POLITECNICO
DI TORINO

Master Degree in
Communication Computer network Engineering (CCME)
Department of Electronics and Telecommunications (DET)

Graph transforms for hyperspectral image compression

SeyedMohammad Jafari

* * * * * *

Supervisor
Prof. Enrico Magli

Politecnico di Torino
April 9, 2019

I hereby declare that, the contents and organisation of this thesis constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

SeyedMohammad Jafari

Turin, April 9, 2019

Summary

Hyperspectral Imaging is acquisition of variety of wavelengths spectral response per each pixel in an image. For example, human eye perceives visible light as a three band signal of low frequencies (red), medium frequencies (green), and high frequencies (blue). Beyond these three spectrum, hyperspectral images include vast number of spectral bands, result in a large volumetric data set. Similarly to video data signals from the 3D nature prospect, it is very important to encode efficiently hyperspectral images from the storage and transmission point of view as they are usually transmitted over space links or must be generated, processed, and stored in devices with limited resources.

In general, lossy coding of hyperspectral images is divided into two categories of fixed transforms, and content-aware adaptive transforms. These lossy compression methods are mostly transform coding based that means the input data is reversibly transformed in a new domain such that data content is more accurately reflects and also signal/data energy is more concentrated. Comparing to different conventional coding schemes, codding data based on Graph Signal Processing is a leveraging field of compression methods that has been shown to easily outperform traditional methods, but the overhead of graph transmission also outweigh the coding efficiency benefit.

Today's challenging data types being eligible to be represented on a graph, such as Hyperspectral images, Video, wireless sensor network, and social media data are growing increasingly and in this thesis, it is attempted to evaluate samples of hyperspectral images from the scratch to find the strongest correlation direction and signal sets to uncorrelate using Graph Signal Processing algorithms, and coming up with a sparse representation of image data on graph based on a graph structure which meets the trade of between sparseness-off image graph representation and final coding efficiency considering the transform graph to be transmitted as side-information to decoder.

Based on recent literature and research, from the set of Graph Signal Processing tool, Graph wavelet and a combination of it with Karhunen–Loève transform

(KLT) and JPEG2000 has been applied with good performance on hyperspectral images. In this thesis, Graph Fourier Transform(GFT) is used as transform in graph domain and a spectral vector created based on mean value of each spatial plane is used to learn the graph transform. Coding performance is determined in terms of rate-distortion and edge weigh metric performance as percentage of retrieved energy and PSNR of decoded image.

In this thesis, basic evaluations are done on a spectral vector GFT. To exploit correlation of data in both spatial and spectral dimensions, a 3D graph structure is proposed, and 2 different coding schemes based on this 3D structure are introduced and evaluated. 3D versions of the coder differ in updating spatial transform strategy and segmentation of water and sand areas spectral signature.

Based on the evaluations, using the given vector graph structure, rate-distortion is approximated based on bitplane entropy and it is shown how this scheme outperforms other schemes applied on the same dataset, if sufficient number of GFT coefficients are taken. in 3D mode with water area segmentation, taking more than 80 percent of coefficients, it outperforms all existing coding schemes and taking 60-75% of coefficients, it gradually degrades to the same performance of coding schemes based on GraphBior, KLT, DWT, and their combinations. However, proposed low-complexity 3D-GFT behaves very stable and has an acceptable competitive performance in low and high bitrate setting.

Acknowledgements

I would like to thank my thesis advisor prof. Enrico Magli at Politecnico di Torino who guided me to the right direction during whole process of my thesis work.
I must express my very profound gratitude to my parents for providing me with consistent support and continuous encouragement throughout my years of study and process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you
SeyedMohammad Jafari

I would like to dedicate this thesis to my loving parents

Contents

List of Figures x

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Objective . . . 2
- 1.3 Thesis outline . . . 3

2 Background 5
- 2.1 Graph transforms . . . 5
- 2.2 Compression of hyperspectral images . . . 6

3 Proposed approach 9
- 3.1 Metric explanation . . . 10
  - 3.1.1 Correlation: . . . 10
  - 3.1.2 Normalized correlation: . . . 11
  - 3.1.3 Structural Similarity Index (SSIM): . . . 12
- 3.2 Correlations evaluation . . . 13
  - 3.2.1 Correlation between two consecutive spectral bands . . . 14
  - 3.2.2 Correlation between every spectral band and first spectral band 16
  - 3.2.3 Correlation between every spatial column frames . . . 19
  - 3.2.4 Correlation between every spatial row frames . . . 21
  - 3.2.5 Correlation between random spectral vectors . . . 23
  - 3.2.6 Correlation between random spectral vectors and their vicinities 25
- 3.3 Differences of coded spectral vectors . . . 28
- 3.4 Discussion on changing trends of correlation in different directions . 28
- 3.5 Looking for decreasing trend in spectral direction . . . 28
  - 3.5.1 Possible reasons . . . 29
  - 3.5.2 Conclusion on decreasing trend of spectral correlations in reference to 1st band . . . 32
- 3.6 Edge weight metric performance . . . 32
  - 3.6.1 Percentage of energy, in function of percentage of retained coefficients . . . 33

3.6.2 PSNR in function of percentage of retained coefficients . . . 34
3.7 Dedicated GL-Vectors . . . 35
3.8 Quantization of coded coefficients . . . 37
3.8.1 Quantization of shared GL-vector . . . 41
3.9 Cauchy weight function . . . 42
3.10 Proposed 3D graph structure . . . 43
3.11 Encoding and decoding schemes . . . 44
3.11.1 Low complexity mode . . . 45
3.11.2 High complexity mode . . . 47

4 Results 51
4.1 Rate-distortion performance spectral vector GFT . . . 51
4.2 Performance of 3D-GFT schemes . . . 53

5 Conclusions 57

6 Important code Snippets 59

Bibliography 63

List of Figures

1.1 Naming of hyperspectral image dimensions scanning. 3

3.1 Cross correlation of two spectral bands in lag=0. 11
3.2 Normalized cross correlation of two spectral bands in lag=0. 12
3.3 Similarity index of two consecutive spectral bands. 13
3.4 Correlation of two consecutive spectral bands for both coded and uncoded images 15
3.5 Similarity of two consecutive spectral bands for both coded and uncoded images 16
3.6 Correlation of spectral bands and first spectral band for both coded and uncoded images. . . 17
3.7 Similarity of spectral bands and first spectral band for both coded and uncoded images. . . 18
3.8 Correlation of two consecutive spatial columns for both coded and uncoded images. 20
3.9 Similarity of two consecutive spatial columns for both coded and uncoded images. 21
3.10 Correlation of two consecutive spatial rows for both coded and uncoded images. 22
3.11 Similarity of two consecutive spatial rows for both coded and uncoded images. 23
3.12 Correlation of two random spectral vectors for both coded and uncoded images. 24
3.13 Correlation of random spectral vectors and their vicinity of image $sc0_{raw}$. 26
3.14 Correlation of random spectral vectors and their vicinity of image *airs9*. 27
3.15 2D spectral Cross-correlation shift from center, image for Sc0raw . . 30
3.16 Correlation coefficient of spectral bands with reference to 1st band using xcorr command 31
3.17 Pearson Correlation Coefficient of spectral bands with reference to 1st band 32
3.18 Percentage of energy, in function of percentage of retained unquantized coefficients using mean spectral vector to learn GFT matrix . 34
3.19 PSNR in function of percentage of retained unquantized coefficients using mean spectral vector to learn GFT matrix 35
3.20 PSNR vs percentages of coefficients using dedicated GL-vectors for scene0 Yellowstone uncalibrated 36
3.21 Percentage retained energy vs percentages of coefficients using dedicated GL-vectors for scene0 Yellowstone uncalibrated 37
3.22 PSNR of quantized coded image using dedicated GL-vectors for scene0 Yellowstone uncalibrated 38

3.23 Percentage retained energy of quantized coded image using dedicated GL-vectors for scene0 Yellowstone uncalibrated 39
3.24 PSNR of quantized coded image using shared GL-vectors for scene0 Yellowstone uncalibrated 40
3.25 Percentage retained energy of quantized coded image using shared GL-vectors for scene0 Yellowstone uncalibrated 40
3.26 PSNR of quantized coded image and GL-vector using shared GL-vectors for scene0 Yellowstone uncalibrated 41
3.27 Percentage retained energy of quantized coded image and GL-vector using shared GL-vectors for scene0 Yellowstone uncalibrated 42
3.28 PSNR in function of percentage of retained coefficients using Cauchy and Gaussian weighting function 43
3.29 Percentage of energy in function of percentage of retained coefficients using Cauchy and Gaussian weighting function 43
3.30 3D graph structure 44
3.31 Low complexity 3D graph coding scheme 46
3.32 High complexity 3D graph coding scheme 48
3.33 Difference spectral signature sand and water 49

4.1 Bitrate VS Percentage coefficient using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated 52
4.2 Bitrate VS PSNR using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated 53
4.3 Bitrate VS Percentage energy retained using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated 53
4.4 Percent preserved energy VS percent coefficients for 3D and vector GFT coding schemes 54
4.5 PSNR VS percent coefficients for 3D and vector GFT coding schemes 55
4.6 Rate-distortion for 3D and vector GFT coding schemes 55

5.1 Comparison of GFT with shared mean GL-vector with other coding schemes on scene0 Yellowstone uncalibrated 57
5.2 Comparison of 3D GFT coders with other coding schemes on scene0 Yellowstone uncalibrated 58

Chapter 1

Introduction

1.1 Motivation

Coding of huge volumetric data has always been on demand for different data types. There is a vast majority of data types being eligible to be represented, processed and consecutively compressed using GSP methods. These data types are not only becoming more popular due to increasing amount of data generated by wireless sensors, social networks, videos and satellites with vast application in agriculture, eye care, food processing, mineralogy, surveillance, astronomy, chemical imaging, and environment also very important to transmit and store in an efficient way. Considering a spectroscopy application of a fairly small map area, or transmitting or locally storing hyperspectral images on satellite links, compression of these images using a method providing random access and excellent coding performance with low complexity, considerably reduces the power consumption of space links and satellites.

In general, when a GFT is to be considered as transform, two key problems arise to be measured for such a compression scheme. Side information overhead introduced by graph transform and complexity of transform. A GFT of a line graph is equivalent to 1D DCT, and 2D DCT is one possible transform matrix for 2D GFT. Complexity of the encoder strongly relays on the number of individual GFT matrices to be calculated. Therefore, a constant effort is always made by researchers to find a sparse light transform graph which preserves image quality using a reasonable bitrate.

Moreover, since GFT and other GSP transform has been shown to preserve discontinuities better than conventional codding schemes, graph based compression is well received by researchers. On 2D images, it has been proved [11] that predictive GFT compression outperforms other compression methods. There it drives the motivation to test and implement GFT compression scheme for 3D hyperspectral images as well, and check how good it performs with a reasonable compromise on complexity and transform graph structure.

It is worthy to mention that in this thesis, basic implementation of GFT on hyperspectral images is assessed and mixing GFT with other sophisticated coding schemes on different dimensionality like in [23] to improve compression performance is neglected for the first parts where vector spectral graphs are taken as signals to decorrelate. On the other hand, a 3D GFT graph structure and 2 different coding schemes are proposed and evaluated to exploit correlation in both spectral and spatial dimensions. 3D coders benefit from coupling with JPEG2000 compression for some parts where integrity and SNR of reference layers are important .These different approaches to select a sparse transform graph and corresponding compression performance in terms of rate-distortion metric are finally evaluated.

1.2 Objective

Goal of this thesis is to evaluate the coding performance of GFT on hyperspectral images. It has been shown in many cases that GFT benefits from the better sparse representation of discontinuities and therefore very sparse representation of image on a graph signal structure. Although, the transform graph coding and its transmission to decoder side, cannot be neglected, since it introduces a large amount of overhead and easily outweighs the coding efficiency. In addition, to the best of our knowledge, a simple GFT has not been tested to compress hyperspectral images up to the date of publishing this thesis. Accordingly, there exists no knowledge about the complication of choosing a good graph transform in this type of images, coding performance on natural and piece-wise smooth images, and compression performance of simple GFT coder on hyperspectral images without segmentation of image in spectral or spatial dimension. In figure 1.1 naming of different dimensions and processing directions are shown which are going to be used in this thesis.

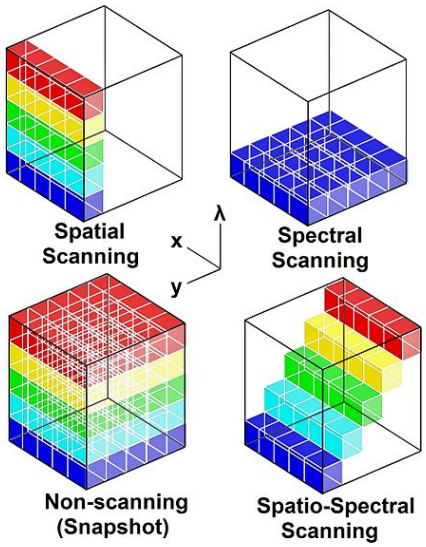


Figure 1.1: Naming of hyperspectral image dimensions scanning.
Source: https://en.wikipedia.org/wiki/File:AcquisitionTechniques.jpg

During the implementation of GFT for hyperspectral images, a fast bitplane encoder is developed to estimate the corresponding bitrate for different coding schemes.

1.3 Thesis outline

This thesis is written in 5 chapters. In chapter 2, background of the subject is given in two main section. First, Graph signal processing and specifically Graph Fourier Transform is introduced. It follows with a section on Hyperspectral image basics and hyperspectral image compression.

In chapter 3, firstly correlation of hyperspectral image samples in different dimensions are evaluated. Then relation and change trend of correlation in different directions are discussed. This correlation evaluation is specially important to choose a direction of a vector/matrix as a basis to learn the graph transform. Also, proposed approach to produce the transform graph and use it to find a sparse graph representation of hyperspectral images are developed.
Also, some intermediate calculations regarding the coding efficiency and performance of coding schemes are done in terms of edge weigh metric performance as percentage of retrieved energy and PSNR.
In addition, Cauchy weight function is compared regarding the edge weight metrics to Gaussian Function. At the end of this chapter, a 3D graph structure is proposed

which is followed by 2 coding schemes to exploit this 3D transform for the sake of exploiting correlation on both spectral and spatial dimensions.

In chapter 4, result of discussed coding schemes in chapter 3 are reported regarding the compression performance. This is basically determined in terms of rate-distortion corresponding to different selection of transform graph and different percentage of codded coefficients taken into account as a metric to control amount of lossyness. These results are reported separately for vector GFT and 3D GFT transform coders.

In the last chapter 5, conclusion and a general comparison made between the compression scheme proposed in this thesis and other recent compression performed by other researchers on the same dataset.

Chapter 2

Background

2.1 Graph transforms

Field of Signal Processing on Graphs founded in 2013 [20] and since then, well received by researchers and industries due to interesting capability of handling high-dimensional data. Nowadays, crazing increase in the number of high-dimensional data where naturally rise the capacity to be represented on vertices and nodes of a weighted graph demands to investigate the application of GSP such as GFT and Graph wavelet transform on eligible datasets and discover the possible potential use cases. Application of Graph Signal Processing has still been warmly welcomed and catch interests as it has been deeply discussed in [17]. Ability of transforming high-dimensional data to a sparse graph representation using Graph Transform tools is also considered by original authors [20] in a survey on recent transform based compression schemes [19].
Since we are addressing lossy compression, transform coding is used to transform data in new sparse representation domain. According to vast application of Fourier Transform in time domain, this transform is also introduced in graph domain and called GFT.
A graph is composed of nodes (vertices) and Edges. Let us set Graph $G = (V, E)$ where $|V| = N$ is a set of nodes and $E \subset V \times V$ is the set of corresponding edges between these nodes. Each edge of this graph can be associated with a weight which describes the similarity and connection between two end nodes of this edge. Therefore, an image as a 2D signal, can be represented on a 2D graph assuming that each pixel of the image is a node of the graph which is connected to its 4 neighbours. Edge weight of the graph is calculated based on the similarity of the neighbour pixels (nodes) using either Cauchy function 2.1 or Gaussian function 2.2

$$\text{Cauchy function:} \omega_{ij} = \frac{1}{1 + (\frac{d_{ij}}{\alpha})^2} \tag{2.1}$$

$$\text{Gaussian function:}\omega_{ij} = e^{-\frac{d_{ij}^2}{\sigma^2}} \tag{2.2}$$

where d_{ij} is the Euclidean distance between pixel i and j ($d_{ij} = |f_i - f_j|$) and α and σ are parameter to set weight sensitivity to intensity difference as explained in [5].

$W \in \Re^{N \times N}$ is the matrix of all weights called adjacency matrix. Therefore, a graph and an image can be represented using its adjacency matrix where ω_{ij} represents the weight of the edge between nodes i and j and $\omega_{ij} = 0$ if there is no edge (connection) between to nodes.
The graph Laplacian is defined as $L = D - W$ where D is a diagonal matrix called degree matrix. The ith diagonal element d_i in D is the sum of all edges incident to node i. Accordingly, signal $f : V \to \Re$ on N nodes of graph, can be represented by vector $\mathrm{f} \in \Re^N$ where the ith component of vector f stands for signal value at the ith position vertex in V.

In graph domain, equivalent of Fourier Transform in time domain is called Graph Fourier Transform. GFT $\widehat{\mathrm{f}}$ of given signal $\mathrm{f} \in \Re^N$ is defined as:

$$\widehat{\mathrm{f}} = U\mathrm{f} \tag{2.3}$$

where U is the matrix whose rows are eigenvectors of the graph Laplacian L. Inverse graph Fourier Transform is also given as:

$$\mathrm{f} = U^T\widehat{\mathrm{f}} \tag{2.4}$$

It is worthy to mention that if λ_is are eigenvalues and u_is are eigenvectors of the Laplacian matrix L, those can be consequently assumed as frequencies and variation across the graph. The set $\{\lambda_0, \lambda_1, ..., \lambda_{N-1}\}$ is the spectrum of Laplacian graph.

2.2 Compression of hyperspectral images

In general, coding of hyperspectral images [22, 9, 18, 10, 24, 7, 8, 23] is divided into two categories of content-aware adaptive compressions and fixed methods. First, coding schemes that are not content aware and the transform basis is fixed like Discrete Wavelet Transform (DWT). This category of coders do not adapt the transform basis to the nature of input signal and regardless of the signal they use a same transform. These category of compression methods can typically be

implemented in low complexity. On the other hand, there are signal adaptive compression methods that adapt transform basis to get a sparser representation of the signal like Karhunen–Loève transform (KLT). This category basically suffers from computation complexity introduced by adaptivity ability.

In graph domain, compression schemes using GSP proposed like in [23] where graph wavelet transform mixture with KLT and JPEG2000 is used on different pair selection of dimensions to achieve good rate-distortion. In the same manner, GFT can be exploited in compression of hyperspectral images which additionally benefits from the adaptivity to the input signal nature referring to the production of transform matrix basis. If Fourier Transform matrix/vector is selected per input signal, maximum adaptivity is then employed.

Given that the correlation in spectral direction is more than spatial frames, It is important to select the signal chunks and transform (uncorrelate) them in this direction. Although, there is an option to perform separate coding in spectral and spatial direction when talking about hyperspectral images. For example, a 1D transform can be applied in spectral dimension and then a 2D transform can be used to uncorrelate spatial data. It is important, to always exploit the better (signal-adaptive here) compression method in the direction carries bigger correlation (spectral dimension here), and let the other directions being encoded using simpler coding schemes. However, in this thesis avoiding complexity of separate compression on different dimensions, a GFT is applied on the spectral dimension as it is proved in following chapters, that correlation is more significant in spectral dimension.

Apart from exclusive effort made on lossless compression of data in order to preserve their exact quality, a lossy compression method allows for higher compression ratio and more flexibility to use transform based compression approaches to make a scalable compression method.

So far, it has been shown that using variations of KLT in spectral domain coupled with JPEG2000 [13] or other simple compression techniques achieve best rate-distortion on hyperspectral images. In order to reduce KLT complexity, many other variations of KLT like spatially sub-sampled KLT [18], low complexity KLT [12], spectrally sub-sampled KLT [7, 8], and also specially crafted transform [2] are proposed and shown to have an acceptable compression performance as transform based lossy compression methods. In addition, substituting KLT with DWT [3, 16, 14], or DCT [1, 15] followed by spatial compression is investigated.

In all mentioned compression schemes, it is well determined, how smooth is hyperspectral image in different regions. A piecewise smooth image allows for a clever

selection of shared adapted transform basis in case that the transform is content-aware. To perceive the regions of smoothness, a correlation discovery amongst spectral dimension can be done. Additionally, in spatial domain a simple segmentation procedure must be taken to segment areas with similar or the same material wavelength.

In the following chapter, we firstly investigate the correlation of hyperspectral images in different directions spatially and spectrally. The correlation and changes in correlation is being discussed to gain a better understanding of image nature. Then a graph Fourier Transform is applied on the selection of image data. Then amount of de-correlation will be examined by calculating the correlation in the same direction on encoded image. De-correlated image then undergoes three examination of percentage of preserved energy with reference to original image, PSNR, and bitrate. As a handle to scale the lossyness, a percentage of GFT coefficients will be taken from encoded data and the rest is cut away (set to zero). Therefore, aforementioned metrics are evaluated in function of percentage of taken GFT coefficients. AT final stage, a quick approximation of bitrate is made on coded data to predict the achievable bitrate using an optimal entropy codding scheme. Results are then related to former PSNR evaluations and a rate-distortion curve will be calculated. The coding performance of this scheme will be finally compared to other known schemes applied on the same scene.

For the sake of better comparison and well standardization, all evaluations in this thesis are performed on scenes from AVIRIS scene 0 and scene 9 of Yellowstone National Park in USA.

Chapter 3

Proposed approach

In this chapter, Firstly a GFT using a concatenation of eigenvectors of Laplacian matrix is implemented. Laplacian matrix is learned per each signal in this run and it is used uniquely for each signal to perform the GFT to achieve the sparsest representation in Graph Fourier domain. The GFT is applied on each spectral vector given all spatial pixel indices.
Images under experiment, are $sc0_{raw}$ and *airs9* which are originally in 16-bit unsigned format. The correlation between different pairs of vectors and frames, before and after applying GFT are going to determined.

Then, we discuss a bit about different possible metrics to asses correlation between image signals. Since, it is crucial to know the amount of correlation remained between image data after applying our transform based compression method, to decide about the compensation could made in terms of transform matrix to lower the overhead of its transmission, in expense of worse SNR.

In section 3.7, impact of taking dedicated Gaussian Laplacian vectors per each spectral vector $x(i, j, :)$ is going to be assessed, where i and j are spacial indices. Although, it is obvious in advanced that it is not making sense to take individual GL-Vectors[1] since then all GL-Vectors must be handed in(included) to decoder for decoding process as side-information.
In the next section 3.8, we take one more step toward quantization of coded data and decide which data must be quantized and which not. The effect of quantization on coded images using shared mean GL-vector and dedicated GL-vectors will be evaluated in terms of codding performance.
In continue, a short evaluation of using Cauchy weight function instead of Gaussian on a sample image is done and results are compared accordingly. At the end

[1]Gaussian Laplacian vectors

of this chapter, final 3D graph structure which exploits both spectral and spatial correlation of data is introduced. This is followed by explanation of 2 proposed encoding and decoding schemes for 3D GFT transform coder. 3D versions of the coder differ in the sense of updating spatial transform strategy and segmentation of water and sand areas spectral signature.

3.1 Metric explanation

Basically, there are 3 possible correlation related metrics for image signals:

1. Correlation

2. Normalized correlation

3. Structural Similarity Index (SSIM)

3.1.1 Correlation:

$$R_{xy}(m) = E\{x_{n+m}.y_n^{\star}\} \tag{3.1}$$

$R_{xy}(m)$ Is cross-correlation of two discrete-time sequences, x and y. Cross-correlation measures the similarity between x and shifted (lagged) copies of y as a function of the lag. Since, we are looking for similarity of two instance of still images (either vector or frame), the lag zero is only important. Because, other lag values stand for similarities for shifted versions of two signals in different directions over each other. Typically, for 2D data, `xcorr2` MATLAB command is used, which can shift in X and Y directions independently. But here, it is not useful since we only want to calculate correlation at zero lag. The fastest way, would be using `xcorr` MATLAB function with lag limited to zero only and vectorize our 2D data using `data(:)` syntax. So, for both 2D and vector data we use syntax: $xcorr(x(:), y(:))$ to compute correlation. The following figure, shows the correlations between every two consecutive spectral bands of a hyper-spectral image. These frame bands can be shown as $x(:, :, i)$ and $y(:, :, i + 1)$ in hyper-spectral indexing.
In this thesis, all correlations are evaluated using MATLAB command $xcorr(x(:), y(:),0)$ which returns cross-correlation of two vector signals x and y with lag zero.

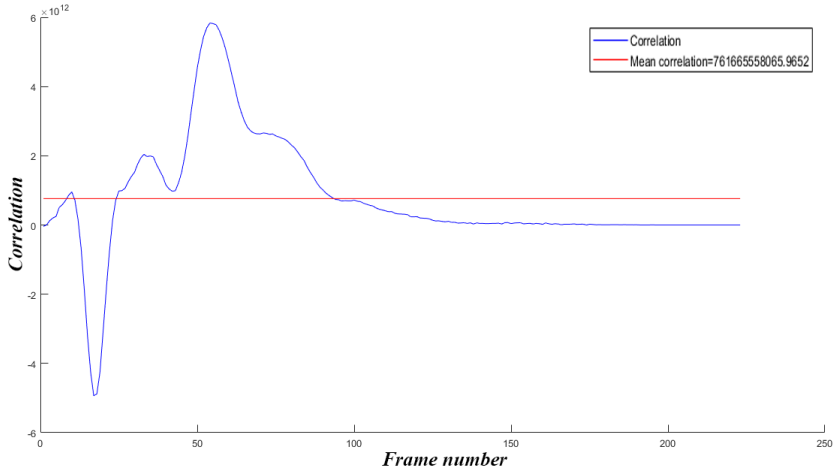


Figure 3.1: Cross correlation of two spectral bands in lag=0.

3.1.2 Normalized correlation:

$$R_{xy,coeff}(m) = \frac{R_{xy}(m)}{\sqrt{R_{xx}(0)R_{yy}(0)}} \tag{3.2}$$

Normalized correlation, normalizes the sequence (for different lags) so that the auto-correlations at zero lag equals to 1. Being known also as correlation coefficients which lies between normalized range of zero to one. It is useful to normalize the correlation values because it is derived by products of signal element amplitudes. Which is, generally not known for every signal without having a look at original range of signal values. But, one must take care when wants to compare to cross correlations. Because, each cross correlation is normalized using zero lag value of itself, and comparing them in normalized span does not make sense. For example, one cross correlation could be in range hundreds but other one may be in range thousands, and after normalization, both are in range zero to one. The figure below, demonstrates normalized cross correlation of two consecutive spectral bands of a hyper-spectral image. Although, they look similarly, The difference around band number 200, arises due to the normalization factor described before. It means, that correlation values were considerably less in comparison to other values in those bands, but after normalization, since their auto-correlations were less as well, they have been divided by smaller normalization factor and look larger in normalized domain. This is the reason to not using normalized/coefficients for comparison applications.

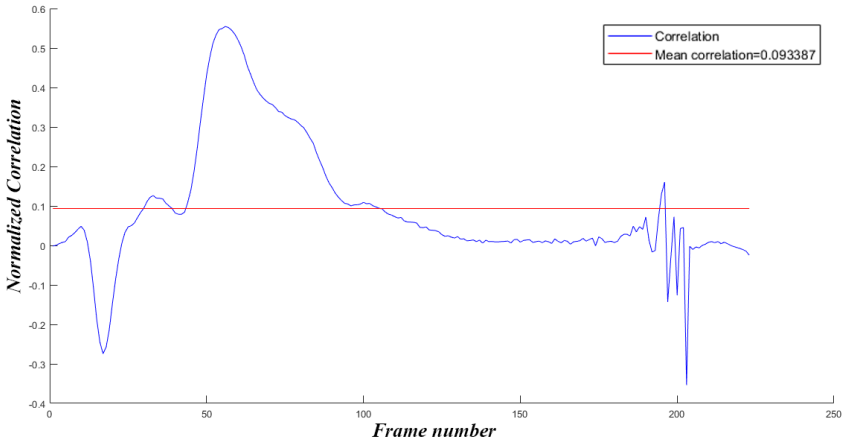


Figure 3.2: Normalized cross correlation of two spectral bands in lag=0.

3.1.3 Structural Similarity Index (SSIM):

It is basically introduced for measuring image quality after applying transforms and filters. But, it could be useful in our context to have a look at it. because, due to its mathematical definition, it returns normalized results similar to normalized correlation, but these values are comparable to each other. It is due to a generally meaningful normalization factor being used to evaluate SSIM. The MATLAB command `ssim` is used to calculate similarity factor between signals. However, it is not completely reasonable metric to reveal similarities between different signal values. Because it is more a structural based measure, but because of having a normalized-easy to compare measure metric, this metric is going to be used too.
SSIM is based on the computation of three terms, namely the luminance term, the contrast term and the structural term. The overall index is a multiplicative combination of the three terms.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.3}$$

Where, μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, and y. The figure below, shows the similarity index for two consecutive spectral bands. Note that, it is following the correlation values like in figure 3.1, but does not have the normalization problem, like normalized cross correlation.

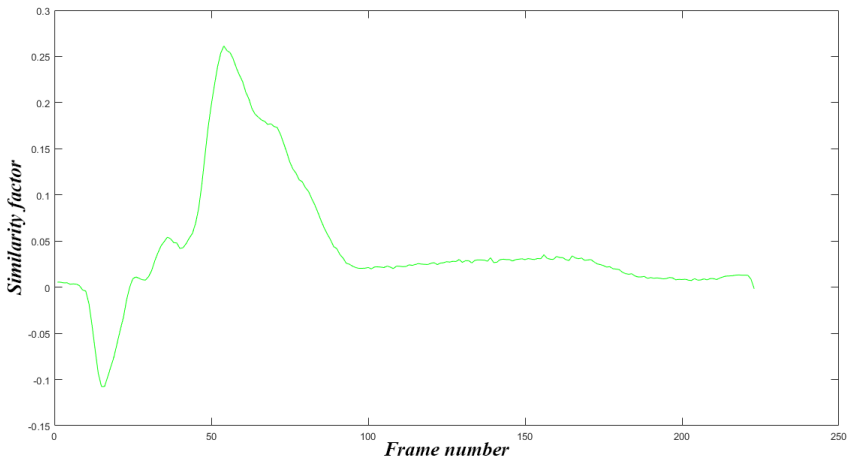


Figure 3.3: Similarity index of two consecutive spectral bands.

3.2 Correlations evaluation

In this section, the correlation between different parts of hyper-spectral image described in bellow scenarios are evaluated.

1. Correlation between two **consecutive** spectral bands. 3.2.1
   These bands are shown as $x(:,:,i)$ and $x(:,:,i+1)$ in MATLAB hyper-spectral matrix indexing, where i stands for index of spectral band. So, we call it spectral correlation from now on since it has been taken between two spectral bands.

2. Correlation between every spectral band and first spectral band.3.2.2
   This means correlation of each spectral band $x(:,:,i)$ vs first spectral band $x(:,:,1)$ is evaluated.

3. Correlation between consecutive spatial column frames.3.2.3
   It means, correlation is calculated between consecutive spatial columns $x(:,i,:)$ and $x(:,i+1,:)$ where i is the index of column in spatial domain.

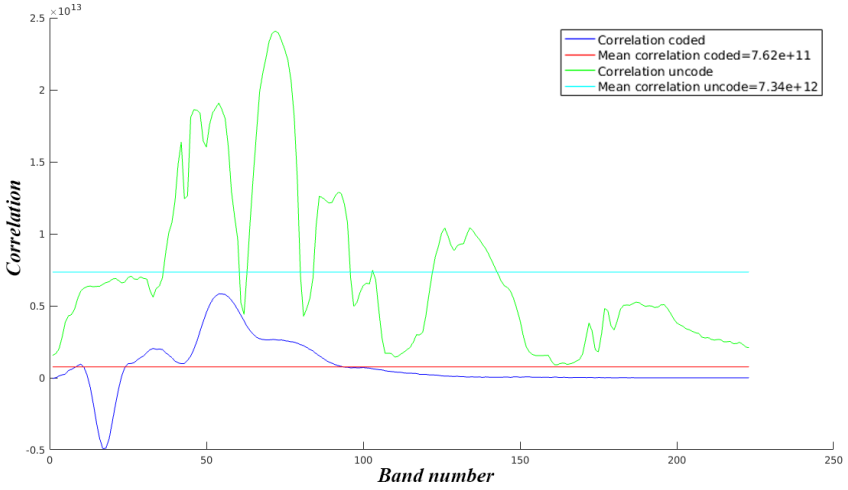
4. Correlation between consecutive spatial row frames.3.2.4
   It means, correlation is calculated between consecutive spatial rows $x(i,:,:)$ and $x(i+1,:,:)$ where i is index of row in spatial domain.

5. Correlation between random spectral vectors.3.2.5
   It means that random number of spectral vectors, shown as $x(i, j, :)$ in MATLAB hyper-spectral indexing, are taken. Indices, i, j stand for spatial index in horizontal and vertical coordinates. Then, correlation between all these vectors were calculated. Number of random vectors are 50, accordingly $C_2^{50} = 1225$ correlations were calculated between all spectral random vectors.

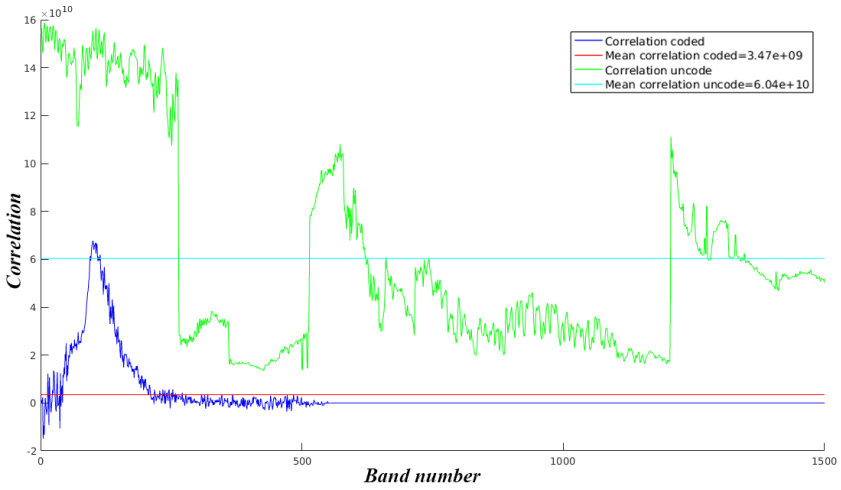
6. Correlation between random spectral vectors and their vicinities.3.2.6
   The same as previous scenario, but correlation is calculated between spectral vectors $x(i, j, :)$ and their spatial vicinity of size 32 by 32. It means,in spatial indexing domain $x(i \pm 16, j \pm 16, :)$ responds to a rectangular vicinity of 32 by 32. Number of $33^2 = 1089$ correlations for every one of 50 random vectors are calculates. It is 33 to the power of two because there are $2 * 16 + 1 = 33$ row and column on each rectangular vicinity, where given spectral vector as reference, lies in center of rectangle.

3.2.1 Correlation between two consecutive spectral bands

The correlation of consecutive spectral bands for coded and uncoded image of $sc0_{raw}$ is demonstrated in figure 3.4a. The same values for *airs9* represented in figure 3.4b.These bands are shown $x(:, :, i)$ and $x(:, :, i+1)$ in MATLAB hyper-spectral matrix indexing. Applying GFT have reduced correlation by factor of 10 in $sc0_{raw}$ and by factor of 17 in *airs9*. But still, considerable amount of correlation remained. It is noticeable that for both coded and uncoded images, correlation decreases in second half of spectral bands. This degradation is more noticeable in coded data. Although, the average correlation for $sc0_{raw}$ after GFT is in terms of $7.3 * 10^{11}$ and $6.3 * 10^{10}$ for *airs9*, which is still noticeable correlation leftover.



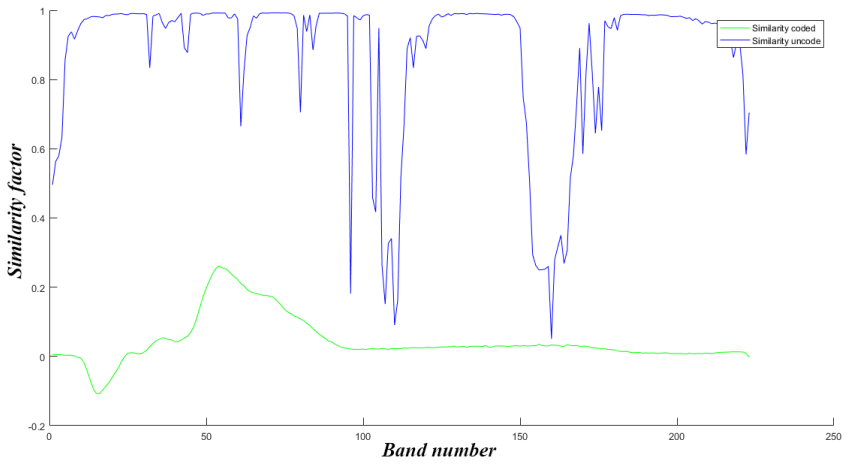
(a) $sc0_{raw}$.



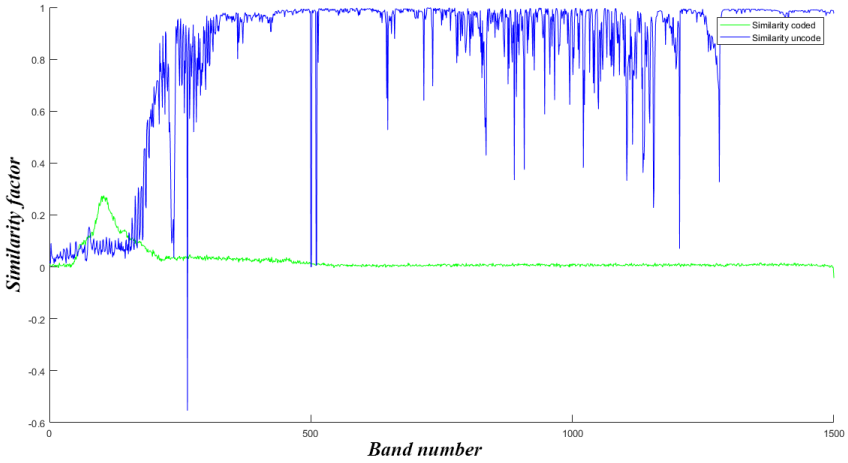
(b) *airs9*.

Figure 3.4: Correlation of two consecutive spectral bands for both coded and uncoded images

To clarify the amount of similarity in coded and uncoded images,similarity index for both $sc0_{raw}$ and *airs9* are present below. Similarity of uncoded consecutive spectral bands for $sc0_{raw}$ in figure 3.5a, similarity of uncoded consecutive spectral bands for *airs9* in figure 3.5b are shown. Structurally, these bands were so similar initially, but the similarity can be said to be vanished significantly after applying GFT for both cases.



(a) $sc0_{raw}$.



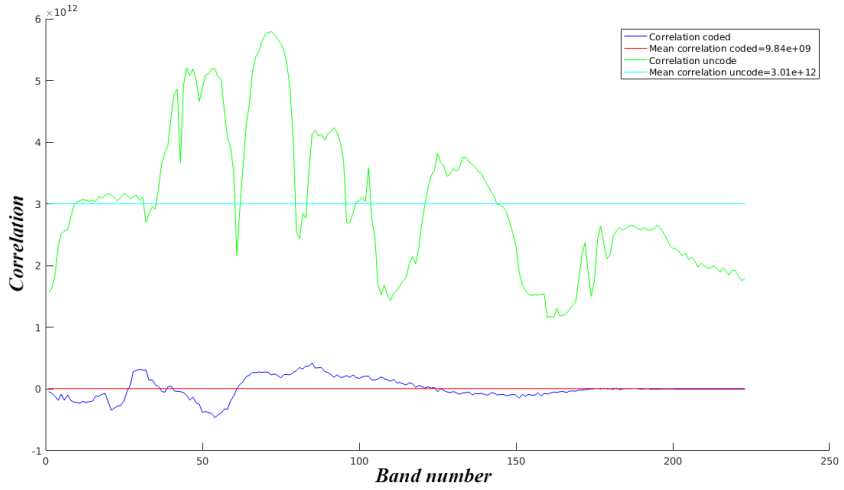
(b) *airs9*.

Figure 3.5: Similarity of two consecutive spectral bands for both coded and uncoded images

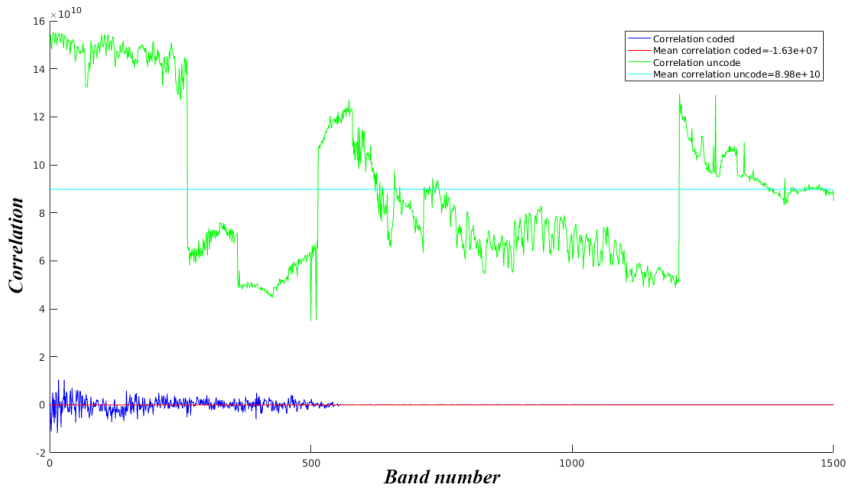
3.2.2 Correlation between every spectral band and first spectral band

The correlation of spectral bands with respect to 1st spectral band, demonstrated for both images in coded and uncoded state. This means correlation of each spectral band $x(:,:,i)$ vs first spectral band $x(:,:,1)$ is evaluated. As expected, increasing the index of spectral band, the amount of correlation dropped, while increasing the band number, similarity is decreasing. Correlation for $sc0_{raw}$

presents in figure 3.6a and for *airs9* in figure 3.6b. Correlation after applying GFT dropped by factor of 304 for $sc0_{raw}$ and equals to $9.84 * 10^9$ in average, and dropped by factor of 5377 for *airs9* and it is $-1.63 * 10^7$ after applying GFT in average.



(a) $sc0_{raw}$.

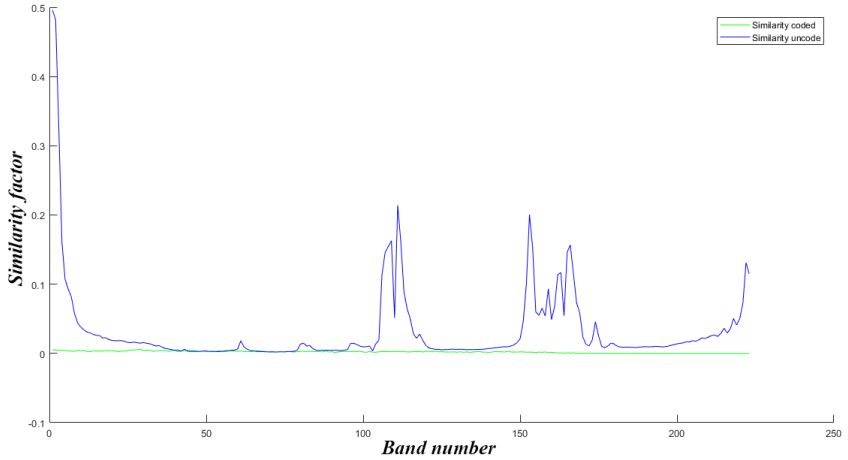


(b) *airs9*.

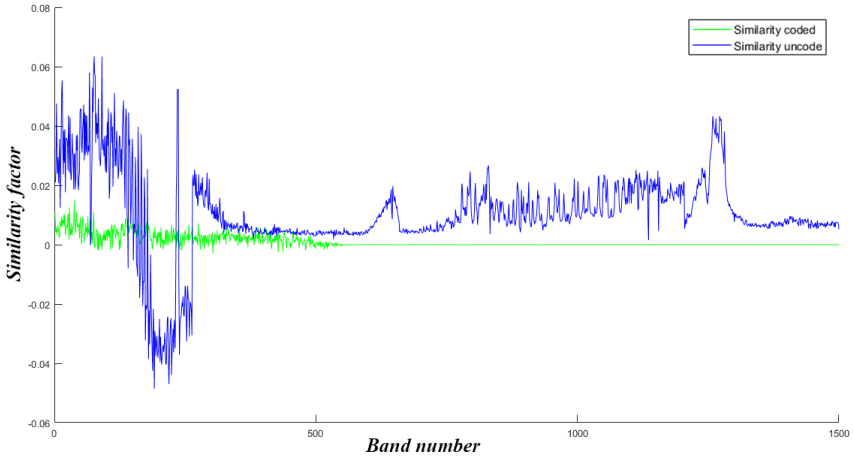
Figure 3.6: Correlation of spectral bands and first spectral band for both coded and uncoded images.

Similarity index metrics, are also available below for both images. Figure 3.7a for $sc0_{raw}$ and figure 3.7b for *airs9*. Apparently, it is less than consecutive case, but for images *airs9* it is too less in average around 0.02 for both coded and uncoded

images. Expected drop by getting far from 1st band occurred in first half of both images, but it recovered in second half which is a sign for similarity of far bands from 1st band. It can be advantageous, because it means relying on first half bands for learning transfer matrix, could be enough in these cases.



(a) $sc0_{raw}$.

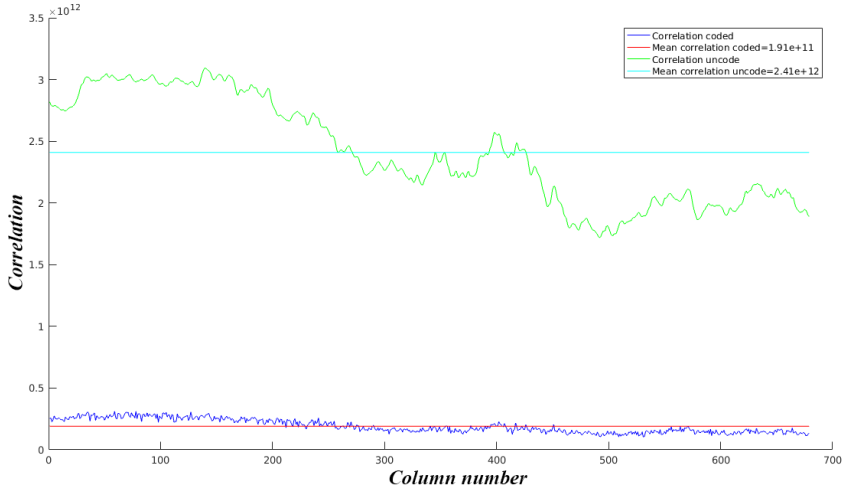


(b) *airs9*.

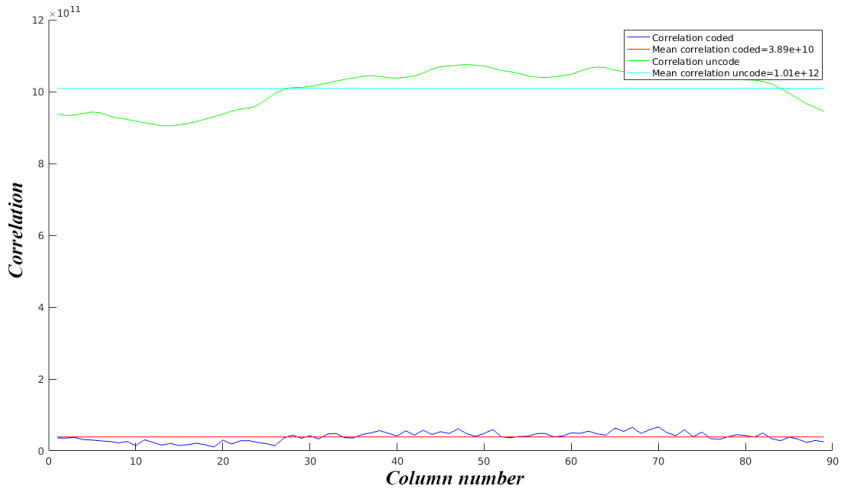
Figure 3.7: Similarity of spectral bands and first spectral band for both coded and uncoded images.

3.2.3 Correlation between every spatial column frames

In this section, correlation is calculated between consecutive spatial column vectors for both coded and uncoded images. This correlation is calculated between consecutive spatial column frames $x(:, i, :)$ and $x(:, i+1, :)$ where i is index of column in spatial domain. For $sc0_{raw}$ figure 3.8a, uncoded image, in average correlation is $2.4 * 10^{12}$ and after applying GFT it decreased to $1.9 * 10^{11}$. So, it dropped by factor of 12.6 which is comparable with case of consecutive spectral bands, but not considerable with case of spectral bands vs 1st band. For *airs9* figure 3.8b, mean correlation before coding is $1.01 * 10^{12}$ and drops to $3.89 * 10^{10}$ by factor of 26. In total, correlation and de-correlation factor is a bit smaller in this direction.



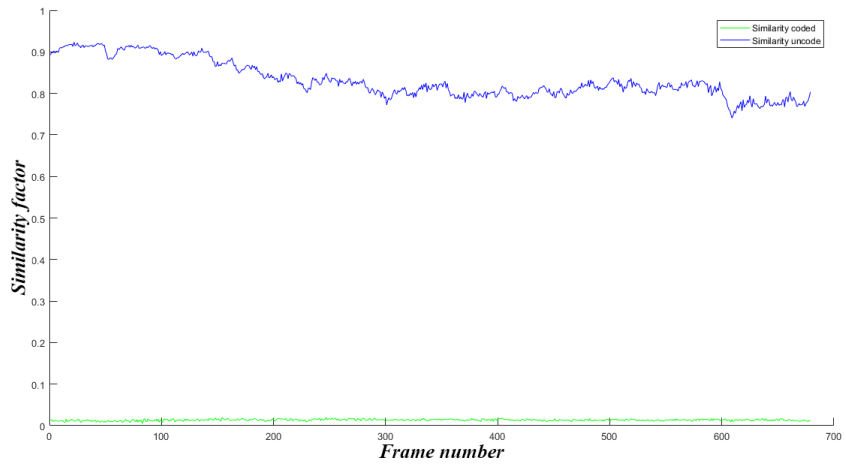
(a) $sc0_{raw}$.



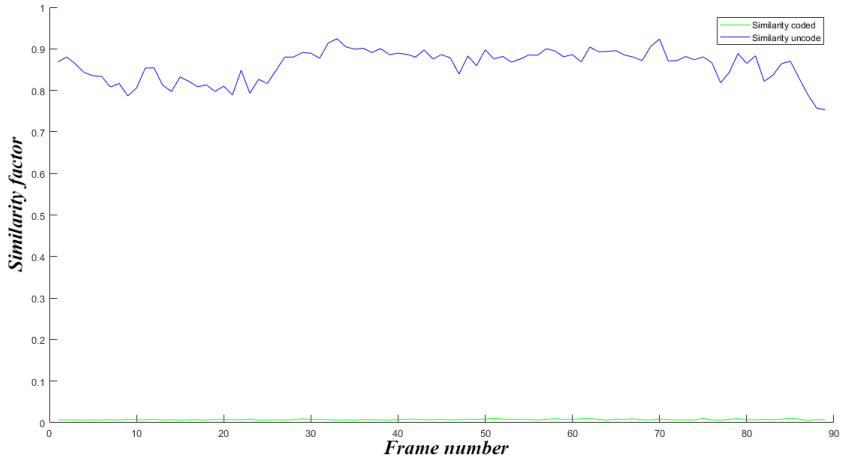
(b) *airs9*.

Figure 3.8: Correlation of two consecutive spatial columns for both coded and uncoded images.

Similarity index metrics, are also available below for both images. Figure 3.9a for $sc0_{raw}$ and figure 3.9b for *airs9*. Similarity index for uncoded images is around 0.9 for both images, but drops to almost zero after de-correlation.



(a) $sc0_{raw}$.



(b) *airs9*.

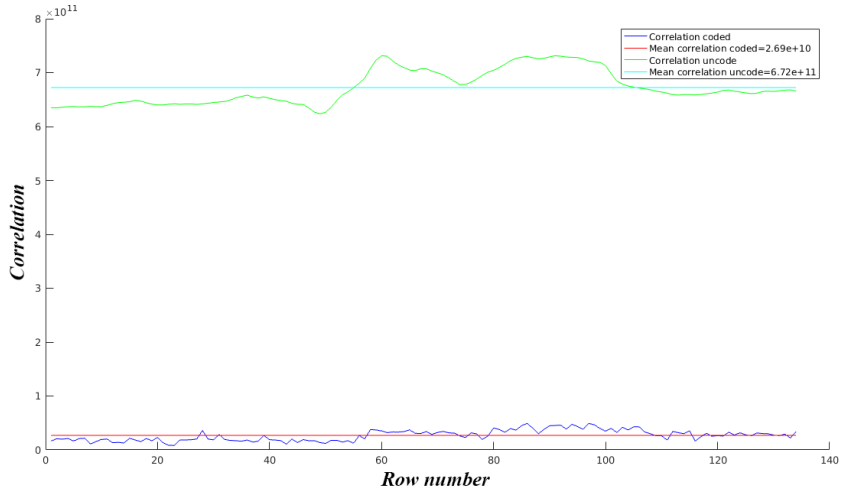
Figure 3.9: Similarity of two consecutive spatial columns for both coded and uncoded images.

3.2.4 Correlation between every spatial row frames

In this section, correlation is being calculated between consecutive spatial rows $x(i,:,:)$ and $x(i+1,:,:)$ where i is index of row in spatial domain for both codded and uncoded images. For $sc0_{raw}$ uncoded image, correlation is $2.4 * 10^{12}$ being dropped to $1.9 * 10^{11}$ by factor of 12.6, demonstrated in figure 3.10b. For *airs9* uncoded image, correlation is $6.7 * 10^{11}$ being dropped to $2.6 * 10^{10}$ by factor of 26, available in figure 3.10b. However, correlations are smaller in this direction, but

de-correlation factors are slightly bigger.

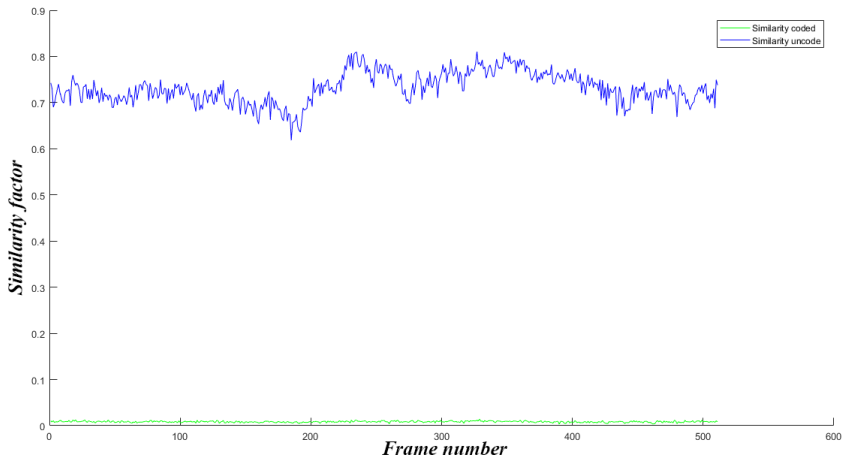
(a) $sc0_{raw}$.



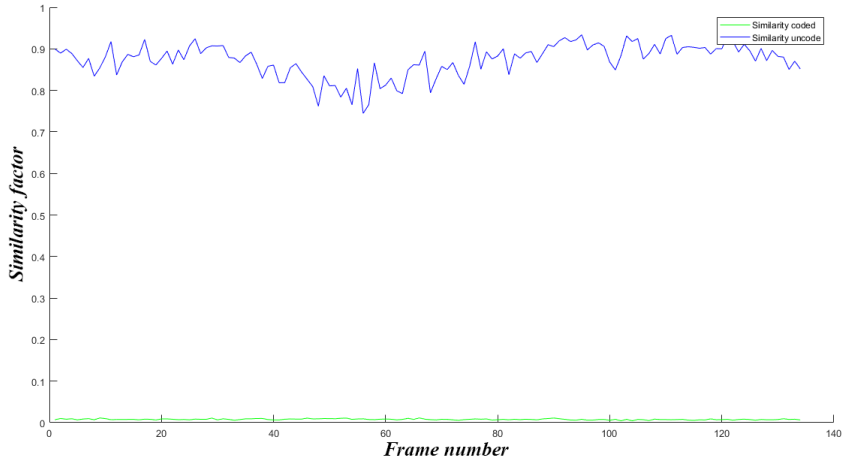
(b) *airs9*.

Figure 3.10: Correlation of two consecutive spatial rows for both coded and uncoded images.

Similarity index metrics, are also available below for both images. Figure 3.11a for $sc0_{raw}$ and figure 3.11b for *airs9*. Similarity index for uncoded images are around 0.7 and 0.9 for $sc0_{raw}$ and *airs9* respectively, but drop to almost zero after de-correlation.



(a) $sc0_{raw}$.



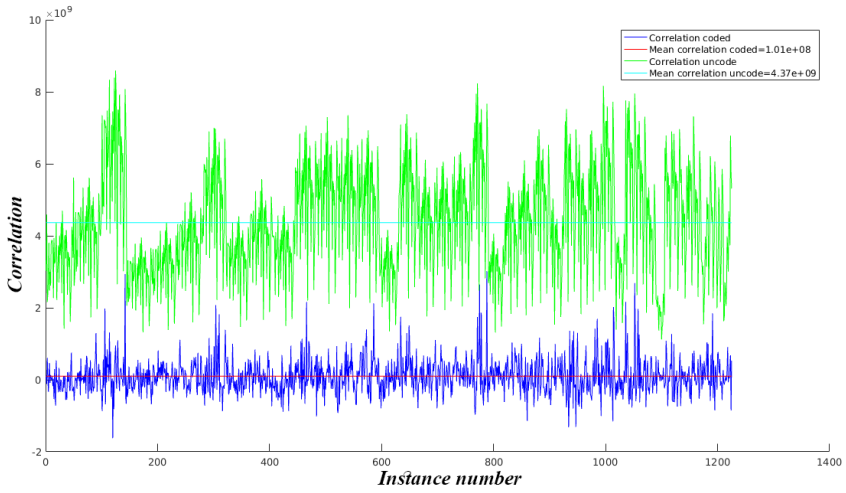
(b) *airs9*.

Figure 3.11: Similarity of two consecutive spatial rows for both coded and uncoded images.

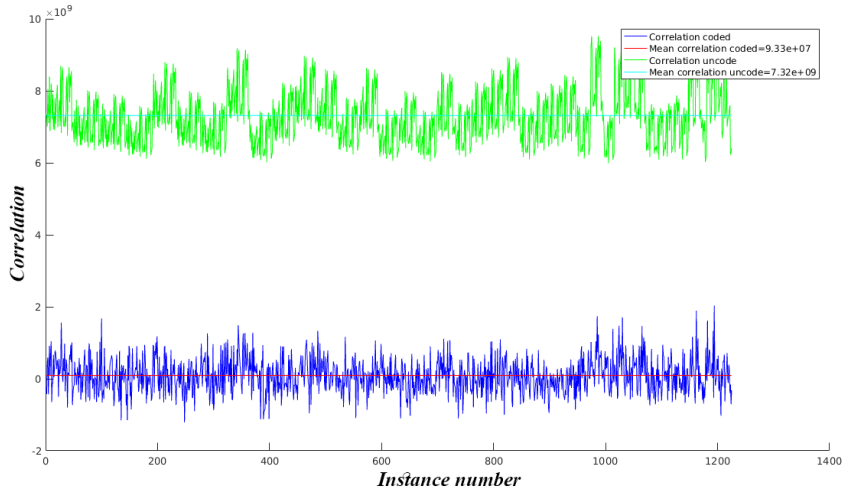
3.2.5 Correlation between random spectral vectors

In this section, correlation between random number of spectral vectors, shown as $x(i, j, :)$ in MATLAB hyper-spectral indexing, are calculated. Indices, i, and j stand for spatial indices in horizontal and vertical coordinates. i, j derived from a uniform distribution over span of horizontal and vertical spatial indices. $i \subset U(1 : size(image,1))$ and $j \subset U(1 : size(image,2))$. This can be the most important scenario in our decision making, because correlations are computed between

random spectral vectors which is the same direction as we applied our GFT. For $sc0_{raw}$ uncoded image, correlation is $4.3 * 10^9$ in average and dropped to $1.01 * 10^8$ by factor of 42.5 which present in figure 3.12a. For *airs9*, correlation is $7.3 * 10^9$ in uncoded image and dropped to $9.3 * 10^7$ after de-correlation by factor of 78.4 visible in figure 3.12b. Given that it is the same direction of applying GFT, still de-correlation factor for spectral frames versus 1st spectrum band is highest.



(a) $sc0_{raw}$.



(b) *airs9*.

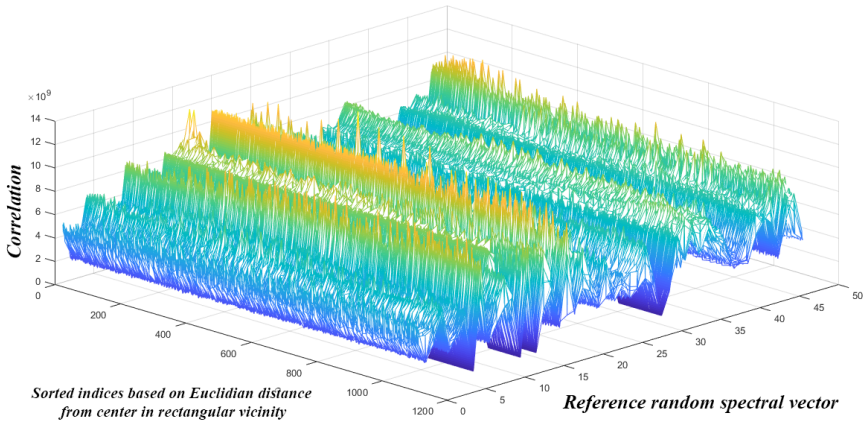
Figure 3.12: Correlation of two random spectral vectors for both coded and uncoded images.

3.2.6 Correlation between random spectral vectors and their vicinities

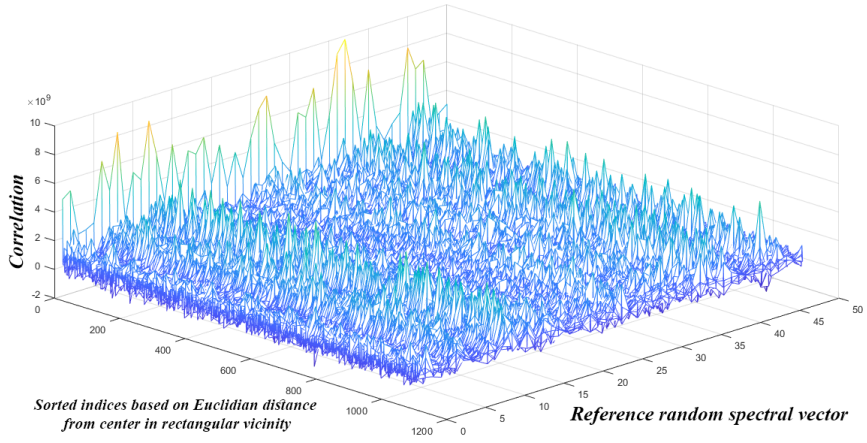
In this section, correlation is computed between random spectral vectors $x(i, j, :)$ and spatial vectors in their spatial vicinity shown as spatial indexing domain $x(i \pm 16, j \pm 16, :)$ corresponds to a rectangular vicinity of 32 by 32. It is a 3D surface consist of 50 items in x coordinates which are 50 random spectral vectors(as reference) same as previous scenario shown as $x(i, j, :)$, where i, j stand for spatial indices in horizontal and vertical coordinates. In y coordinate, 1089 elements exist, per each row, which are spectral vectors in 32 by 32 vicinity of random vectors. These vectors are sorted ascending based on Euclidean distance from center vector. So, it is expected that by increasing the index in y coordinate up to 1089, correlation decrease per each row (represents vicinity spectral vectors). Euclidean distance inside spatial vicinity for two vectors $x(i', j', :)$ as neighbour spectral vector where $i' \subset [i - 16 : i + 16]$ and $j' \subset [j - 16 : j + 16]$ and $y(i, j, :)$ as reference random spectral vector is defined as:

$$Dist(x, y) = \sqrt{(i' - i)^2 + (j' - j)^2}$$

Almost always, first element of each row which is the auto-correlation of center spectral vector, gets maximum value. Correlations for uncoded $sc0_{raw}$ presents in figure 3.13a and coded $sc0_{raw}$ in figure 3.13b. Correlations for uncoded *airs9* presents in figure 3.14a and coded one in figure 3.14b. Obviously, after coding, correlations drop significantly for vectors in vicinity than auto-correlations which remains constant. For example, correlation of vectors in rectangular vicinity of image $sc0_{raw}$ on uncoded ones is $4.56 * 10^9$ in average while drops to $2.09 * 10^8$ in average after codding, so dropped by factor of 21.8. These values respectively, are $4.58 * 10^9$ in average for uncoded image while drops to $1.96 * 10^8$ in average after codding, so dropped by factor of 23.3 for image *airs9*.

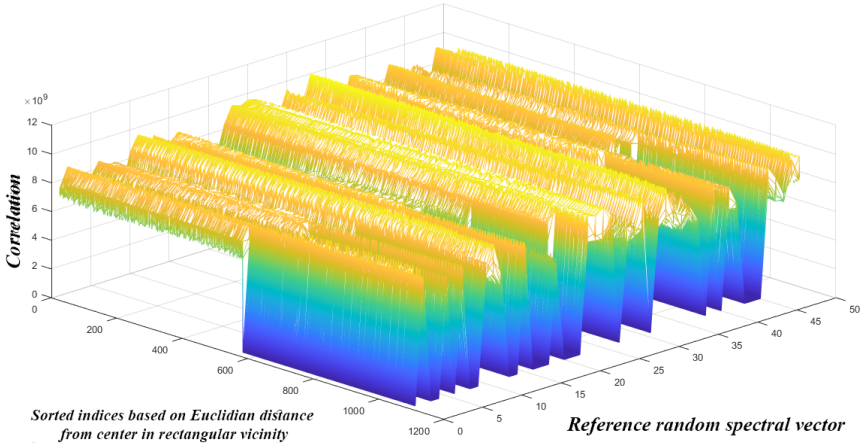


(a) Uncoded images.

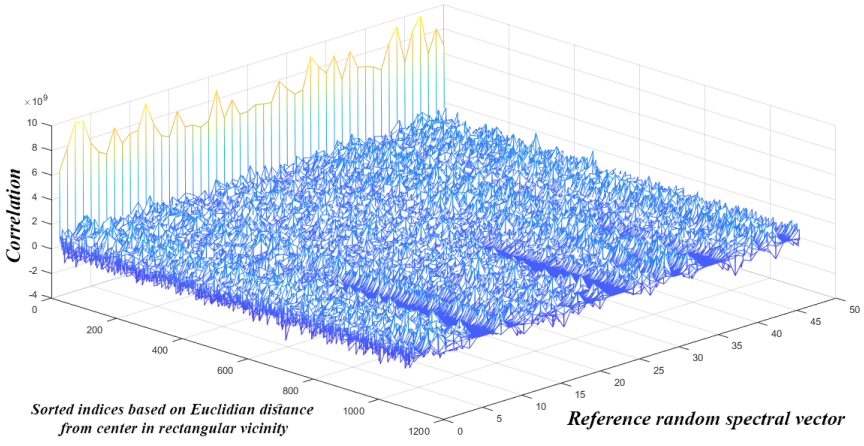


(b) Coded image.

Figure 3.13: Correlation of random spectral vectors and their vicinity of image $sc0_{raw}$.



(a) Uncoded image .



(b) coded image.

Figure 3.14: Correlation of random spectral vectors and their vicinity of image *airs9*.

3.3 Differences of coded spectral vectors

Finally, differences are evaluated between intensity of all coded random spectral vectors shown as $x(i, j, :)$ where i, and j stand for spatial indices in horizontal and vertical coordinates. It is evaluated using equation: $difference\_coded\_spectral\_vector = x(i, j, :) - x(i', j', :)$ After calculating differences, I account for number of bits needed to store these data. This is driven by equation

$$Bits = floor(\log_2^{max\_data})$$

where max_data is:

$$max(difference\_coded\_spectral\_vector(:, :, :))$$

which is maximum value of differences between all coded spectral vector intensities. For both images $sc0_{raw}$ and *airs9*, after 100 runs with different seeds for selecting 50 random spectral vectors, revealed that in average still 16 bits are required to store these differences. It means, that the largest difference after applying GFT is still big in range of uncoded intensities.

3.4 Discussion on changing trends of correlation in different directions

The changing trend of spectral correlation, between spectral frames $x(:, :, i)$ which is the i_{th} spectral frame and first frame $x(:, :, 1)$ will be discussed. It will be assessed whether environmental factors and natural properties of the scene affect these correlations, and in how extend it is possible to elaborate this effect. Then we will come up with the best metric which can represent the expected trend for this correlations and visualize the results. In section 3.6, we go forward toward the compression aspect of using GFT method in compression of hyper-spectral images. We first establish the compression approach and graph structure which is going to be used during the assessment, then the performance of codding is examined by means of preserved percentage energy of encoded and then decoded signal in terms of percentage of GFT coefficients used in encoding. And also, PSNR of compressed image in terms of percentage of GTF coefficients being used in encoding phase.

3.5 Looking for decreasing trend in spectral direction

As we have already discussed in previous sections, the relevance of cross-correlation value, correlation coefficient (provided by xcorr(x,y,0,'coeff')) at lag zero), and

structural similarity index to the comparison made for different cases and sets of signals spectrally and spatially selected, none of them could satisfy the decreasing trend which is expected when we calculate the correlation of a signal with respect to first signal from that batch. In the signal selection set, except those with consecutive pairs of signals, others compared correlation of a signal either spatial or spectral with respect to first signal from that environment. In such a set, it is expected that the trend constantly decrease as signal goes far in the selection context. It means, for instance for a pair of spectral signals consist of $x(:,:,1)$ and $x(:,:,i)$ where i is the spectral band index, correlation should be higher or equal than correlation for a pair consists of $x(:,:1)$ and $x(:,:,j)$ where $i < j$. It must be true since the i_{th} signal is closer to first signal $x(:,:,i)$ than j_{th}.

3.5.1 Possible reasons

Among the three metrics being discussed, correlation coefficient, and structural similarity index are both normalized and cross correlation value is not. For a not normalized value it could not be expected that the decreasing slope shows up. Since, the correlation is evaluated numerically out of multiplication and summation of correspondent signal values. These values are obtained in different environmental status like if consider in visual context and a regular camera, pixels being acquired in different illumination, brightness and even with different camera cells characteristics, could not have meaningfully comparable correlations. It is due to the direct impact of these factors on the signal values which alter correlation value. There are two possible ways to ignore this difference and make the correlation comparable:

- Remove environmental effect from the signal before calculating correlation
- Normalize in a way that was globally meaningful

Remove environmental effect from the signal before calculating correlation

Two possible methods to elaborate difference in environmental, camera cell, and signal intensities could be to divide each signal by its mean, or subtract norm and then calculate the correlation. Well, this is already done in calculation of correlation in MATLAB commands.
There is also another possibility that apart form the effect of environmental factors, if either the object or the hyper-spectral camera had been moving, then the maximum correlation which is trustworthy for comparison between spectral signals is not located in lag zero. Because, taking lag zero in this case, is based on the assumption that those two frames matched with all spatial property kept identical. If it was the case, with the study of 2D cross-correlation for all pair of spectral frames $x(:,:,1)$ and $x(:,:,i)$ where i is the spectral band, there should be considerable

shifts where the maximum value for the 2D cross-correlation lies with reference to spatial center of the image. In figure 3.15 the result of 2D correlation for an image of Sc0raw is demonstrated.

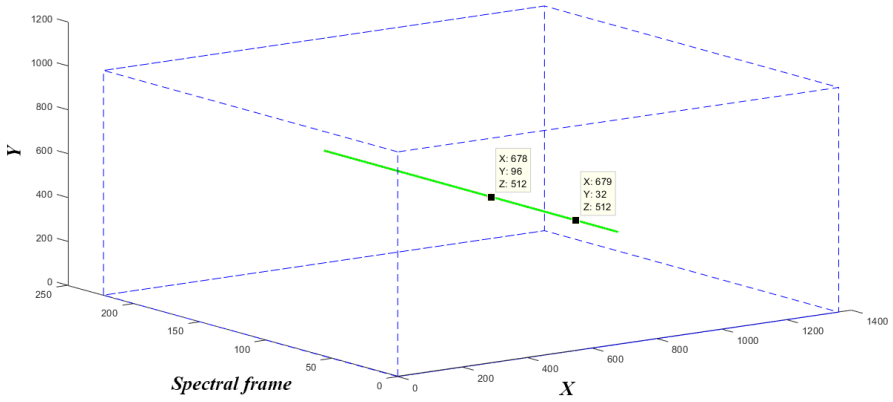


Figure 3.15: 2D spectral Cross-correlation shift from center, image for Sc0raw

The code snippet used to generate the green line showing the position of maximum cross-correlation in function of spatial position is as below:

```
2Dcorrelations=xcorr2(x,y);
[maxValue,MaxIndices] = max(2Dcorrelations(:));
[I_row, I_col] = ind2sub(size(2Dcorrelations),MaxIndices);
```

Listing 3.1: Code snippet used to extract maximum 2D cross-correlation

Where I_{row} and I_{col} are row and column index of maximum correlation in each spectral frame.
As is shown, only in two points, a slight deviation of 1,2 pixel(s) in Y spatial direction were found. For the all other points (lags), the maximum is located at the center of spatial frames. Therefore, this could not be the reason that dropping trend in the cross-correlation of spectral frames is not derived using exact cross-correlation values.

Normalize in a way that was globally meaningful

This time, we evaluate Pearson correlation coefficient [21] and compare it with the normalized correlation coefficient already calculated using xcorr(x,y,0,'coeff')). The results are show in figure 3.16 and figure 3.17. It is finally figured out that the

normalization done by Pearson is the best which preserves characteristics of the signal but eliminates environmental effects the best. The xcorr command gives a normalized value, but the point is that based on equation 3.4, it normalizes each cross-correlation so that the auto-correlations at zero lag equal 1.

$$R_{xy,coeff}(m) = \frac{R_{xy}(m)}{\sqrt{R_{xx}(0)R_{yy}(0)}} \tag{3.4}$$

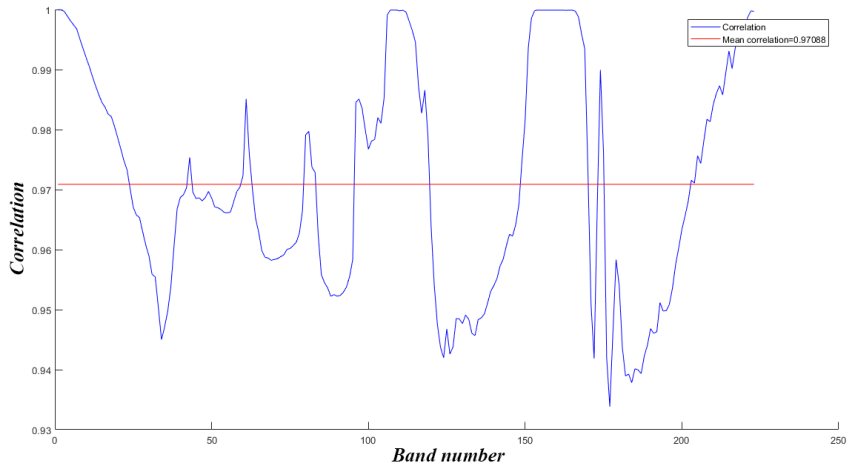


Figure 3.16: Correlation coefficient of spectral bands with reference to 1st band using xcorr command

But this implies that regardless of the real actual value of correlation, the auto-correlation of signal are normalized to have value one in zero lag and this is exactly where the value and worthiness for the comparison is lost for this metric. On the other hand, using Pearson correlation coefficient, with equation 3.5 it standardizes signal values by changing their mean and variance to fixed values.

$$\rho_{xy} = \frac{COV(x,y)}{\sigma_x \times \sigma_y} \tag{3.5}$$

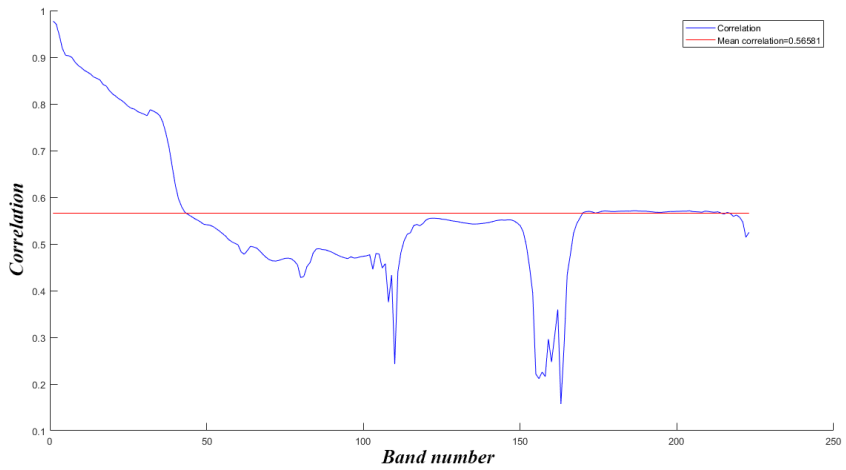


Figure 3.17: Pearson Correlation Coefficient of spectral bands with reference to 1st band

Using this standardization, one could claim that for any signal, signal values would be first normalized in the identical extent without loosing their characteristics, then a meaningful normalized correlation value is calculated which does not carry any environmental information and solely stands for signal characteristic.

3.5.2 Conclusion on decreasing trend of spectral correlations in reference to 1st band

With the results form Pearson Correlation Coefficient, it could be claimed that this metric is good-to-know, and qualifies for the aim of comparison where the trend of correlation changes is needed to be studied. Otherwise, the cross correlation value without any kind of normalization or standardization, must be used when a judgement in terms of compression cost and worthiness should be done.

3.6 Edge weight metric performance

After all, in this section the performance of GFT compression method in terms of number of GFT coefficients being taken into account to reconstruct (decode) the encoded image will be studied. Moreover, PSNR of an image passing through encoding and decoding using different number of GFT coefficients will be studied. Since the GFT mimics Fourier Transform in the Graph Signal representation domain, the very end coefficients stand for details and in expense of loosing some details, a portion of them could be cut out from the end of coefficient vectors.
To construct the graph, each spectral vector denoted as $x(c, r, :)$ is taken where

c is the column in spatial domain, r is the row in spatial domain, and the third index is all spectrum span, this spectral vectors are taken as signals to de-correlate since we are interested in de-correlating in the direction which carries most if the correlation. Nodes of our line graph are pixels in these spectral vectors and edge values are calculated using the Gaussian function as follow:

$$\omega_{ij} = e^{-\frac{d_{ij}^2}{\sigma^2}} \tag{3.6}$$

where $d_{ij} = |f_i - f_j|$ is the Euclidean Distance between pixels i and j, and σ is defined as in [6]. Since the graph should be transferred as side information to the decoder which increases the information volume to almost as twice times bigger than original image, a quantization or compromise should be seen to lighten the graph size. In this evaluation, we used a single identical graph structure for all signals to be encoded. This unique graph, is composed of means of every spectral frame $x(:,:,i)$ and then those values composed a line graph which is used to calculate the adjacency matrix W and the Gaussian Laplacian Matrix L. From the eigenvectors of this matrix L, transform matrix U is taken to calculate the GFT using $\hat{f} = Uf$ equation. The code snippet below is used to generate the average vector stands for whole image to construct the graph:

```
for i=1:spectralIndex
   mean_Signal(i,1)= mean(mean(image(:,:,i)));
end
```

Listing 3.2: Code snippet used to construct the mean signal on graph

3.6.1 Percentage of energy, in function of percentage of retained coefficients

In the figure 3.18, an image from Sc0raw is codded and decoded using different percentage of GFT coefficients and the resulting preserved energy of decoded image is calculated and divided by the energy of original image and is reported in percentage. The weights are not quantized in this evaluation. The graph Fourier Transform is learned from the mentioned mean spectral vector.

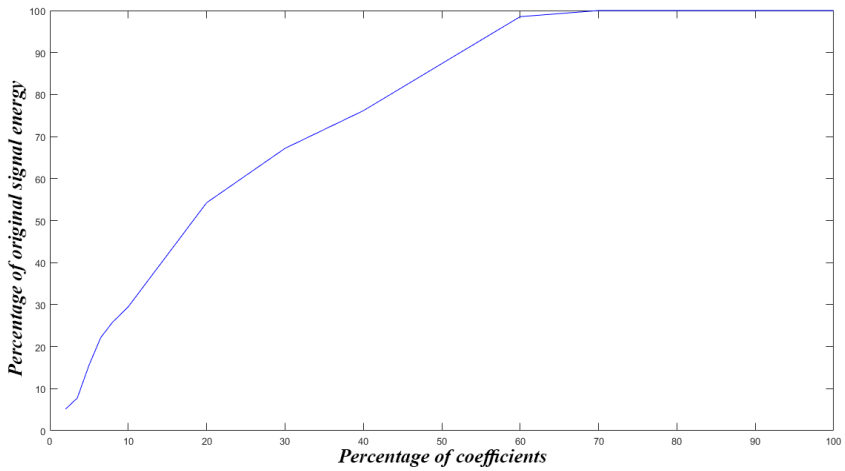


Figure 3.18: Percentage of energy, in function of percentage of retained unquantized coefficients using mean spectral vector to learn GFT matrix

The code snippet which is used to calculate the signal energy is given in below, and it calculates the energy by multiplying Fourier Transform of the signal by its conjugate Fourier Transform and sums up all signal energies in the image.

```
function energyImg =signalEnergy(img)
F = fft(img);
pow = F.*conj(F);
energyImg=sum(sum(sum(pow)));
end
```

Listing 3.3: Code snippet used to calculate signal energy

The performance in function of percentage of the GFT coefficients retained is very acceptable, since they carried close to 100 percent of the original signal energy by only 60 percent of the coefficients. And it decrease gradually to almost 5 percent taking less percentage of coefficients into account.

3.6.2 PSNR in function of percentage of retained coefficients

The more strict metric to evaluate the de-correlation performance is peak signal to noise ratio (PSNR). It is evaluated in the function of different percentage of GFT coefficients taken into account in the encoding and decoding phases. From the described graph structure, the PSNR is evaluated in range of 2 to 98 percent of GFT coefficients cutting from the tail of coefficient vectors. The weights are not

quantized in this evaluation.

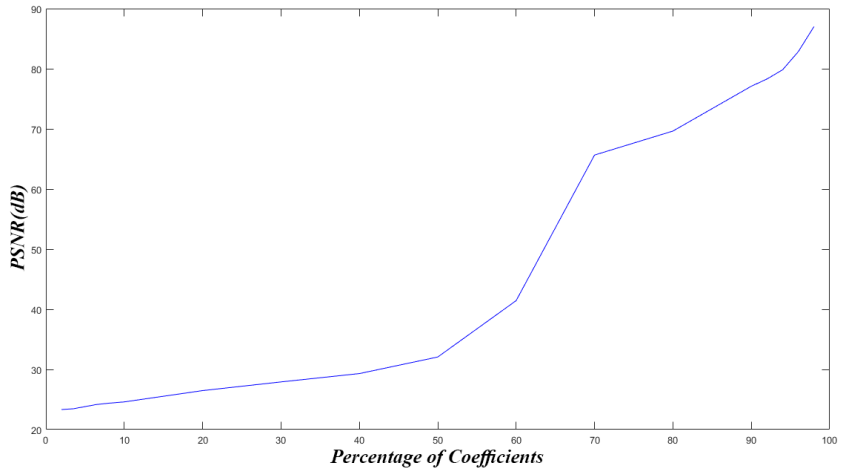


Figure 3.19: PSNR in function of percentage of retained unquantized coefficients using mean spectral vector to learn GFT matrix

The *psnr* internal function of MATLAB is used to calculate PSNR values. It is very satisfying since to achieve around 45dB PSNR, only 65 percent of coefficients are enough and it could be a good point in the next decisions to perform final steps of compression and obtain the performance in function of bitrate. In a very lossy manner, by retaining only 2 percent of coefficients, almost 25dB is achieved.

3.7 Dedicated GL-Vectors

So far the coding performance was demonstrated in terms of PSNR and percentage of retained energy of decoded image in reference to original image both versus difference percentages of coefficients. In the previous evaluations due to decrease of side-information and based on intermediate correlation examinations, an identical GL-Vector was used for all spectral vectors in the encoding stage. This GL-vector was created based on mean of each spectrum plane $mean(mean(x(:,:,i)))$ where i is the plane index in image context and it is element index in shared GL-vector. It is worthy to clarify, due to smaller size of the vector that this shared GL-vector is calculated from, the mean spectral vector itself is transferred as side-information, and in the decoder side, GL-vector is calculated based on this sample mean vector. Therefore the sample mean spectral line graph has a known shape and structure in the decoder side and it has the size of 1 by d where d is the length of each vector being encoded, in this case number of spectral vectors in the hyper-spectrum image.
In this section though, an individual dedicated GL-vector is taken for encoding.

This vector is the same spectral vector under encoding, represented in graph structure and Gaussian-Laplacian is calculated based on these dedicated vectors.
In figure 3.20, the PSNR versus percentage of coefficients is demonstrated for the case of taking dedicated GL-vectors.

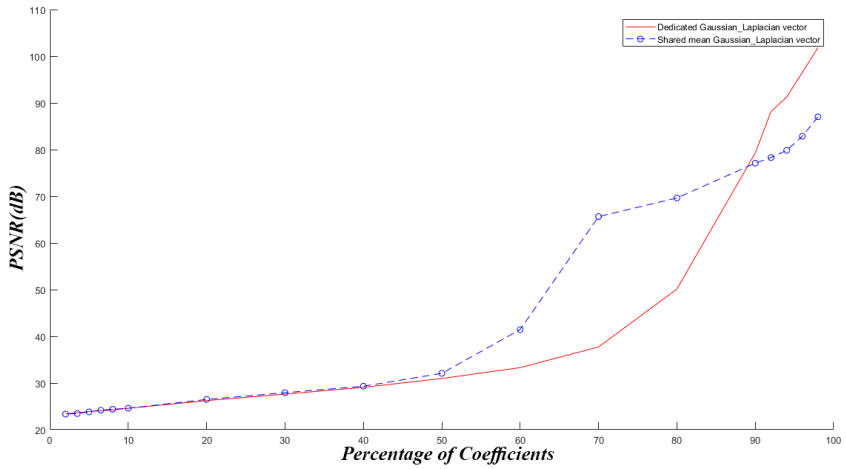


Figure 3.20: PSNR vs percentages of coefficients using dedicated GL-vectors for scene0 Yellowstone uncalibrated

In the figure 3.21, the percentage retained energy versus percentage of coefficients is demonstrated for the case of taking dedicated GL-vectors.

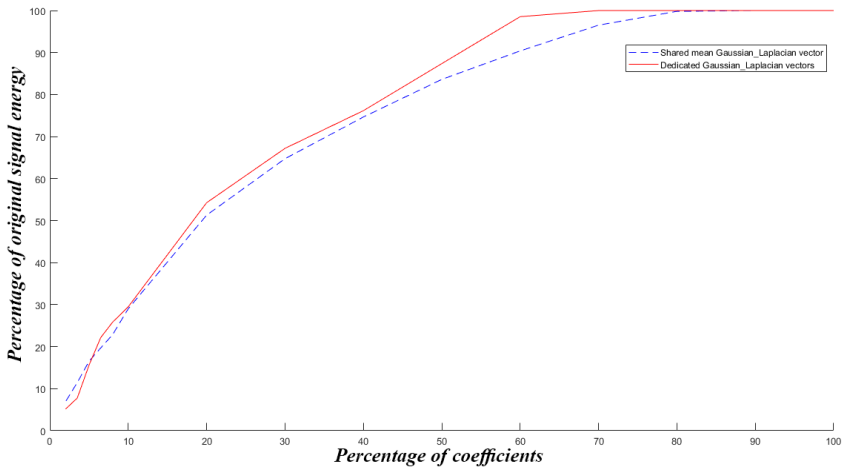


Figure 3.21: Percentage retained energy vs percentages of coefficients using dedicated GL-vectors for scene0 Yellowstone uncalibrated

Based on PSNR metric, dedicated GL-vectors as it is expected improve the PSNR when 100 percent of coefficients taken into account. However, between 50 to 80 percent of coefficients taken, shared mean GL-vectors outperforms the dedicated ones and in other parts, there is not a noticeable improvements between shared and dedicated GL-vectors. According to percentage of retained energy after encoding and decoding, one can hardly find a considerable advantage of taking all dedicated GL-vectors over the shared vectors used for compression. Overall, considering that each GL-vector is a square matrix of size $d * d$ where d is the length of each vector being encoded, in this case number of spectral vectors in the hyper-spectrum image. So, for example in scene0 of Yellowstone uncalibrated, there exist 224 spectrum planes which leads to 224 by 224 square metrics per each GL-vector. It is even bigger than the size of image itself and makes no sense to continue with dedicated GL-vectors either based on gain it brings or side-information size.

3.8 Quantization of coded coefficients

Up to this section, all the encoded images using GFT in this thesis, were stored in double precision format due to not lose of data. But, since the original images were 16-bit unsigned images, and based on the scale of encoded coefficients after applying GFT, coefficients will be rounded to closest integer number and stored as int16 format. The effect of quantization on dedicated GL-vectors will be determined first, and then evaluation of PSNR and retained energy percentage in function of percentage of coefficients taken into account will be done. Note that, due to Fourier

Transform like inherit of encoding, coefficients are sorted decreasingly from the beginning of each encoded vector. So the percentage of coefficients are taken from the beginning of each vector and the rest is set to zero. Therefore, we make sure than coefficient with higher energy and bigger values are taken when a cut is made on coefficients. The code snippet below is used for this purpose.

```
function cutedImage =Cut_Image_Tail_Percentage(img,percentageOfCoeff)

w=size(img,2);
h=size(img,1);
d=size(img,3);
CoeffFraction=(100/percentageOfCoeff);

indexToCut =floor(d/CoeffFraction);
cutedImage = img;
cutedImage(:,:,indexToCut:end)=0;
end
```

Listing 3.4: Code snippet used to take section percentage of coefficients

In the figure 3.22, effect of quantization of coded image is visualized in term of PSNR compared to non-quantized coded image taking dedicated GL-vectors. Also, in the figure 3.23, this effect is demonstrated in term of percentage energy retained for dedicated GL-vectors both in function of percentage of taken coded coefficients.

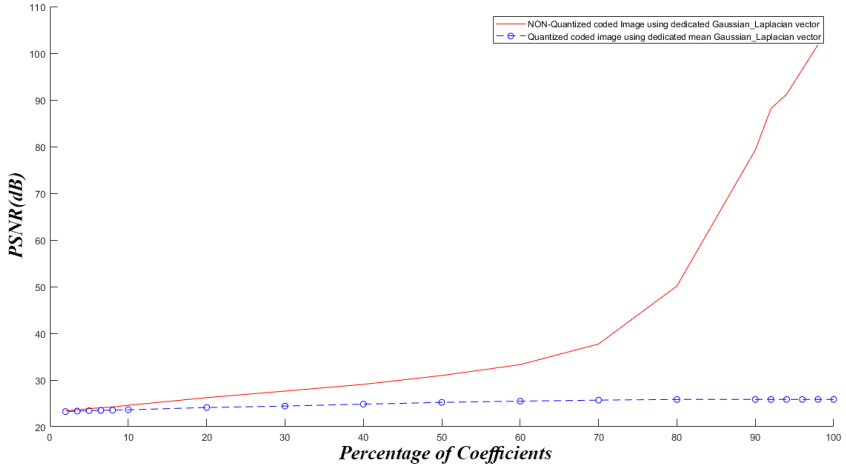


Figure 3.22: PSNR of quantized coded image using dedicated GL-vectors for scene0 Yellowstone uncalibrated

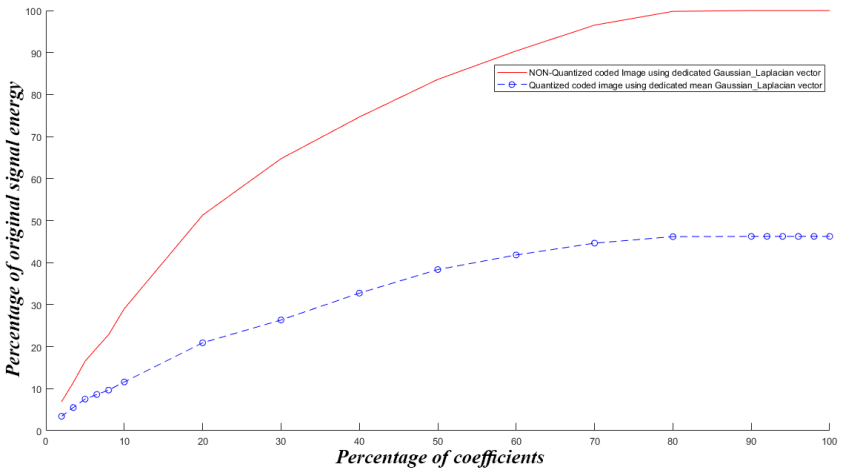


Figure 3.23: Percentage retained energy of quantized coded image using dedicated GL-vectors for scene0 Yellowstone uncalibrated

Based on figure 3.23 the amount of energy is almost halved in the right side of graph where still good percentage of coefficients exist. Although, it does not resemble a reasonable PSNR in the decoded image based on figure 3.22 as the PSNR is constantly dropped to 23 dB.

The same evaluation is however done in the case that shared mean GL-vector is taken for encoding all spectral vectors. Surprisingly, quantization does not have an impact on the retained energy of decoded image as is visualized in figure 3.25. And PSNR only drops 20 to 10 dB in the right side of graph based on figure 3.24. From 60 percent of coefficients down, there is no impact due to quantization on PSNR of decoded image compared to full precision coded image.

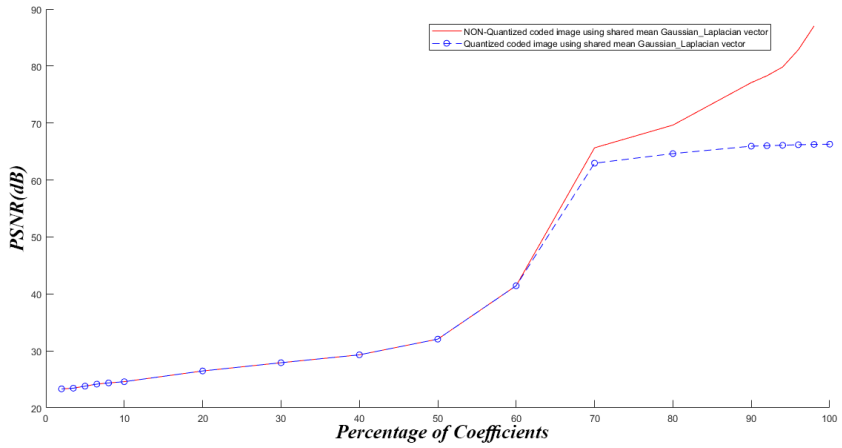


Figure 3.24: PSNR of quantized coded image using shared GL-vectors for scene0 Yellowstone uncalibrated

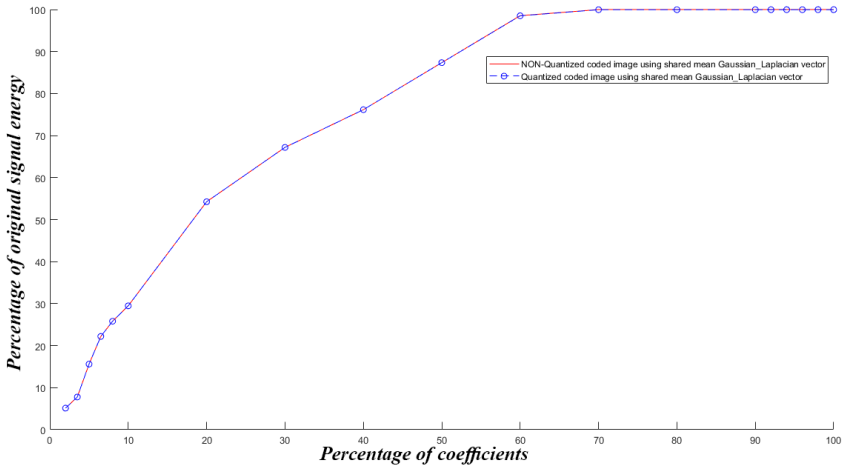


Figure 3.25: Percentage retained energy of quantized coded image using shared GL-vectors for scene0 Yellowstone uncalibrated

According to these results, the quantized coded images based on shared mean GL-vector is going to be assessed for the final section of rate distortion performance. Still one question remains regarding the side information. This side information which is to be transferred to decoder in extra, is a d big vector of doubles (64 bit per value) where $d = size(HyperSpectralImage,3)$ as number of spectrum planes in the image. In continue, the impact of quantizing this GL-vector (actually the

graph vector which this Gaussian-Laplacian is calculated from) on the performance of coding is evaluated.

3.8.1 Quantization of shared GL-vector

In this part, the base vector in which shared mean GL-vector is calculated from, is going to be quantized using uint16 (fractional parts were missed). The effect of this quantification can be seen in the figure 3.26 in terms of PSNR and in the figure 3.27 in terms of energy retained form decoded image.

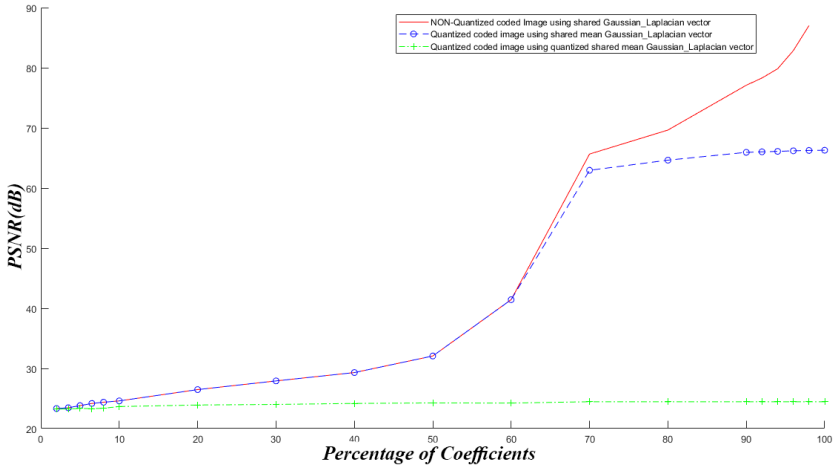


Figure 3.26: PSNR of quantized coded image and GL-vector using shared GL-vectors for scene0 Yellowstone uncalibrated

For this cases the same as the case where dedicated quantized GL-vectors where used, the energy is almost halved and PSNR respectively decreased to 23dB. Therefore, the quantization of shared GL-vector is too greedy and it is better to be transferred in double precision.

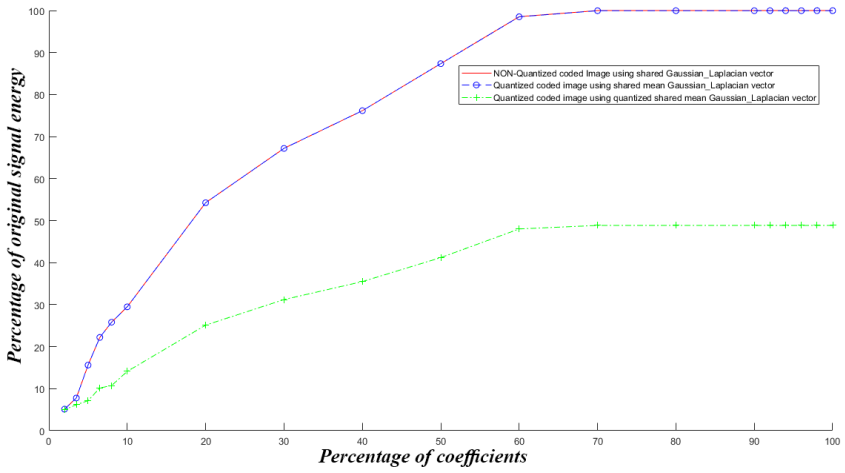


Figure 3.27: Percentage retained energy of quantized coded image and GL-vector using shared GL-vectors for scene0 Yellowstone uncalibrated

Note that a sophisticated quantization matrix/vector is not applied on the codded image data since the simple rounding to signed integer sounds enough for early evaluation of the codding performance.

3.9 Cauchy weight function

In [11] it is shown that to calculate graph weighs, Cauchy weight function 2.1 outperform the Gaussian function 2.2 as it better preserves energy of original signal. To test this statement, considering transform coding is applied on sample image AVIRIS scene0 of Yellowstone National Park USA using GFT on spectral vectors. Then, PSNR and percentage of preserved energy are computed in function of percentage of taken coefficients and compared to previous results using Gaussian function. PSNR and amount of preserved energy for a complete coding and decoding vector GFT are demonstrated consecutively in figure 3.28 and 3.29.

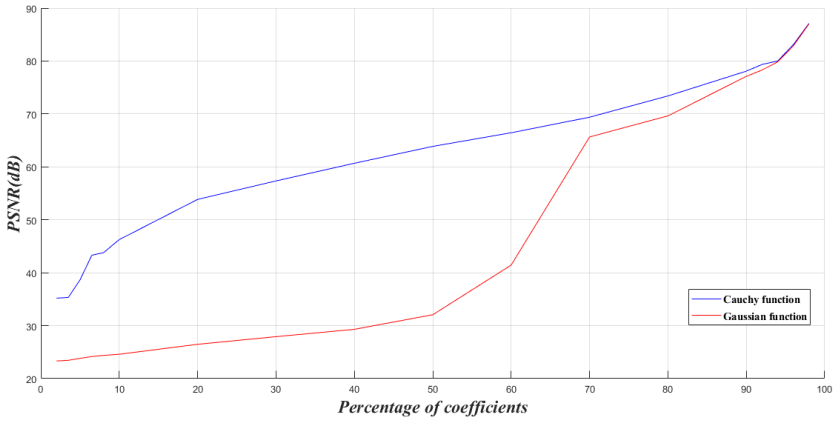


Figure 3.28: PSNR in function of percentage of retained coefficients using Cauchy and Gaussian weighting function

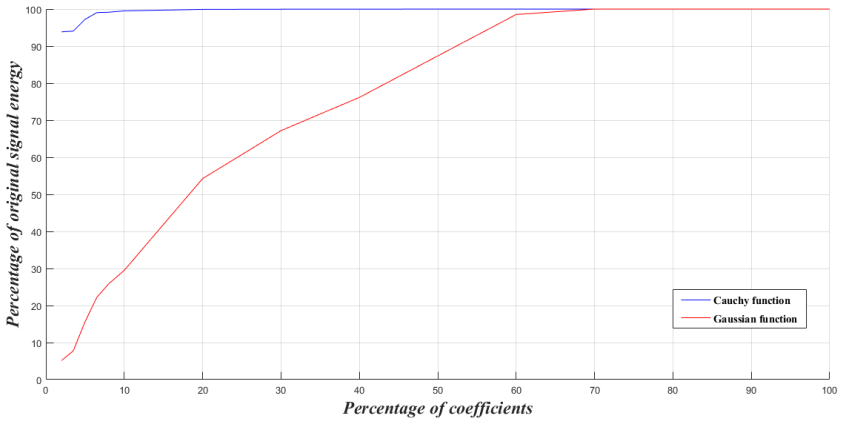


Figure 3.29: Percentage of energy in function of percentage of retained coefficients using Cauchy and Gaussian weighting function

It is shown that Cauchy function brings some improvement in both metrics. But, it is not going to be used in the rest of this thesis as main weighting function, since weight values of Cauchy equation are very small and they require high resolution of presentation, or sophisticated quantization to be used and stored.

3.10 Proposed 3D graph structure

In this section, structure of the 3D graph is introduces to achieve a sparse representation of the image signal applying GFT on the graph domain. Let us assume that x is given hyperspectral image of K spectral bands and spatial dimension of M.

$x_{i,k}$ indicates the intensity of i-th pixel in k-th spectral band where $i \in \{1, ..., M\}$ and $k \in \{1, ..., K\}$. A graph then is represented by its vertices and links between these vertices. A link can be spacial S or spectral F in which $S \cup F$ includes all graph links and they do not have link in common.

For spatial and spectral connection of links, consider the figure 3.30. In this structure, spectral bands are grouped every 8 bands and called group of bands (GOB) to gain better random access and apply block-wise standard 3D-transform.

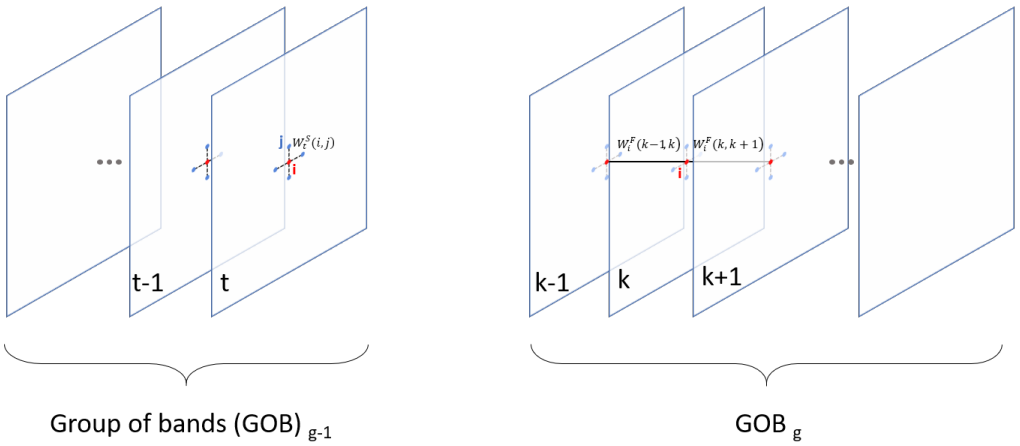


Figure 3.30: 3D graph structure

Inside each band spatially 4 connections are considered per pixel. To the left, right, up and down sides. Considering spatial band t, weight of the link between pixel i and j is calculated based on Gaussian function. Since spatial correlation is almost the same amongst spectral direction, for all the band in a GOB, same spatial edges are used to form the transform basis. On the other hand, on spectral dimension we perform similarly to the case of spectral vector GFT and per each pixel in first band, a spectral vector graph based on $x(r, c, :)$ is shaped where r,c are spacial indices. This vector is called spectral signature. Spectral weight are calculated based on Gaussian function on this vector structure and transform basis is also learned from a mean spectral vector similarly to previous cases of spectral vector GFT.
It is worthy to mention, that spatial and spectral signatures which transform basis is going to be learned from, must be sent to decoder as overhead side-information.

3.11 Encoding and decoding schemes

Based on the explained graph structure, 2 different coding schemes are proposed, one with lower complexity and other one with higher complexity. Complexity is

determined in sense of demand to re-calculate and update GFT transform basis as it requires to calculate eigenvector and many multiplication operations. In both approaches, spatial and spectral GFT basis are learned separately, then they are coupled as a 3D transform graph basis and used to transform and de-correlate image signal on the graph.

3.11.1 Low complexity mode

In low complexity mode, whole first band is used as reference spatial signature to learn spatial edge weights of transform. Therefore, this first band is compressed using JPEG2000 compression with quality factor 1, since signatures are very sensible regarding reconstruction errors and this error could lead to big performance loss in GFT. Having first band compressed using JPEG2000, it is transferred to decoder as side-information and its size is added to final bit per pixel per band(bpppb). For this particular case of scene0 AVIRIS Yellow Stone, it adds a bias of 0.0221 to final bpppb due to transition of first band. Similarly to the case of spectral vector GFT which is used for all basic evaluation in the thesis, an average spectral vector is calculated and used to learn spectral transform basis. Then, having the 3D transform basis, picture is divided into 8x8x8 blocks and transform is applied per each block of image. Figure shows how 3D transform low complexity is designed.

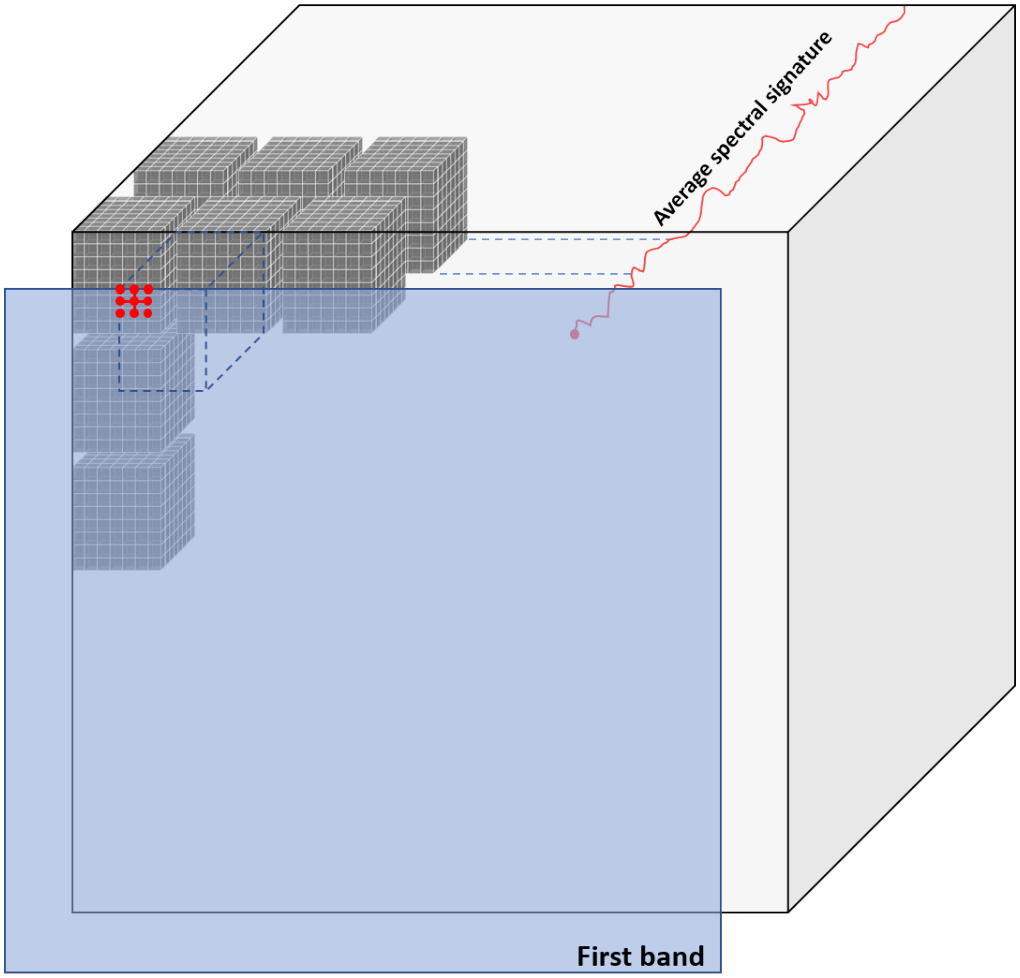


Figure 3.31: Low complexity 3D graph coding scheme

In this scheme, first band is uniquely used to learn transform basis of GFT for all spatial edge weights. It means, that in the time of coding or decoding each block, spatial area in the first band corresponds to the projection of the undergoing block transform, is taken as spatial signature to learn the spatial transform basis. Therefore, spatial edge weight for edge between pixels i and j both in band t is calculated based on following equation.

$$w_t^S(i,j) = exp(-\frac{(\hat{x_{1,i}} - \hat{x_{1,j}})^2}{\sigma^2}) \tag{3.7}$$

Where $\hat{x_{1,i}}$ and $\hat{x_{1,j}}$ are corresponding pixel projections on the first band.

In the same way, corresponding projection part of the block on average spectral signature β is taken as reference to learn spectral transform basis. Therefore, spectral edge weight for edge between pixels i in consecutive bands with indices k and $k+1$ is given using following equation.

$$w_i^F(k, k+1) = exp(-\frac{(\beta(k+1) - \beta(k))^2}{\sigma^2}) \tag{3.8}$$

where $\beta(k+1)$ and $\beta(k)$ are corresponding pixels to the projection of block to average spectral signature.

3.11.2 High complexity mode

In high complexity, there are two addition differences to make transform basis more adaptive to the input signal with almost no cost regarding the side-information. First, when GOBs progress in spectral direction, next GOBs which are located in preceding spectral indices, they have the chance to use their previous spatial GOB decoded data as reference to learn spatial transform weights. In this way, reference if learn is closer to the band and probably is more correlated to the undergoing band transform. Based on figure 3.32, it is demonstrated that only first layer of GOBs use JPEG2000 decoded first band as reference to learn edge weights, but preceding ones, use their corresponding spatial area of last frame in previous GOB.

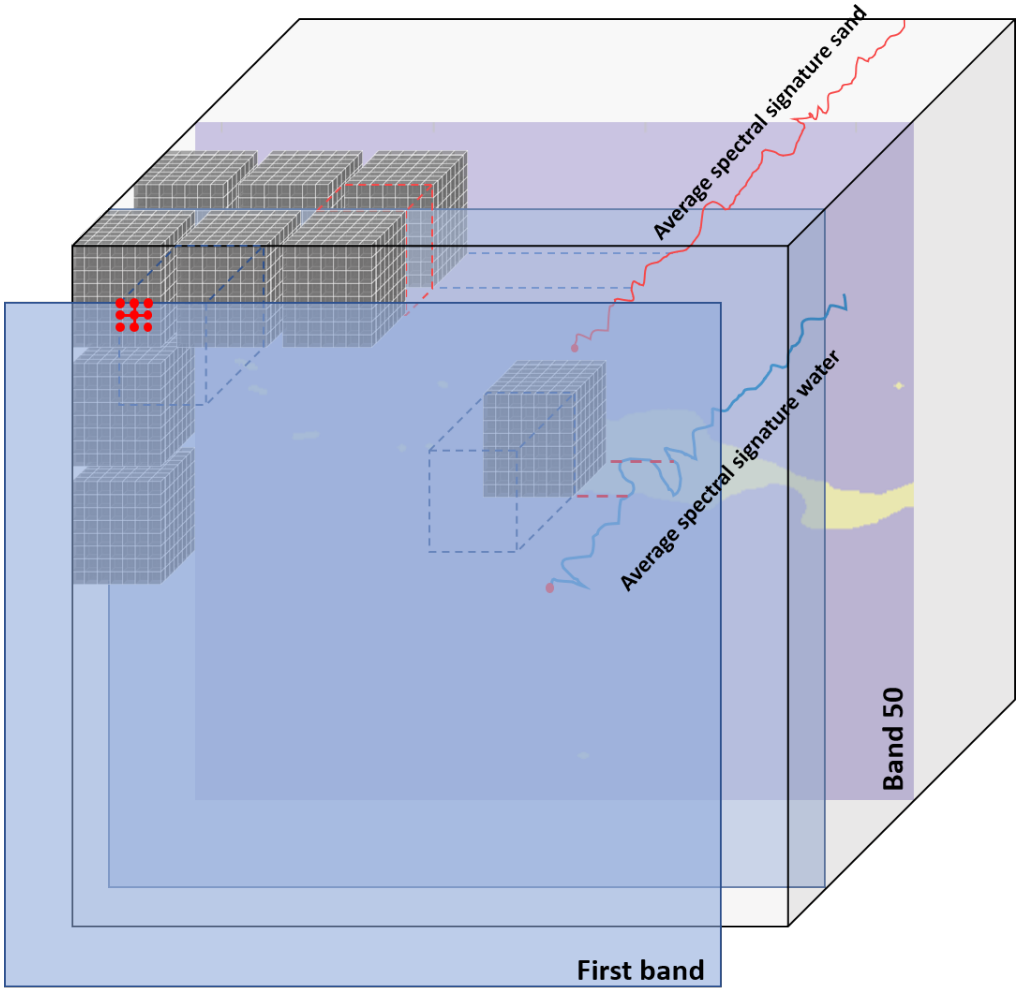


Figure 3.32: High complexity 3D graph coding scheme

Therefore, spatial edge weight for edge between pixels i and j both in band k inside GOB_g if g is not the first spectral group, is re-calculated based on following equation.

$$w_k^S(i,j) = exp(-\frac{(\hat{x_{t,i}} - \hat{x_{t,j}})^2}{\sigma^2}) \tag{3.9}$$

Where $\hat{x_{t,i}}$ and $\hat{x_{t,j}}$ are corresponding pixel projections on band t which is the last band in previous GOB_{g-1} and all spectral weights are the same inside each GOB.

Second improvement is related to spectral signature. Based on figure 3.33, examining two random samples of sand and water pixel spectral signature, it is found

that they have very low correlation which is also cited in [23]. There fore a separation is made between pixels of water and sand where spectral edge weights are calculated. Examining band 50, using the thresholding on value 2200 and Canny edge detector on Binary image, water areas are detected. Additional average spectral signature vector is calculated for only water areas, based on pixels in water areas of all bands only, called β_p where $p = 1$ for water area and $p = 0$ for other area which we call sand area from now on.

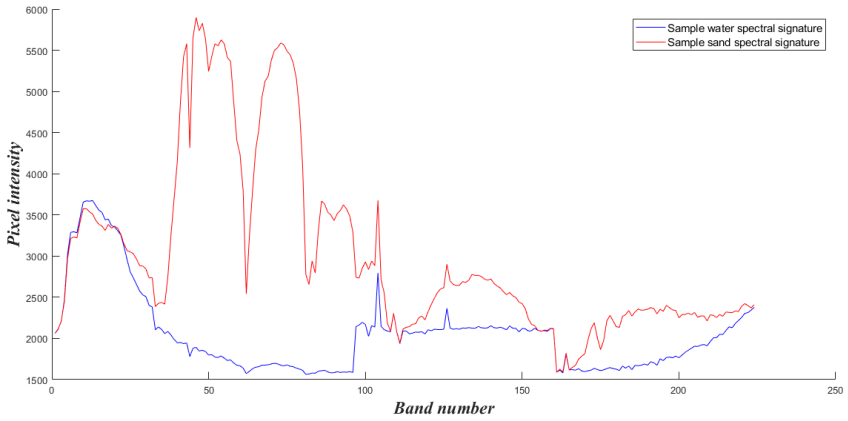


Figure 3.33: Difference spectral signature sand and water

Consequently, spectral edge weights are calculated based on projection to the correct spectral average signature vector based on the location of pixel inside or outside water contours using following formula.

$$w_i^F(k, k+1) = exp(-\frac{(\beta_p(k+1) - \beta_p(k))^2}{\sigma^2}) \tag{3.10}$$

where $\beta_p(k+1)$ and $\beta_p(k)$ are corresponding pixels to the projection of block to average spectral signature whether $P == 1$ or $P == 0$ then is consequently uses water and sand spectral signature vector to learn spectral weights. In this case of Scene0 AVIRIS Yellow Stone Park ,this adds $2.1e-5$ to the final bpppb as cost of transmitting additional spectral signature.

The given graph structure leads to an acceptable weight performance for Graph Fourier Transform on the Hyper-spectral images. It is due to the high correlation and similarity of spectral and spatial frames. To bring the actual compression performance of this method, it is required to quantize the coefficient values and perform all possible entropy encoding and measure bitrate and evaluate the PSNR in function of bitrate. To avoid difficulty of further compression efforts, a mapping to bit plane could be used to estimate the bitrate based on the entropy of this bit planes.

Chapter 4

Results

In section 4.1, the approach to approximate the bitrate for coded images using spectral vector GFT coder is introduced and then bitrate for different selections of coefficient percentages will be examined. This will relate bitrate to the lately calculated PSNR and Percent Energy retained in reference to original image. The performance of codding is also compared to recent coding schemes offered in literature related to the same dataset.

4.1 Rate-distortion performance spectral vector GFT

Last but not least, according to all evaluations, a signed integer quantized coded image using shared mean GL-vector is going to be examined in terms of bitrate (bits per pixel per band [1]) versus PSNR to come up with the distortion rate of coding scheme using GFT with mentioned arrangement.
To this end, the entropy is considered only and an approximation is used based on mapping the coded image data to bitplane. The corresponding bitplane, then evaluated for entropy of whole planes and this entropy is considered as approximation of an optimal entropy coder. This is an approach used to perform scalable compression [25] and rate-controlled entropy coding of 3D video compression [4]. The snippet code below is used to shape 16bit bitplanes and add one plane layer as 17th layer for sign since coded images are signed integers.

```
if(signed)
    signs=img>=0;
    img=abs(img);
```

[1] bpppb

```
    coded_data(:,:,:,1)=logical(signs);
    for k=1:d
        coded_data(:,:,k,2:q_bits)=reshape(logical(de2bi(img(:,:,k),
            num_planes-1)),[h,w,num_planes-1]);
        totalbits=totalbits+16*h*w;
    end
else
    for k=1:d
        coded_data(:,:,k,:)=reshape(logical(de2bi(img(:,:,k),num_planes))
            ,[h,w,num_planes]);
        totalbits=totalbits+16*h*w;
    end
end
```

Listing 4.1: Code snippet used to create bitplane

As first insight, bitrate is shown in figure 4.1 as a function of percentage of coefficient retained from coded image.

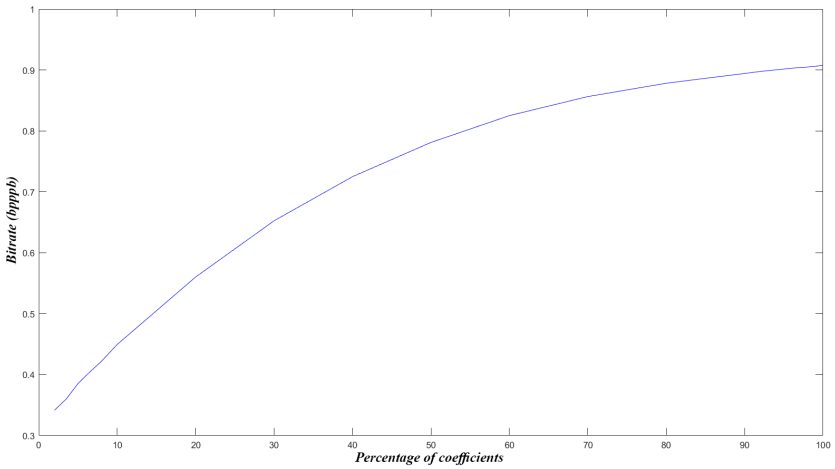


Figure 4.1: Bitrate VS Percentage coefficient using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated

For a better comparison, this result is related to the former correspondence of same arrangement from percentage of coefficients retained to PSNR and Percentage energy retained for shared GL-vector and quantized coded image. The Rate distortion for scene 0 of Yellowstone uncalibrated is shown in figure 4.2 and bitrate versus percentage of energy retained is shown in figure 4.3.

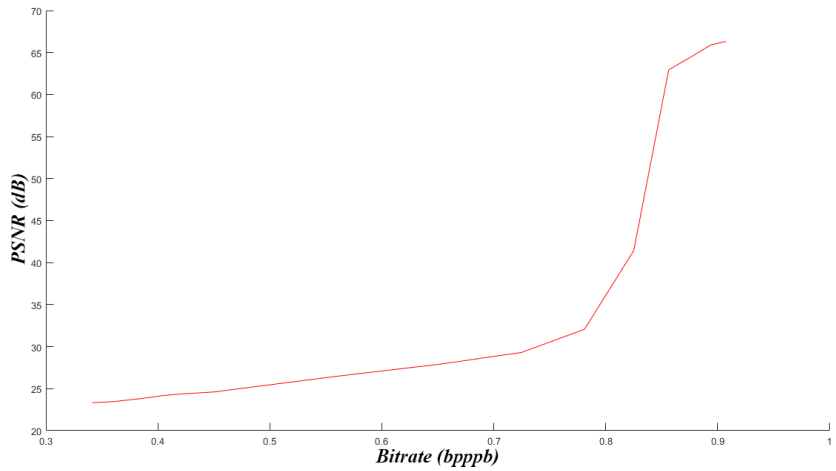


Figure 4.2: Bitrate VS PSNR using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated

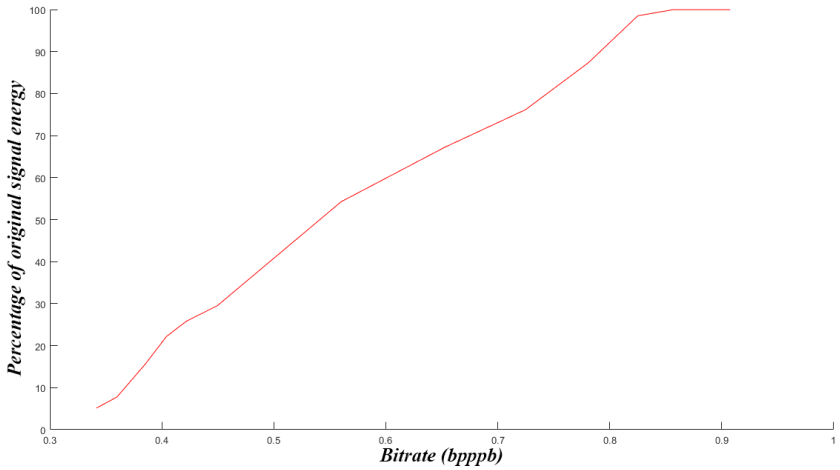


Figure 4.3: Bitrate VS Percentage energy retained using quantized coded image and shared GL-vectors for scene0 Yellowstone uncalibrated

4.2 Performance of 3D-GFT schemes

For proposed 3D coding schemes using water segmentation and without it, edge weight metrics including percent preserved energy, and PSNR are calculated in

function of percent taken coefficients. Figure 4.4 shows the comparison of preserved energy in function of percent coefficients for GFT spectral vector, 3D low and high complexity.

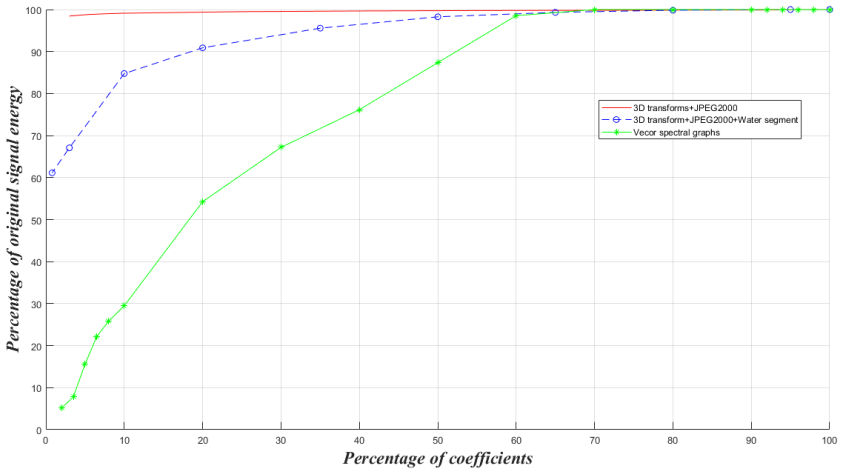


Figure 4.4: Percent preserved energy VS percent coefficients for 3D and vector GFT coding schemes

Surprisingly, adding water segmentation does not help to retrieve energy and in low quality factors, it results in destructive effect in reconstruction of coefficients. So far, 3D GFT+JPEG2000 sounds very good in terms of energy.

Figure 4.5 shows the comparison of PSNR in function of percent coefficients for GFT spectral vector, 3D low and high complexity. It seems that again without water segmentation and spatial reference update, 3D GFT does a better job and reaches a compromise in high number of taken coefficients to achieve acceptable PSNR and stays steady loosing number of coefficients.

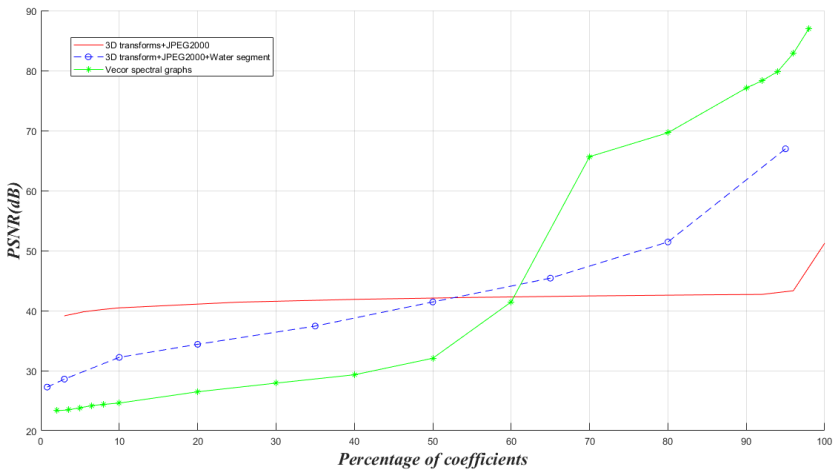


Figure 4.5: PSNR VS percent coefficients for 3D and vector GFT coding schemes

Figure 4.6 shows the comparison of rate-distortions for GFT spectral vector, 3D low and high complexity. Although, 3D GFT with water segmentation has the best rate in high section of quality factor, but it loses its performance in lower parts which makes it not practical. More steady behaviour of low complexity 3D GFT looks more like an acceptable transform coder.

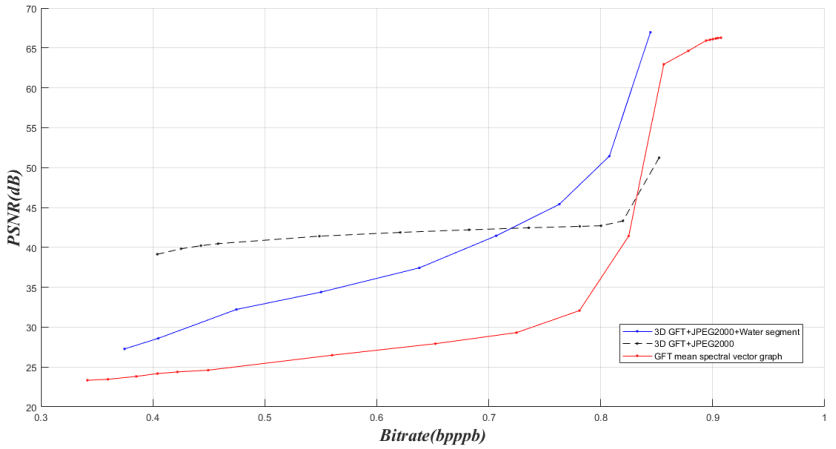


Figure 4.6: Rate-distortion for 3D and vector GFT coding schemes

Chapter 5

Conclusions

Here presents the rate-distortion result of vector spectral GFT being compared to other coding schemes applied on Hyper-spectral images [23] on the same scene. The result of comparison is demonstrated in figure 5.1

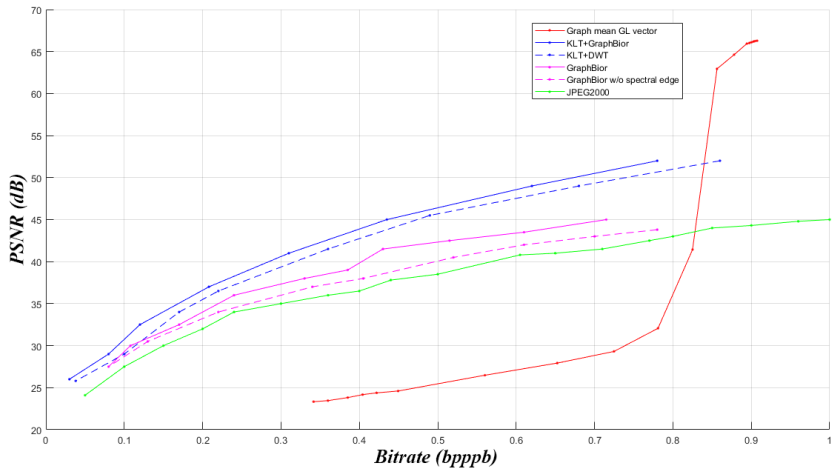


Figure 5.1: Comparison of GFT with shared mean GL-vector with other coding schemes on scene0 Yellowstone uncalibrated

For the sake of comparison, if more than 80 percent of coded coefficients are taken, the coding performance based on rate distortion results is competitive to other schemes. Taking over 85 percent of coefficients, it easily outperforms all coding schemes. However, when the percentage of coefficients decreases, PSNR drops dramatically which makes this scheme not suitable for greedy lossy compression. It may be rooted in the fact, that more sophisticated selection is not made for GL-vectors. Maybe, with a content-aware algorithm spectral vectors

been de-correlated better and results in more spars coded data and better coding performance. Also, one may exploit correlation exists in spatial domain and apply a separate 2D-transform on spatial domain data to decorrelate and improve this compression scheme using GFT on spectral direction of hyperspectral images.

Finally, rate-distortion is evaluated for both offered 3D coding schemes and compared to results of the same scene in figure 5.2.

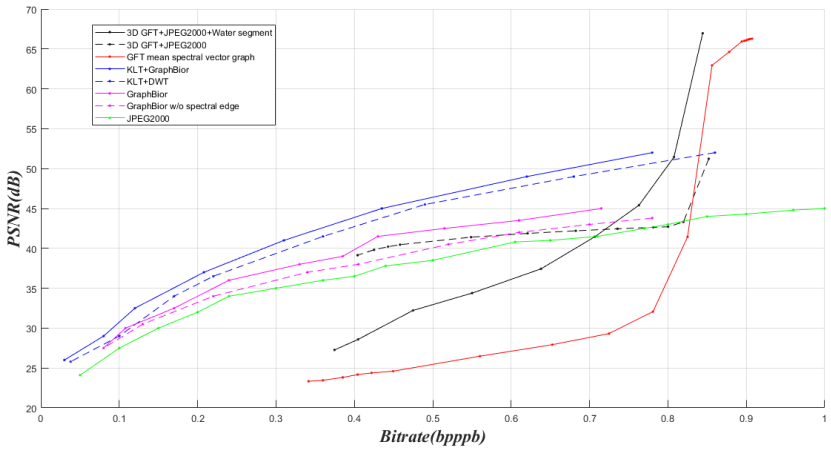


Figure 5.2: Comparison of 3D GFT coders with other coding schemes on scene0 Yellowstone uncalibrated

As is demonstrated, amongst introduced coding schemes, low complexity 3D GFT simulates an acceptable behaviour in low and high bitrate and stays steady during degradation of coefficients. It outperforms, JPEG2000 and GraphBior without spectral edges. The reason of unsuccessful performance of high complexity 3D GFT, could be destructive effect of taking badly decoded spatial signatures from previously decoded GOBs. Also specially in scene0 of Yellow stone, there are some band which are totally ruined by noise. Taking these band as spatial signature can ruin the PSNR for next spectral band of GOBs and significantly degrades image quality. For further attempts, one could check the quality of band before taking it as spatial signature and in case of high noise, take fist band alternatively, which is detectable in decoder side without extra transmission of side-information. Also, effect of updating spatial signature to last decoded band is not evaluated separately than taking individual spectral signature for water regions which could be assessed to extract and take off the destructive one and keep the advantageous technique instead.

Chapter 6

Important code Snippets

Here are some important code snippets ranging from general encoding and decoding algorithm, to how graph is shaped and Gaussian-Laplacian weights are calculated.

Build Gaussian-Laplacian Matrix code snippet

```
function [L]=build_gaussian_laplacian(I,blk_size,sigma)
l1=size(I,1);
l2=size(I,2);
W=zeros(l1*l2);

I=double(I);
%4-regular grid
for c=1:l2 %columns
    for r=1:l1 %rows
        if r<l1 %vertical connections
            W(r+blk_size*(c-1),r+1+blk_size*(c-1))=exp(-(I(r,c)-I(r+1,c))
                ^2/(2*sigma^2));
            W(r+1+blk_size*(c-1),r+blk_size*(c-1))=W(r+blk_size*(c-1),r+1+
                blk_size*(c-1));
        end
        if c<l2 %horizontal connections
            W(r+blk_size*(c-1),r+blk_size*(c))=exp(-(I(r,c)-I(r,c+1))
                ^2/(2*sigma^2));
            W(r+blk_size*(c),r+blk_size*(c-1))=W(r+blk_size*(c-1),r+
                blk_size*(c));
        end
    end
```

```
end

L=weight_laplacian(W);
```

Listing 6.1: Code snippet used to create Gaussian-Laplacian Matrix

Laplacian Weights calculation

```
function m = weight_laplacian(W,s)
% laplacian(W) --- get the Laplacian matrix of g, with matrix weight W
% equal to D-A where A is the adjacency matrix and D is a diagonal matrix
% of the degrees of the vertices.
% laplacian(W,'normalized') --- get the normalized Laplacian of g

n = size(W,1);
d = W*ones(n,1);
D = -W;
% create diagonal matrix
for k=1:n
    D(k,k) = d(k);
end


if nargin>1
   S = zeros(n);
   for v=1:n
      if (d(v)>0)
         S(v,v) = 1/sqrt(d(v));
      else
         S(v,v) = 0;
      end
   end
   D = S*D*S;
end

m=D;
```

Listing 6.2: Code snippet used to create Laplacian Weights

Encoding section

```
for i=1:h
    for j=1:w

sample_blk(:,:,1)=img(i,j,1:blk_size);
% imagesc(sample_blk);
I_vec=sample_blk(:);
L=build_gaussian_laplacian(sample_blk,blk_size,sigma);
[V, D]=eig(L);
coef=V'*I_vec;

coded_img(i,j,:)=coef;

    end
end
```

Listing 6.3: Code snippet used to encode whole image called "img"

Decoding section

```
for i=1:h
    for j=1:w

coef_rec=zeros(size(coef));
coef_rec(1:floor(blk_size/1))=coef(1:floor(blk_size/1)); %we take only
    the first xx% of the coefficients
I_blk_rec=V*coef_rec;
I_blk_rec=reshape(I_blk_rec,blk_size,blk_size);

    end
end
```

Listing 6.4: Code snippet used to decode a complete image

Bibliography

[1] G. P. Abousleman, M. W. Marcellin, and B. R. Hunt. "Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT". In: *IEEE Transactions on Geoscience and Remote Sensing* 33.1 (Jan. 1995), pp. 26–34. ISSN: 0196-2892. DOI: 10.1109/36.368225.

[2] N. Amrani et al. "Regression Wavelet Analysis for Progressive-Lossy-to-Lossless Coding of Remote-Sensing Data". In: *2016 Data Compression Conference (DCC)*. Mar. 2016, pp. 121–130. DOI: 10.1109/DCC.2016.43.

[3] and and. "Compression for hyperspectral images using three dimensional wavelet transform". In: *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No.01CH37217)*. Vol. 1. July 2001, 109–111 vol.1. DOI: 10.1109/IGARSS.2001.976072.

[4] Evgeny Belyaev, Karen Egiazarian, and Moncef Gabbouj. "A Low-Complexity Bit-Plane Entropy Coding and Rate Control for 3-D DWT Based Video Coding". In: *IEEE Transactions on Multimedia* 15 (Dec. 2013). DOI: 10.1109/TMM.2013.2269315.

[5] M. J. Black et al. "Robust anisotropic diffusion". In: *IEEE Transactions on Image Processing* 7.3 (Mar. 1998), pp. 421–432. ISSN: 1057-7149. DOI: 10.1109/83.661192.

[6] M.J. Black et al. "Robust anisotropic diffusion". In: *Image Processing, IEEE Transactions* 7.3 (1998), pp. 421–432.

[7] Ian Blanes and Joan Serra-Sagristà. "Cost and Scalability Improvements to the Karhunen–Loêve Transform for Remote-Sensing Image Coding". In: *IEEE Transactions on Geoscience and Remote Sensing* 48 (2010), pp. 2854–2863.

[8] Ian Blanes and Joan Serra-Sagristà. "Pairwise Orthogonal Transform for Spectral Image Coding". In: *IEEE Transactions on Geoscience and Remote Sensing* 49 (2011), pp. 961–972.

[9] Kerry Cawse-Nicholson et al. "An investigation of data compression techniques for hyperspectral core imager data". In: (Feb. 2019).

[10] Qian Du and James E. Fowler. "Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis". In: *IEEE Geoscience and Remote Sensing Letters* 4 (2007), pp. 201–205.

[11] G. Fracastoro and E. Magli. "Predictive graph construction for image compression". In: *IEEE* (2015). DOI: 10.1109/ICIP.2015.7351192.

[12] Jing Huang and Rihong Zhu. "Hyperspectral image compression using low complexity integer KLT and three-dimensional asymmetric significance tree". In: 7444 (Aug. 2009). DOI: 10.1117/12.826815.

[13] "JPEG2000 image coding systempart 1: Core coding system," in: *ISO/IEC 15444-1, 2000* (), Part1.

[14] Arto Kaarna and Jussi Parkkinen. "Comparison of compression methods for multispectral images". In: *Proc. norsig-nordic signal process. symp.* 2000, pp. 251–254.

[15] D. Markman and D. Malah. "Hyperspectral image coding using 3D transforms". In: *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*. Vol. 1. Oct. 2001, 114–117 vol.1. DOI: 10.1109/ICIP.2001.958966.

[16] James W. Modestino and William A. Pearlman. "Hyperspectral Image Compression Using Three-Dimensional Wavelet Coding". In: 2002.

[17] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[18] B. Penna et al. "Transform Coding Techniques for Lossy Hyperspectral Data Compression". In: *IEEE Transactions on Geoscience and Remote Sensing* 45.5 (May 2007), pp. 1408–1421. ISSN: 0196-2892. DOI: 10.1109/TGRS.2007.894565.

[19] R. Rubinstein, A. M. Bruckstein, and M. Elad. "Dictionaries for Sparse Representation Modeling". In: *Proceedings of the IEEE* 98.6 (June 2010), pp. 1045–1057. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2040551.

[20] D. I. Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2235192.

[21] H. E. SOPER et al. "The distribution of the correlation coefficient in small samples. Appendix ii to the papers of "student" and r. A. Fisher. A cooperative study". In: *Biometrika* 11.4 (1917), pp. 328–413.

[22] Xiaoli Tang and William A. Pearlman. "Three-Dimensional Wavelet-Based Compression of Hyperspectral Images". In: *Hyperspectral Data Compression*. Ed. by Giovanni Motta, Francesco Rizzo, and James A. Storer. Boston, MA: Springer US, 2006, pp. 273–308. ISBN: 978-0-387-28600-6. DOI: 10.1007/0-387-28600-4_10. URL: https://doi.org/10.1007/0-387-28600-4_10.

[23] J. Zeng et al. "Hyperspectral image coding using graph wavelets". In: *2017 IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 1672–1676. DOI: 10.1109/ICIP.2017.8296566.

[24] Jing Zhang, James E. Fowler, and Guizhong Liu. "Lossy-to-Lossless Compression of Hyperspectral Imagery Using Three-Dimensional TCE and an Integer KLT". In: *IEEE Geoscience and Remote Sensing Letters* 5 (2008), pp. 814–818.

[25] Rong Zhang et al. "A New Bit-Plane Entropy Coder for Scalable Image Coding". In: *2005 IEEE International Conference on Multimedia and Expo* (2005), pp. 237–240.