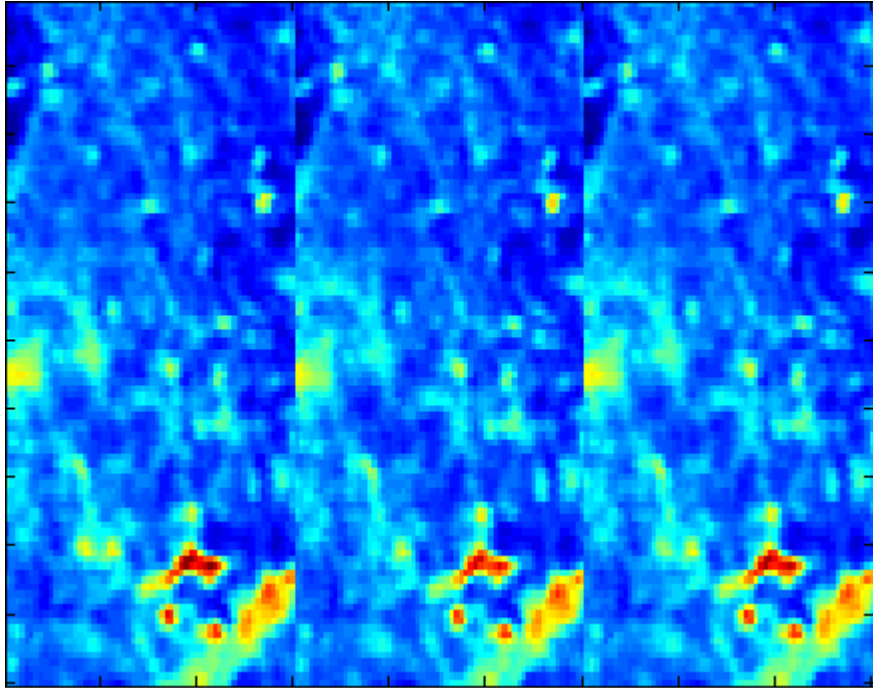


# COMPRESSION OF HYPERSPECTRAL IMAGES



MISG 2008

FEBRUARY 1, 2008

# 1 Introduction

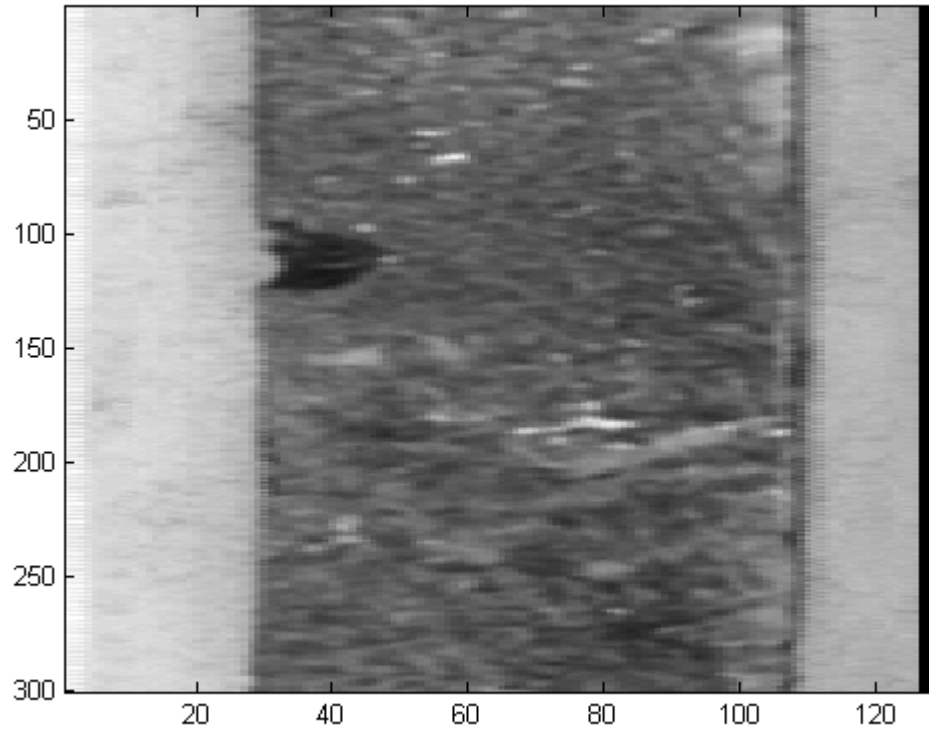


Figure 1: A view of the core sample test data at a particular frequency.

The image on this page show a picture of the core sample at a certain frequency. The Hyperspectral Core Imager (HCI) scans each point of the core at 400 different frequencies. This results in a large amount of data. However, some of the data is useless to us, for instance the sides which are clearly the tray, these are usually masked out, as shown in the next image. For convenience, we used a still smaller sample of the core, as shown in the third image. The fourth image is a three- dimensional view of what the data looks like - there are  $s$  frequencies, and for each frequency, there is a map of dimension  $p \times q$ .

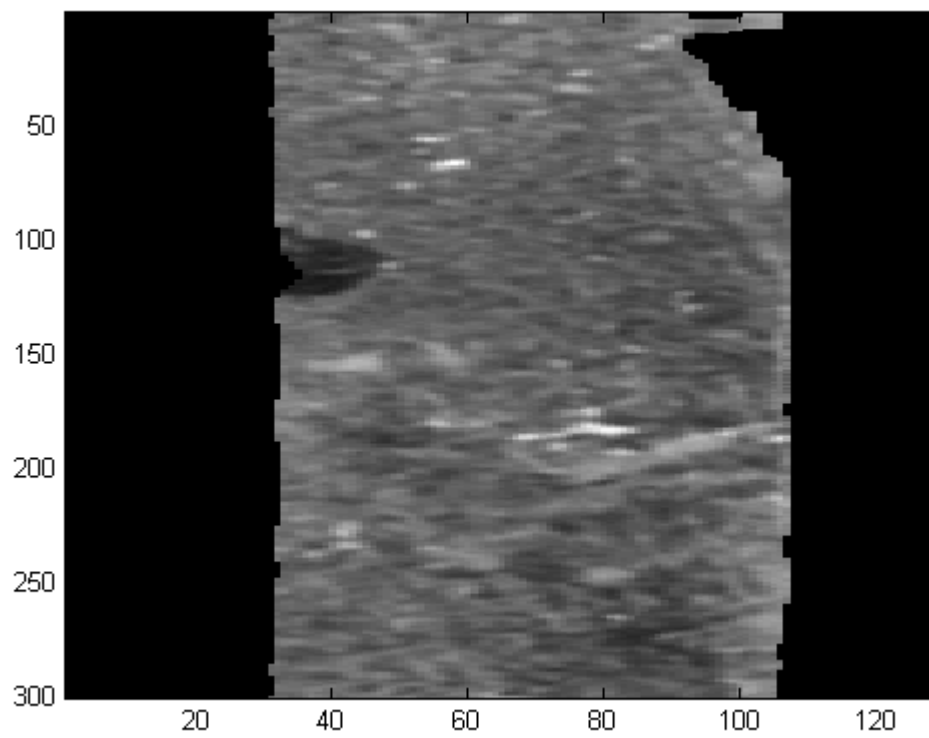


Figure 2: The masked image of the core at a particular frequency.

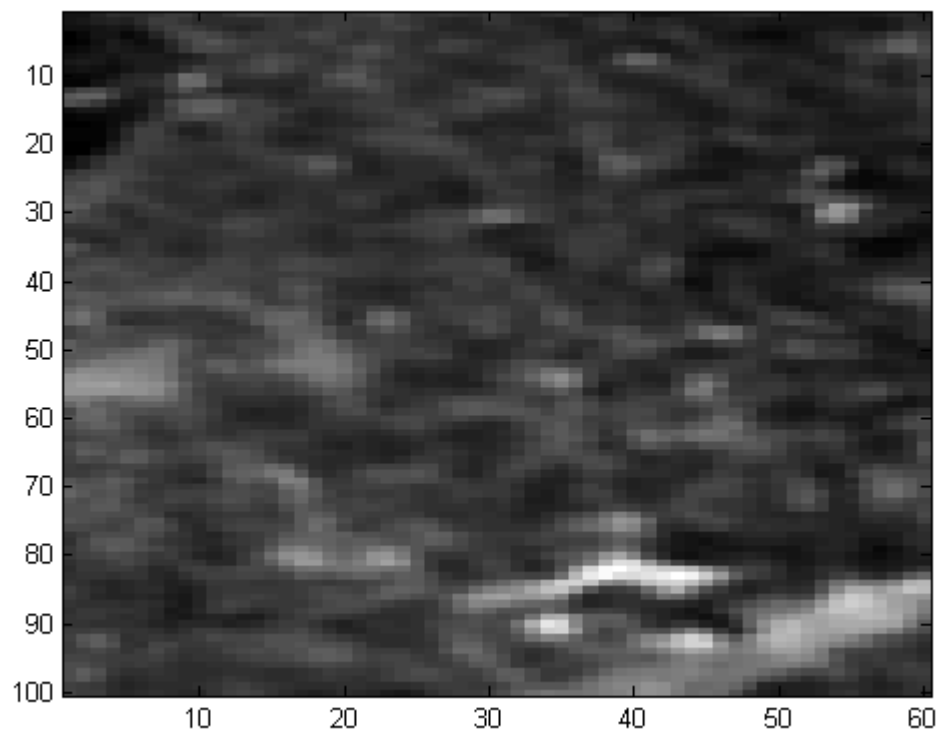


Figure 3: The sample of core we worked with.

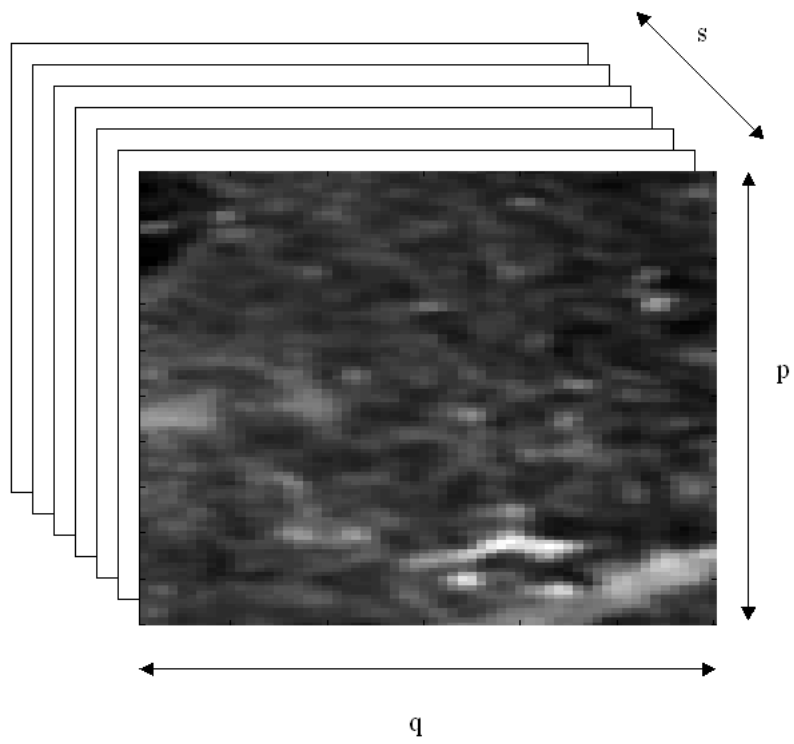


Figure 4: A view of what the data really looks like.

## 2 Diffusion Metrics

### 2.1 Motivation

The idea behind using diffusion metrics is to reduce dimensionality while preserving local distance between points. The current problem with analyzing hyperspectral images is the sheer amount of data. The Hyperspectral Core Imager (HCI) analyzes each point at 400 different frequencies. For every point, 400 values are produced. These values could be used to compare the different points and cluster parts of the core sample containing the same mineral together. However, comparing 400 values for every point would take extremely long.

This is the motivation for trying to reduce the dimension of the data. We would like to reduce the amount of values for every point to a much smaller number.

### 2.2 How is it done?

We have a matrix of dimension  $p \times q$  and for every entry on this matrix we have  $s$  samples. Essentially we have  $m = pq$  data vectors in  $\mathfrak{R}^s$ , call this matrix  $X$ .

$$X = [x_1, x_2, \dots, x_m] \quad x_i \in \mathfrak{R}^s, X \in \mathfrak{R}^{s \times m}$$

We use the heat kernel with tolerance factor  $\sigma$ , defined as follows:

$$K_\sigma(x, y) = \exp\left(\frac{\|x-y\|^2}{\sigma}\right)$$

We use this kernel to construct a comparison matrix  $A$  as follows:

$$A = [K_\sigma(x_i, x_j)] \quad A \in \mathfrak{R}^{m \times m}$$

We calculate the eigenvalues  $\lambda_i$  and associated eigenvectors  $\phi_i$  for  $A$ . Since the heat kernel is positive semi-definite,  $A$  is positive semi-definite and therefore all the eigenvalues are real and positive. We want to reduce the dimension to a number much smaller than  $s$  (which it currently is), say  $n$ . Hence, we only select the  $n$  eigenvectors associated with the  $n$  largest eigenvalues. We now define a mapping  $\psi$  to this lower dimension, from the original data vectors in  $X$ .

$$\psi : \mathfrak{R}^s \rightarrow \mathfrak{R}^n \quad \psi(x_i) = [\phi_{i,1}, \phi_{i,2}, \dots, \phi_{i,n}]^T$$

After applying  $\psi$  to every column of  $X$ , we have a matrix of dimension  $m \times n$  consisting of only the most important features of the original core sample. We can now reshape this to a three dimensional matrix of dimension  $p \times q \times n$  that represents the data in a more familiar form. We could then use some clustering algorithm to group together data points that have similar values. This will give us the location of the different minerals.

### 2.3 Problems

In practice we can expect to be working with about a metre of core at a time, maybe more. This gives a lot of data points, which in turn means  $A$  will be large as well. Calculating the spectral decomposition of large matrices is problematic. However, by choosing the correct  $\sigma$  for the heat kernel,  $A$  can be made sparse, which vastly simplifies the spectral decomposition.

Another way to simplify the spectral decomposition is to only consider a subset of the points, for example every  $10^{\text{th}}$  point. We used both approaches in our calculations.

### 2.4 Results

As stated above, we only considered a subset of the points, specifically, we only sampled a quarter of the points. We also have not clustered the results into different groups that can be displayed. The result we obtained was made up from reducing the dimension to 3 and plotting the resulting 3 images as a normal colour image. Although the results have not been clustered, and we only considered the three most important eigenvectors, the different bands of minerals can still be clearly observed.

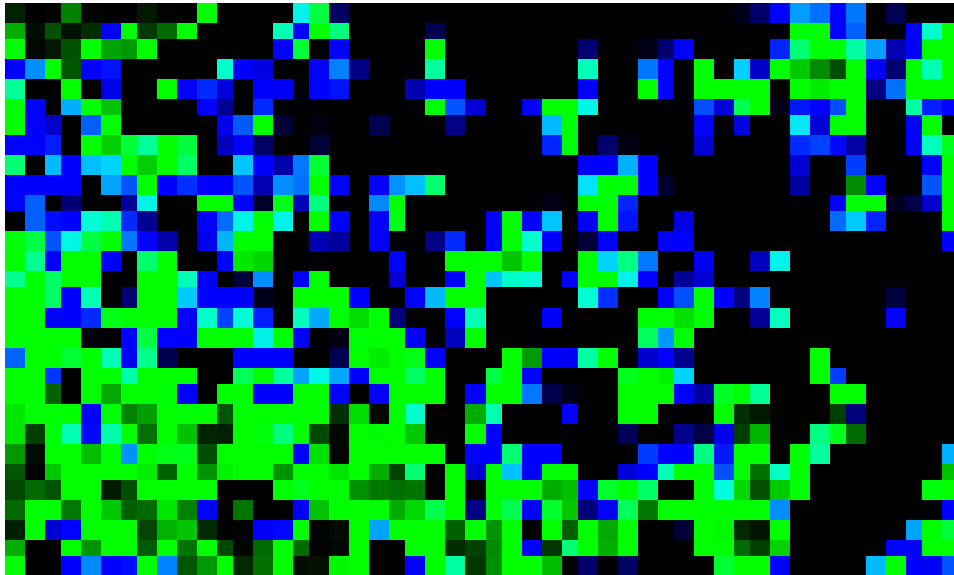


Figure 5: A plot of the results obtained when the dimension is reduced to 3.

## 3 Wavelets

### 3.1 Wavelets

What are wavelets? Wavelets are scaled and translated copies of a mother wavelet, which form an orthogonal set of functions that are either of finite-length, or are fast-decaying. These scaled and translated copies, called daughter wavelets, satisfy the same properties. I will illustrate these properties (rather than trying to explain them) by using the simplest (conceptually) wavelet, the Haar wavelet.

This is a discrete wavelet, definable with Heaviside step functions on the interval  $[0, 1]$ , which looks like this:

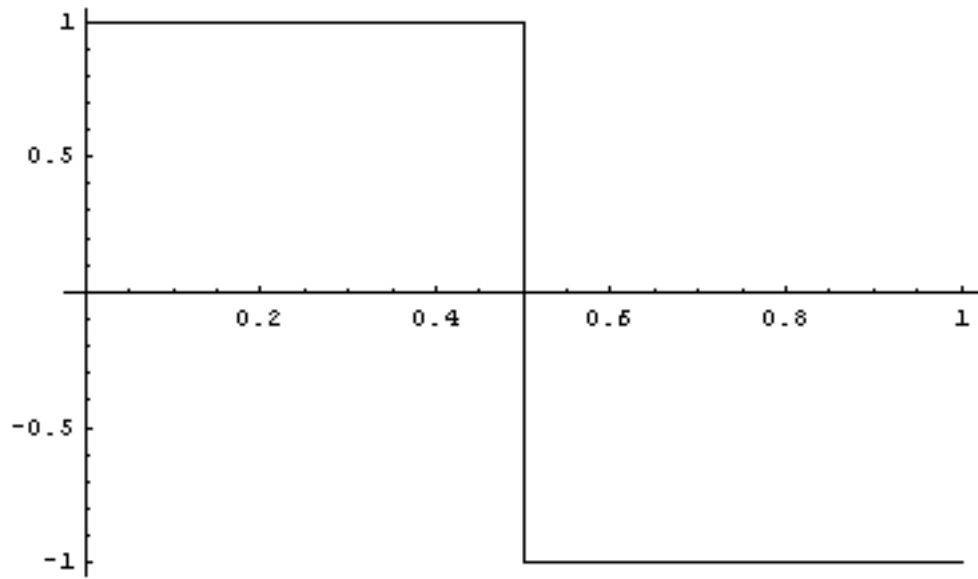


Figure 6: A plot of the Haar mother wavelet.

Its called discrete due to its discrete scale and shift parameters (since it is in fact piecewise smooth, with a finite number of jump discontinuities). Its integral across the real domain is zero, and its square integral is one. The first daughter wavelets are scaled to occupy half the domain of the mother wavelet, with each subsequent 'order' of daughter wavelets halving the domain further. The amplitude of the daughter wavelets is increased by a rooted factor of two. The daughter wavelets retain the zero integral and unity square integral properties of the mother wavelet. All the wavelets are orthogonal.

As can be seen from the daughter wavelets, they are localised by amplitude as well as frequency (if we consider the domain to be some frequency measure, as it was in our data). This gives wavelets an advantage over Fourier transforms for

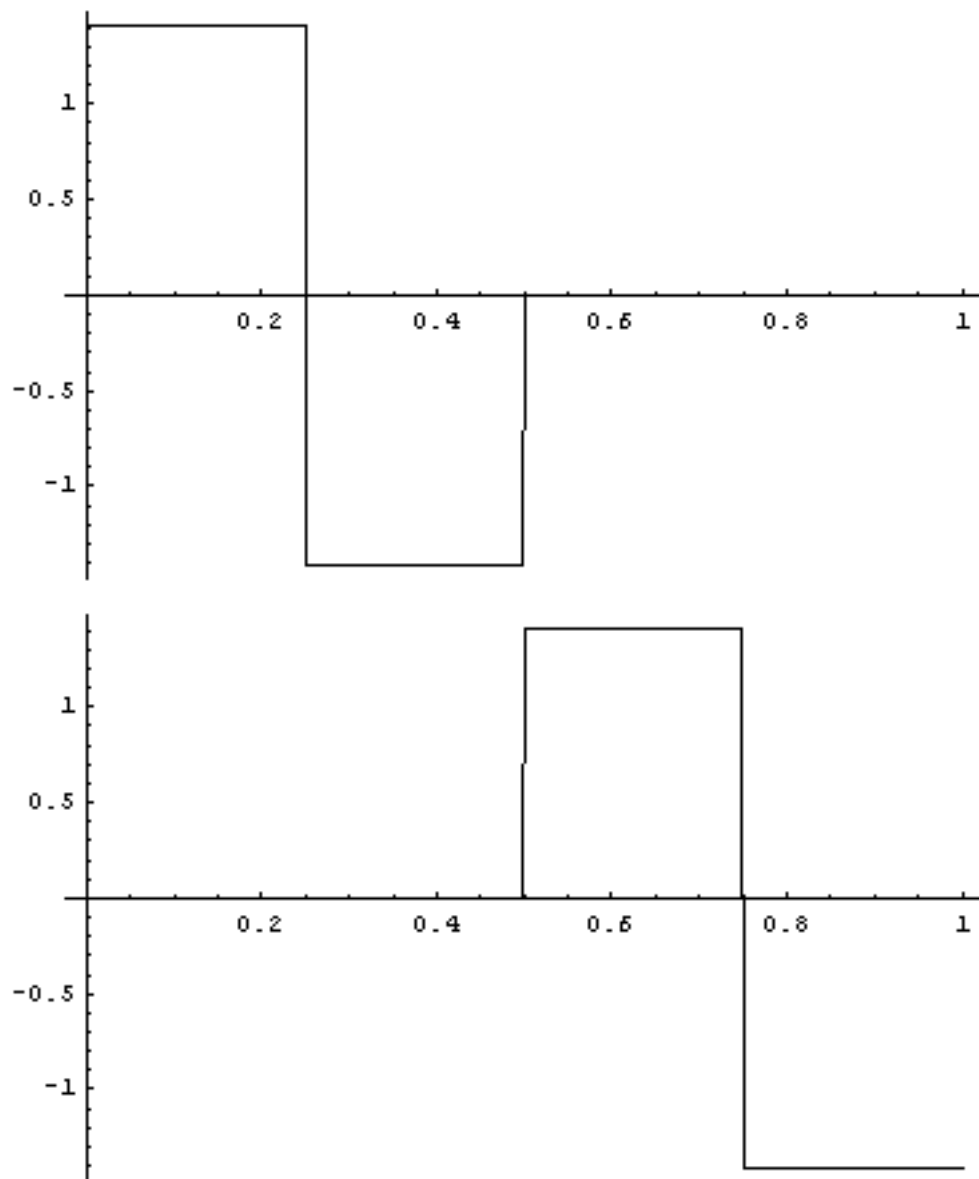


Figure 7: A plot of the Haar daughter wavelets.

data that is not necessarily periodic, or has frequency-dependant features (for instance, in the hyperspectral data, most of the differentiating features occurred in the near-infrared spectra, while the visible spectra were quite flat (as most rocks are grey)).

Haar wavelets were first proposed by Alfred Haar in 1909, and were later found to be the first in a family of wavelets discovered by Ingrid Daubechies (in that family it is called the D2 wavelet). We used Haar wavelets in our project for their simplicity in programming, given the short amount of time available.

Haar wavelets are typically used in simple image compression (the more sophisticated D4 wavelet is used in the JPEG2000 compression format). However, for our project, we tried using wavelets in two different ways. I will outline the first method, and go into more detail in the second.

We attempted to use wavelets to smooth the extremely noisy data produced by the HCI, but this was found to interfere with the feature extraction techniques used in the clustering algorithm. A possible solution which we did not get a chance to try out would be to subsample the wavelet-smoothed data to prevent the redundant values generated from the approximation from interfering with the feature extraction.

The second goal was aimed more at the problem of data storage, rather than that of processing. The HCI generates a tremendous amount of data per second, which must be stored until there is time to analyse it. Even if storage is not a problem (as storage media are becoming less expensive as time goes on), the transporting of that data still presents many issues. A possible solution is then to map the data onto its approximation. That is, for each pixel of the core, we store only the coefficients of the approximating functions. This requires a shift in how we consider the spectra: instead of a vector of spectral values (for each pixel of the core), lets consider it as a continuous function.

Approximation theory tells us that we can approximate any function with any family of orthogonal basis functions. Furthermore, we can compute the coefficients of this approximation using an integral:

$$\mathbf{a}_n = \frac{\left( \int_a^b \mathbf{f}(\mathbf{x}) \phi_n(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \right)}{\left( \int_a^b (\phi_n(\mathbf{x}))^2 w(\mathbf{x}) d\mathbf{x} \right)}$$

, by taking advantage of the orthogonality of the functions. If the basis functions are orthonormal, this is further simplified:

$$\mathbf{a}_n = \int_0^1 \mathbf{f}(\mathbf{x}) \phi_n(\mathbf{x}) d\mathbf{x}$$

However, our data occurs at discrete intervals, so we must split this integral up:

$$\mathbf{a}_n = \sum_i \mathbf{f}_i \int_{\mathbf{x}_{i-1}}^{\mathbf{x}_i} \phi_n(\mathbf{x}) \, d\mathbf{x}$$

We can represent this as a matrix operation. And here, we run into the first disadvantage of the Haar basis. The spectral bands that we are dealing with must be a power of 2. In our project, we had to reduce the 141 bands to 128 bands by discarding the outlying frequencies. Furthermore, the number of wavelets we use must also be a power of 2, if we are to keep the same resolution throughout our approximation.

The advantage of this approach is that the matrix is constant. The coefficients can be worked out nearly instantaneously, for all reasonable data sets. Our data set of 60 x 100 pixels and 128 spectral bands could be approximated in under a second.

To compute the coefficient vector for each pixel, we multiply the matrix with the spectral vector for that pixel. To compute the approximated data, we simply multiply the vector of coefficients by the transpose of the matrix.

Here is the Haar approximation of a specific pixel, with the approximation in blue, and the actual data in red.

The approximation was applied to each pixel of the data set (that is, the spectral data for each pixel was approximated). Here is a specific spectral band (in the near-infrared) across the core sample. Here is the same core sample, approximated. The colour values here indicate the intensity of this specific frequency. As can be seen, not much information is lost. However, we are storing only half of the original data if we store the coefficients associated with each pixel.

The approximation can also be performed on the fly, using a fast wavelet transform. The matrix method was used here as it is easier to manipulate and understand.

By comparison, I attempted to compute an approximation matrix for the Chebyshev functions of the first kind. Using the same computer, and to a similar order, the computation took over two hours.

### 3.2 The Future

We could improve on these results in several ways. Firstly, more sophisticated wavelet transforms and approximations could be used (loosely speaking, one could choose a wavelet that had a better 'fit' to the data). An orthonormal continuous wavelet with a compact support may yield better results than the wavelet used. Another method that was not tried would be to apply a wavelet transform in all three directions; that is, apply the transform to the spectra per pixel, and to the matrix of pixels given by each spectral band. Compression of the data would then be an encoding issue. The advantage of this approach

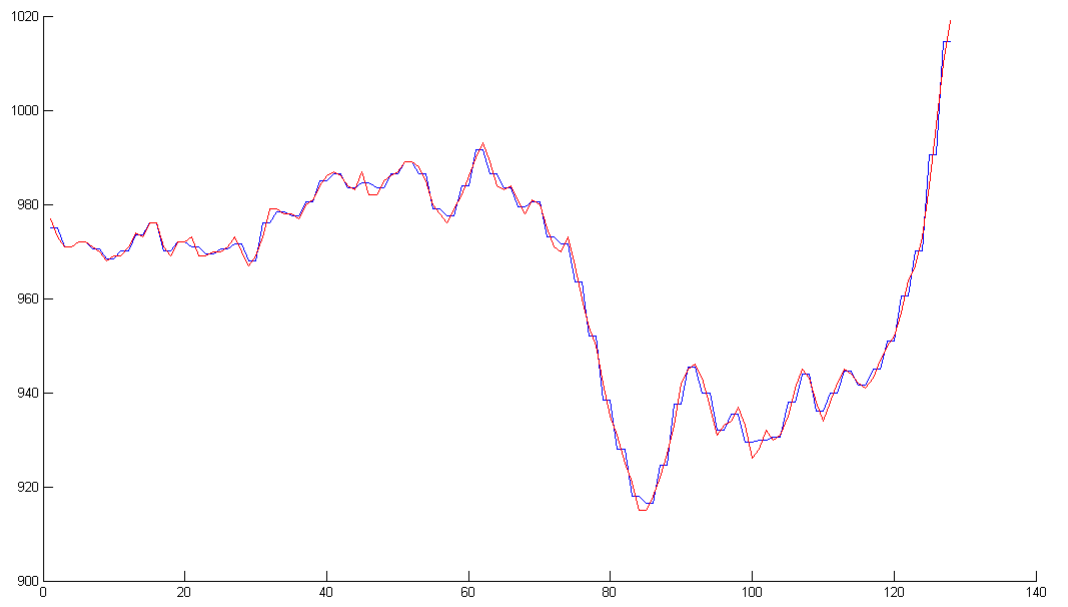


Figure 8: A plot of the haar approximation on the spectrum for a particular point.

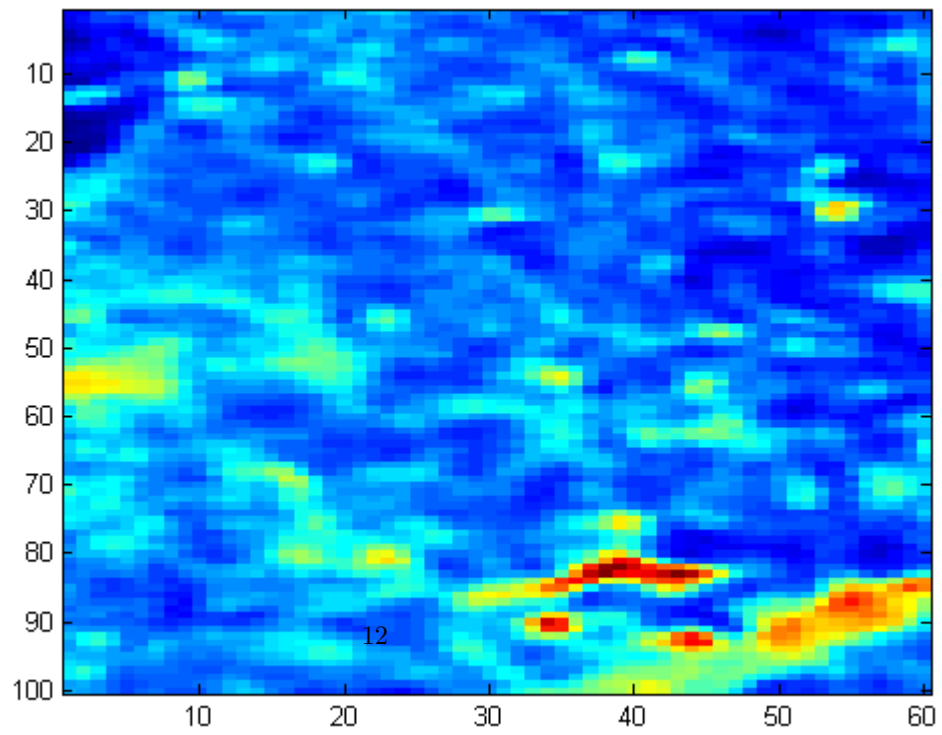
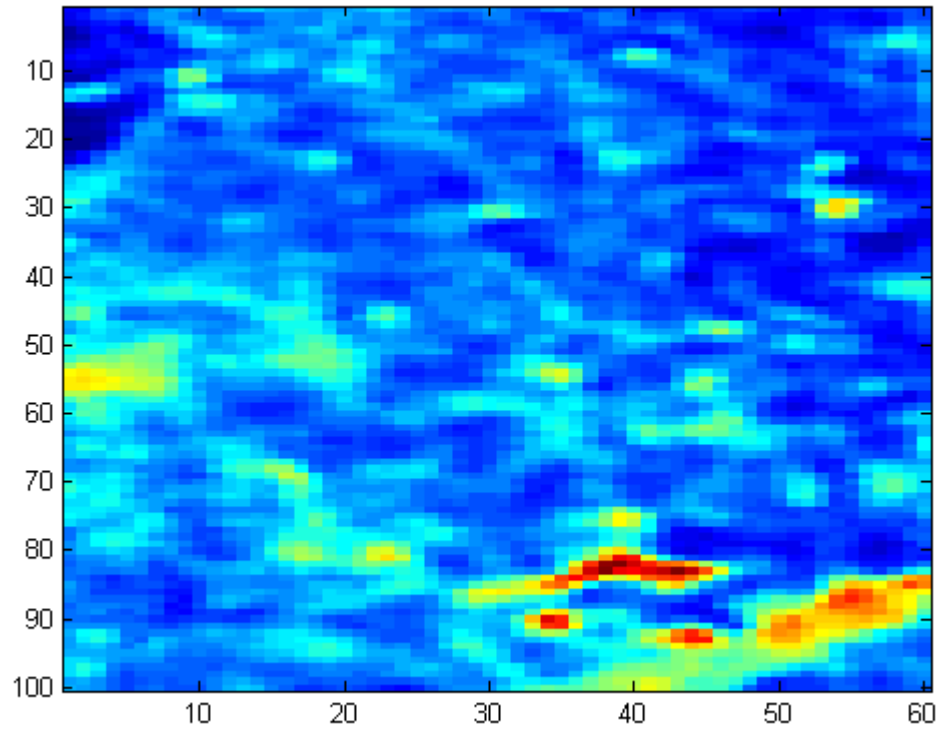


Figure 9: A comparison of the original and approximated data (approximated data below).

would be that lossless compression could be achieved (that is, the original data could be perfectly reconstructed).

## 4 Feature Extraction