k} = 2^{0.5j}\sum_{i=0}^{n}\left(f_i\frac{\psi(2^j x_i - k)}{n}\right) \tag{14}$$

which can be represented as a matrix operation. By careful reshaping of the data cube into a matrix, we can thus compute all the required coefficients at once.

If the order of approximation is chosen such that the number of wavelets used in the approximation is greater than or equal to the number of data points in the spectral vector, the dimensionality of the data is not reduced. However, through successive passes, and encoding techniques, lossless compression can be achieved. Many wavelet compression schemes use this technique, resulting in less data to store, and more importantly, in the application of hyperspectral remote sensing, less data to transmit.

However, if the order of approximation is chosen such that the number of wavelets used is less than the number of bands, a different kind of dimension reduction can be used. Clustering techniques can be applied directly to the coefficient data, which may be less than an eighth of the size of the original data. This claim is motivated with a simple Euclidean distance clustering technique, although the more sophisticated K-means clustering algorithm was used to produce results.

Consider two pixels with coefficient vectors $\mathbf{a}$ and $\mathbf{b}$. For a suitable mapping from a single index i to the wavelet daughter indices j, k, it is clear that the approximated data can be thought of as a continuous function given by $\sum_i a_i\psi_i(x)$ and $\sum_i b_i\psi_i(x)$. If the Euclidean distance between these two functions is less than some tolerance, the pixels are similar, and can be grouped together. This is computed as

$$\int_0^1 \left(\sum_i a_i\psi_i(x) - \sum_i b_i\psi_i(x) \right)^2 dx \tag{15}$$

which, through algebraic manipulation, we can rewrite as

$$\int_0^1 \left(\sum_i \psi_i(x)(a_i - b_i) \right)^2 dx$$

$$= \sum_i \sum_j \int_0^1 \psi_i(x)\psi_j(x)(a_i - b_i)(a_j - b_j)dx$$

$$= \sum_i \sum_j (a_i - b_i)(a_j - b_j) \int_0^1 \psi_i(x)\psi_j(x)dx$$

However, the wavelet basis chosen is orthonormal, and so this becomes

$$\sum_i \sum_j (a_i - b_i)(a_j - b_j)\delta_{i,j}$$

where $\delta_{i,j}$ is the Kronecker delta function, and so the Euclidean distance between the two pixels is simply given by

$$\sum_i (a_i - b_i)^2 \tag{16}$$

which is the square of the Euclidean distance between the coefficient vectors of the two pixels. There are two important features of this to note. Firstly, we can find a clustering using only the approximating coefficients. That is to say, it is not neccessary to reconstruct the original data. Secondly, because the coefficients are those of an approximation, with a careful choice of order we can reduce the errors within the data by smoothing it out, at the expense of the smaller features. It is hoped, however, that the features neccessary to identify a particular mineral are not dominated by error.

This result implies that we can simply apply the K-means clustering algorithm to the coefficient data. The results of this application are presented in the next section.

4.2 Results

For the wavelet method, a larger data sample was taken, spanning 226 x 62 pixels, with 226 spectral bands per pixel, artificially expanded to 256 bands through the addition of zeros. The true-colour image is compared against the 6 group clustering obtained from a fifth-order Haar wavelet approximation in Figure 8. The sample data used in the MISG and presented in other sections is a subset of this data, approximately delineated with a white border in the true-colour image.

The shortage of time, however, prevented further investigation into more sophisticated techniques. Work was later performed in a tangentially related project on Chebyshev approximation of hyperspectral data, using the same methodology. It was found that a similar clustering could be obtained using Chebyshev approximation, with less than ten percent of the pixels being grouped differently. Figure 9 shows the clustering obtained from a twentieth-order Chebyshev approximation of the second kind. It can be shown that taking an appropriately weighted distance measure between any two pixels approximated by Chebyshev polynomials can also be reduced to the squared Euclidean distance of the coefficients of approximation.

It can be seen that the clustering is coherent, showing geologically-sound intrusion features. As Haar wavelets were used, it is possible that the method described above could benefit from greater sophistication. The data, which

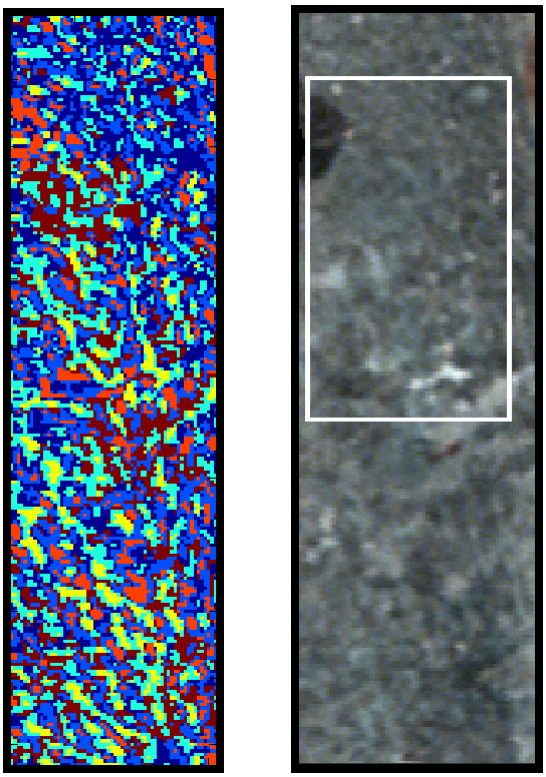

Figure 8: The k-means clustering obtained, together with the true-colour image of the core sample on the right. The subset of data used in the other methods is delineated in white on the true-colour image.

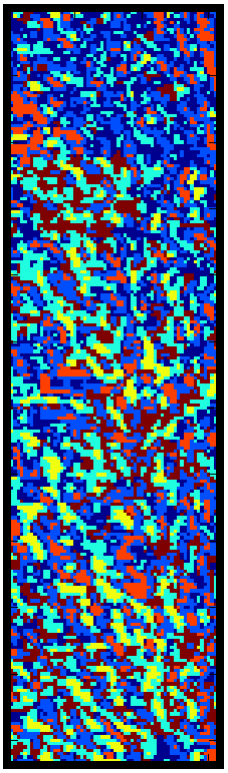

Figure 9: A clustering obtained from a 20th order Chebyshev approximation of the hyperspectral data.

consists of over three million values, was approximated in under a second and clustered in under a minute.

5 Conclusion

The main focus of the project, was to provide (1) real time time hyperspectral data and (2) to investigate several algorithms for tractible analysis of real hyperspectral data. We looked at feature extraction methods, nonlinear dimension reduction methods and wavelet approximation methods. All methods performed well and have given a strong basis for a more rigorous study in the future. For example, the diffusion methods we used, need to be made more robust and less memory intensive. We also wish to study compressive sampling methods on our data and improve our wavelet approximation methods. This ongoing project and its outcomes will be used

for honours and graduate work of Louis du Plessis and Michael Mitchley under the supervision of Steve Damelin and Michael Sears.

Acknowledgments

We are grateful to the staff at GRG of ATD of Anglo American for their support with advice and test data. We would also like to thank Ian M. Howat from the Applied Physics Lab at the University of Washington for the function `enviread.m`, which was used to extract the test data into Matlab. The function is available at `http://www.mathworks.com/matlabcentral/`. The research of Steven Damelin was funded, in part by `NSF-DMS-0555839`, the Unit for Advances in Mathematics and its Applications and the School of Computational and Applied Mathematics at the University of the Witwatersrand, Johannesburg.

A Feature Extraction Code

A.1 "interp.m"

This function finds special points in frequency graphs for each pixel in the image.

```
function s=interp(data)

for outer=1:141
    %outer and inner loops loop through every pixel in the image
    for inner=1:60

        %renumber pixels so that they form a vector
        pixelnum=60*(outer-1)+inner;
        y=reshape(data(outer,inner,:),1,141);

%determine the special (p,q) co-ordinates of special points
        A=peak(y,1,5);
        p=A(1,:);
        q=A(2,:);

%show the special points on a graph
plot(p,q,'r');


%we could reconstruct an approximation to the original graph
%using pchip (piecewise cubic Hermite interpolating polynomial)
```

```
%which uses the special points found by peak2
        %regr=pchip(p,q,1:141);

        %we would then normalise each point
        %regr(pixelnum,:)=regr(pixelnum,:)/norm(regr(pixelnum,:),'fro');

    end

end

%s=regr;
```

A.2 "peak.m"

This function finds turning points in a graph smoothed to degree n.

```
function p=peak2(x,n,m)

t=1:length(x);
next=1;
plot(t,x);
o(1)=t(1);

for loop=n+1:length(x)-n
    p1=polyfit(t(loop-n:loop),x(loop-n:loop),1);
    p2=polyfit(t(loop:loop+n),x(loop:loop+n),1);
    %draw straight lines to either side of a point and determine if turning point

    if p1(1)*p2(1)<=0
        o(next)=t(loop);
        next=next+1;
        %if turning point, save in o
    end
end

o(end+1)=t(end);
q=clusterfeatures(o,x,m);
%clusterfeatures clusters the special points in o

p=[q;x(q)];
```

A.3 "clusterfeatures.m"

This function clusters special points and saves cluster centers.

```
function y = clusterfeatures(o,x,m)

next=1;
hold on

%begin a cluster
cluster = o(1);

for loop=2:length(o)


        if  sqrt((o(loop)-cluster(end))^2+(x(o(loop))-x(cluster(end)))^2)>m
            %distance more than m, start new cluster
            %compare cluster(end) to o(loop)

            %get central point of cluster and store in c
            c(loop-1) = cluster(round(end/2));

            plot(c(loop-1),x(c(loop-1)),'o');

            %start new cluster
            cluster = o(loop);
        else
            %distance less, add to cluster
            cluster(end+1) = o(loop);
        end

end

if c(1)~=o(1)
    c=[o(1),c];
    %if the first point is a cluster on its own
end
if c(end)~=o(end)
    c(end+1)=o(end);
    %if the last point is a cluster on its own
end

c=unique(c);
%only store unique cluster values

y=c;
```

B Diffusion Map Code

B.1 "diffMap.m"

Performs a diffusion map with time parameter t, that reduces dimension to *newdim*, using a Gaussian kernel with width *sigma* on the image. The result is clustered into *cluster* clusters using K-means.

```
function y = diffMap(image,sigma,tol,t,newdim,clusters)

[p,q,r] = size(image);
P       = GetP(image,sigma,tol);
P       = sparse(P);
[v,e]   = eigs(P,newdim+1);
clear P;      % Don't need P anymore
y = doMap(v,e,t,newdim,p,q);
y = kmeans(y,clusters);
y = reshape(y,p,q);
```

B.2 "GetP.m"

Gets the transition matrix P on image, with kernel width *sigma*. All entries less than *tol* are set to zero.

```
function P = GetP(image,sigma,tol)

[p,q,r] = size(image);
image   = reshape(image,p*q,r);
sigma   = 1/sigma^2;
P = ones(p*q,p*q);
for (i = 1:p*q)
    for (j = i:p*q)
        if (i ~= j)
            P(i,j) = exp((-norm(image(i,:)-image(j,:))^2)*sigma);
            if (P(i,j) < tol)
                P(i,j) = 0;
            end
            P(j,i) = P(i,j);
        end
    end
    P(i,:) = P(i,:)/sum(P(i,:));
end
```

B.3 "doMap.m"

Does the actual diffusion map.

```
function y = doMap(v,e,t,newdim,oldx,oldy)

e = diag(e);
m = length(v(:,1));
e = e(2:end);
for (i = 1:newdim+1)
    if (v(1,i) < 0)
        v(:,i) = -1.*v(:,i);
    end
end
y = zeros(m,newdim);
for i = 1:m
    y(i,:) = (e.^t)'.*v(i,2:end);
end
y = reshape(y,oldx,oldy,newdim);
```

C Wavelet Code

C.1 "haarapproximate.m"

This is the program used to find the grouping of the hyperspectral data. Its primary function is to handle the reshapings required.

```
function flags = haarapproximate(data, order, cluster)
% Finds the grouping of normalised hyperspectral data, given an order of
% approximation, and the number of groups to supply to kmeans.

% First, find the coefficients of the approximation
[r, s, t] = size(data);
data = reshape(data, [], t);
if ceil(log(t)/log(2)) ~= log(t)/log(2)
    pad = 2^(ceil(log(t)/log(2))) - t;
    datapad = zeros(size(data, 1), pad);
    data = [data, datapad];
    t = size(data, 2);
end
A = generateHaarcoeffmatrix(t, order);
coeff = (2 / pi) * (data*A');

% The coefficients are used to cluster
flags = kmeans(coeff, cluster);
```

```
flags = reshape(flags, r, s);
```

C.2 "generatecoeffmatrix.m"

This generates the matrix used to find the approximating coefficients. When this matrix is multiplied with the reshaped data, all the coefficients are obtained at once.

```
function A = generateHaarcoeffmatrix(n, ord)
% generates the matrix used to compute the coefficients of a haar
% approximation, assuming power of 2 bands. n is the number of bands, ord is the
% order of approximation required.

h= 1/n;
A = zeros(2^ord, n);
A(1, :) = ones(1, n);
r = 2;
for i=0:ord-1
    for j=0:(2^i - 1)
        for k=1:n
            A(r, k) = 2^(0.5*i) * haarmother((2^i) * h * k - j);
        end
        r = r + 1;
    end
end
A = A/n;
```

C.3 "haarmother.m"

This defines the Haar mother wavelet.

```
function val = haarmother(x)
    if x <= 0
        val = 0;
    elseif x <= 0.5
        val = 1;
    elseif x <= 1
        val = -1;
    else
        val = 0;
    end
```

References

[1] Aboufadel, E and Schlicker,S. Discovering Wavelets. Wiley-Interscience, Hoboken, New Jersey, 1st edition, 1999.

[2] Bruce,L.M., Koger, C.H. and Jiang, L. Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, **40** (2002), 2331-2338.

[3] Burden, R and Faires, J. Numerical Analysis. Brooks/Cole, Pacific Grove, California, 7th edition, 2001.

[4] Coifman, R.R. and Lafon, S. Diffusion maps. *Journal of Applied and Computational Harmonic Analysis*, **21** (2006), 5-30.

[5] Fowler, J.E. and Rucker, J.T. Three-dimensional wavelet-based compression of hyperspectral imagery. In: Hyperspectral Data Exploitation: Theory and Applications. Chein-I Chang, editor, Wiley (2007), pp 379-407.

[6] Keller, Y, Lafon, S and Krauthammer, M. Protein cluster analysis via directed diffusion. In: Fifth Georgia Tech International Conference on Bioinformatics, Number 2005.

[7] Kempeneers, P., De Backer, S., Debruyn, W., Coppin, P. and Scheunders, P. Generic wavelet-based hyperspectral classification applied to vegetation stress detection. *IEEE Transactions on Geoscience and Remote Sensing*, **43** (2005), 610–614.

[8] Kerekes, J.P. and Schott, J.R. Hyperspectral imaging systems. In: *Hyperspectral Data Exploitation: Theory and Applications* Chein-I Chang, editor, Wiley (2007) pp 19-45.

[9] Lafon, S. and Lee, A.B. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28** (2006), 1393-1403.

[10] Nadler, B., Lafon, S., Coifman, R.R. and Kevrekidis, I.G. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Journal of Applied and Computational Harmonic Analysis*, **21** (2006), 113-127.

[11] Xu, R., Damelin, S. and Wunsch II, D.C. Applications of diffusion maps in gene expression data-based cancer diagnosis analysis. In: Engineering in Medicine and Biology Society. 29th Annual International Conference of the IEEE, August 2007, pp 4613–4616.