Politecnico di Torino

POLITECNICO DI TORINO

Master's Degree in Communication and Computer Networks Engineering

A.Y. 2020/2021

Performance Optimization of Hyperspectral Image Compression Algorithms

Supervisor

Prof. Enrico Masala

Candidate

Sarah Chamas

Abstract

Hyperspectral imaging is widely used in various fields. This thesis refers to a specific remote sensing application. It aims to find and implement an algorithm able to compress hyperspectral images which are used to detect the presence of ice on metallic surfaces. The basic concept is that different materials bring to different spectral signatures, which allows us to recognize them through specific software. A hyperspectral image is obtained with a special camera which collects information of the same spatial scene at different spectral bands, i.e. narrow portions of the electromagnetic spectrum. As a consequence, the amount of data to be processed is huge, and has to be reduced without losing essential information for spectral recognition. Specifically, according to the state of the art, lossless compression algorithms bring a poor compression ratio even if they preserve totally the original information, while lossy ones achieve better compression, getting rid mostly of spectral and spatial redundancy. In the thesis, the first analysed technique uses PNG and JPEG standards applied to all images corresponding to each spectral band of the HSI. Both techniques are applied in combination with a two-dimensional Wiener filter, in order to get rid of the noise and obtain a better compression ratio. Another implemented and studied lossless technique is based on the Recursive Least Squares (RLS) filter, used for pixel prediction on the previous spectral bands. Eventually, the last algorithm investigated in this thesis, provides hyperspectral data representation in tensor notation and it is based on the combination of the two-dimensional Discrete Wavelet Transform (2D-DWT) and the Tucker Decomposition in its Alternating Least Square (ALS) implementation.

Acknowledgements

Most of all, I would like to express my appreciation to my supervisor, Professor Enrico Masala. I want to thank him for his willingness and for the opportunity of this thesis, from which I could learn much.

A special thanks goes to my parents, who taught me to be determined and kind, and to all my family, to always be supportive and caring.

I should like to thank my friends, old and new ones, for bringing joy and laughter in moments both easy and difficult. Thanks also to my student association, the Mu Nu Chapter of IEEE-Eta Kappa Nu, this experience gave me a way to grow both professionally and as a person and to meet many amazing people.

Dulcis in fundo, thanks to Giovanni, my life partner and my port in a storm.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, multispectral and hyperspectral imaging (HSI) are used for many different applications, such as detecting some types of cancer and retinal diseases, forensic laboratory, exploration of oil and gas, environmental monitoring on pollution levels, biomedical analysis, food quality and control and remote sensing. Through the measurements acquired by a special sensor, i.e. the hyperspectral (or multispectral) camera, it is possible to categorize vegetation, surfaces and atmospheric conditions according, indeed, to the spectrum behaviour. Hyperspectral and multispectral images, in fact, differently from traditional digital images, contain not only spatial information, but also spectral one. Usually, they are referred to as "data cubes" or "hypercubes", since data are organized in a three dimensional structure: spatial (bidimesional) and spectral. The data cube is composed by images acquired at the same time at multiple wavelengths, or alternatively it can be seen as a set of spectra corresponding to each pixel of the image. The difference between hyperspectral and multispectral images lies with the wavelength bands used to acquire them: if they are consecutive it is the case of hyperspectral images, otherwise it is multispectral one.
Many of the application scenarios have low bandwidth occupancy requirement and/or limited storage available, while the total load of HSIs is huge. As a consequence, the amount of data has to be reduced, but without loosing a large amount of information, otherwise spectral signature recognition could be compromised. It, in fact, has to be preserved as it is used for feature extraction and recognition, and classification. In order to do so, some techniques have been developed during the years. On one hand, lossless compression algorithms bring a poor compression ratio, even if they preserve totally the original information, on the other lossy ones achieve better compression, getting rid mostly of spectral and spatial redundancies. Consequently, the need is finding a good compromise between complexity and amount of discarded information, depending on the dataset and the application field.

This work refers to a remote sensing case, specifically it aims to study and implement algorithms able to compress hyperspectral images, which are used to detect the presence of ice on metallic surfaces. Specifically, it aims to analyse and test some algorithms available in the state of the art to compress some hyperspectral images, acquired in laboratory, simulating ice on metallic surface in different conditions.
The thesis content is organized as follows: at first the basics to understand principles of hyperspectral imaging are given, focusing on the electromagnetic radiation and sensor characterization. Here it is also explained what spectral signatures are, and why they are subject to variability. Going on, some compression techniques used in hyperspectral imaging as well as digital images are illustrated, in particular main transforms and prediction filters are explained in detail. Eventually, the state of the art of HSI compression algorithm is presented. Next chapter, instead, contains the comprehensive description of the working principle of the investigated ones, i.e. PNG and JPEG applied to each per-band image of the hypercube, the solution proposed by Song et al. in [1] which uses Recursive Least Square Filter (RLS) for lossless compression, and the one presented by Karami et al. in [2], which instead works with tensors. Indeed, this solution foresees the usage of two dimensional discrete wavelet transform (2D-DWT) and of the Tucker decomposition (TD). Finally, the tested data are illustrated and implementation and experimental choices for each of the studied solutions are explained. Results are shown comparing the ones obtained with different techniques and different features.

Chapter 2

Hyperspectral Remote Sensing

Hyperspectral Imaging (HSI) consist of data collected through special remote sensing devices, such as air-crafts and satellites, which contain spatial and spectral information. Usually, hyperspectral images are presented as "data cubes" or "hypercubes", since information is organized in a three dimensional (x, y, λ) structure, with (x, y) spatial dimension and λ spectral one (wavelength). The data cube is composed by images acquired at the same time at multiple wavelengths, or alternatively as a set of spectra corresponding to each pixel of the image.[3]

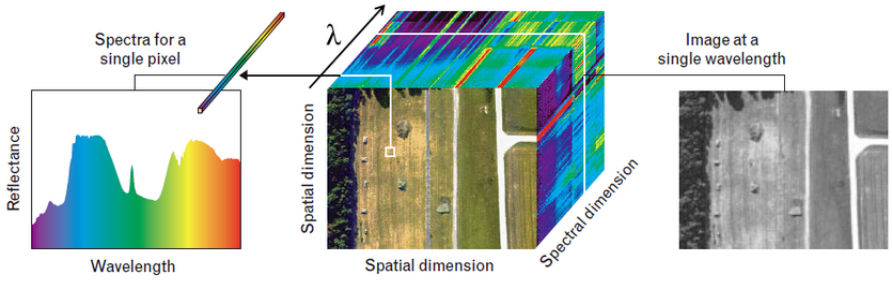


Figure 2.1: Illustration of a HSI data cube.[4]

Comparison between RGB image and hypercube is presented in Fig.2.2. HSI combines imaging and spectroscopy, in fact, as just explained, the generated hypercube is a 3-D dataset, covering a contiguous portion of the light spectrum and with higher spectral resolution with respect to multispectral imaging, which is another similar technique used in remote sensing applications. Actually, if hyperspectral imaging covers up to few hundreds of spectral bands, multispectral imaging uses up to few dozens (usually around 30) of non consecutive spectral bands.[5] Multispectral

imaging produces discrete spectral information, while hyperspectral one translates onto continuous spectral reflectance (spectral signature), which is the key parameter for applications in different areas, as it will be explained later on in this chapter.

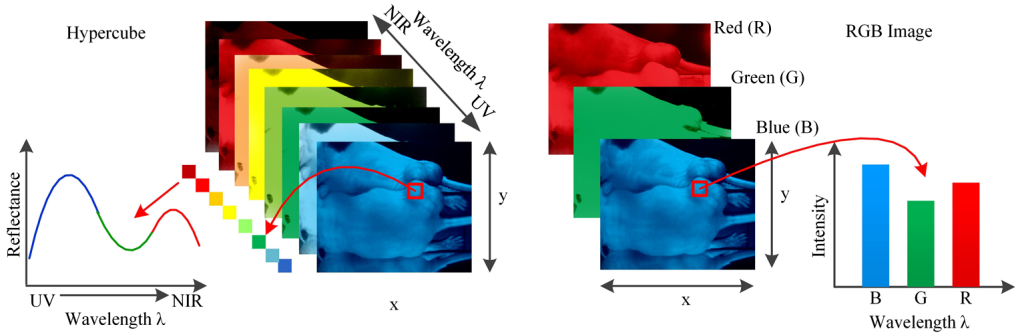


Figure 2.2: Comparison between hypercube and RGB image.[5]

HSI is widely used in many fields, depending on the need: it could be for detection of vegetation features, in order to monitor pollution effects or to counteract narcotics; it can be used to find man-made materials in natural environments, helping in rescuing operations, or it could help in resource management. HSI is employed also for studying food compositions, in biomedical and many other fields.
All of these application are related to remote sensing, which includes acquisition, processing and interpretation of images, related data obtained by recording the interaction between matter and electromagnetic radiation. The source of this radiation can be natural (e.g. earth's emitted heat and sun's reflected light) or man-made, i.e. that produce itself radiation. Indeed, systems can be classified as active and passive, depending on how the radiation is emitted and then analysed. In Fig.2.3 are depicted the two cases, notice that range information can be provided only in the case of active systems.[3]

Figure 2.3: Remote sensing with active and passive sensors.[6]

2.1 Electromagnetic Radiation and Sensor Characterization

The electromagnetic radiation is the means by which energy propagates in the form of a waves. Electromagnetic waves are located within the electromagnetic spectrum (see Fig.2.4) depending on their parameters, i.e. wavelength and frequency. In hyperspectral imaging only a portion of electromagnetic spectrum is exploited, from 0.4 [μm] to 14 [μm]. More precisely, the exploited windows are the visible one (**V**), which includes Blue, Green and Red wavelengths, from 0.4 to 0.7 [μm], Near Infrared (**NIR**) from 0.7 to 1.1 [μm], Short Wave Infrared (**SWIR**) from 1.1 to 2.5 [μm], Mid Wave Infrared (**MWIR**), from 3 to 5 [μm] and Thermal or Long Wave Infrared (**TIR** or **LWIR**), form 5 to 14 [μm]. In Tab 2.1 are reported in details used wavelengths in hyperspectral imaging, with their relative application fields.[3]

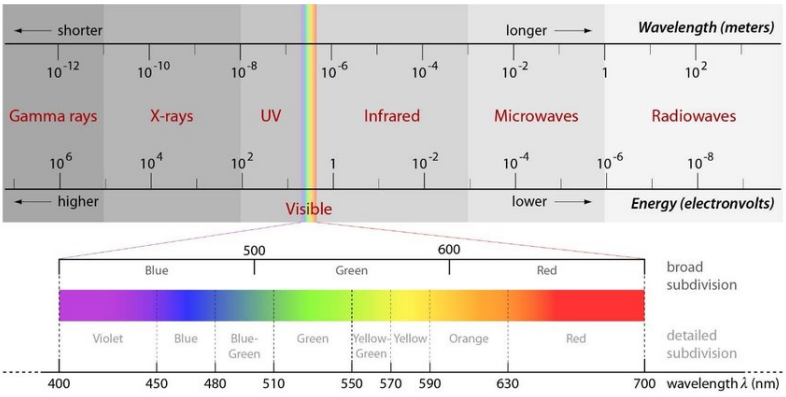


Figure 2.4: Complete electromagnetic spectrum.[7]

In V and NIR regions the radiation emitted from the sun is modified by earth's atmosphere and surface (reflective energy is predominant), while in TIR region the study is focused on the radiation emitted by the earth's atmosphere and surface (emissive energy is predominant).[3]

Table 2.1

Band	**Wavelength**	**Property**	**Application**
Blue	400-500 nm	Reflective	Illuminates material in shadows, water penetration for bathymetry
Green	500-600 nm	Reflective	Water penetration for bathymetry, discrimination of oil on water
Red	600-700 nm	Reflective	Limited water penetration for bathymetry, vegetation differentiation
NIR	700-1100 nm	Reflective	Artificial material detection, shoreline mapping, vegetation analysis
SWIR	1100-3000 nm	Reflective/ Emissive	Discrimination of oil on water, snow/cloud differentiation, change detection, artificial material detection, plume detection
MWIR	3000-5000 nm	Emissive	Nighttime target detection, ocean temperature analysis, daytime reflected/emitted thermal analysis, nighttime thermal analysis, smoke penetration

Table 2.1

Band	Wavelength	Property	Application
TIR or LWIR	5000-14000 nm	Emissive	Thermal analysis, vegetation density and cover type, gas detection and identification, mineral and soils analysis

In remote sensing, imaging spectrometers, most commonly known as hyperspectral sensors, collect at once digital images in several contiguous narrow spectral bands of the just described windows of the electromagnetic spectrum. They can be classified as active or passive sensors, depending on the source of the electromagnetic radiation which carries the radiant energy measured and converted into data from the device itself. To successfully complete the acquisition procedure, all imaging spectrometers require some sort of positioning and scanning systems, allowing to accumulate data assigning the three coordinates (x, y, λ) to each pixel. Consequently, to construct the hyperspectral cube spatial, spectral and radiometric sampling are needed. Temporal sampling, instead, refers to how often data are obtained for the same area and it has to be considered as an operational procedure, since it is not interesting from a signal processing perspective.[3]
Focusing on passive remote sensing, there are two main sources of radiations collected by the sensors: in the V to the SWIR it is the sun, while in TIR (or LWIR) it is the thermal radiation, which is emitted directly by materials and combines with self-emitted thermal radiation in the atmosphere as it propagates upward. In fact, part of the radiation received by the sensor has been reflected at earth's surface and possibly scattered by the atmosphere. Solar radiation can be considered to be emitted at maximum efficiency possible for a body at its effective temperature, indeed sun is approximated as a near-perfect *blackbody* radiator. Therefore, spectral radiance exitance from the sun is described by *Plank's equation*. One important physical dimension in HSI is *spectral irradiance*, as it measures the radiation on top of the atmosphere. Units of spectral radiance exitance and spectral irradiance are the same (i.e. $W/m^{-2}/\mu m^{-1}$), so they can be referred to as *spectral flux density*. It can be demonstrated that as the wavelength increases to the SWIR, less radiation is available from the sun for signal detection by remote sensing. Furthermore, radiation components in the visible to shortwave infrared region are depicted in Fig.2.5.[8]

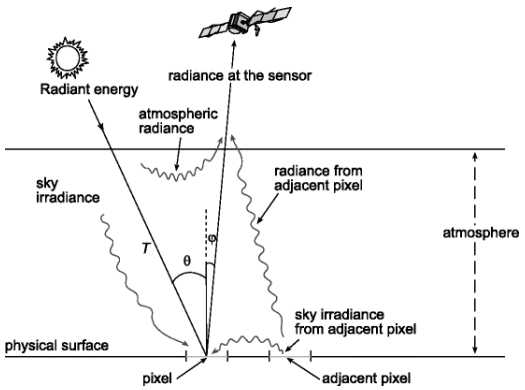


Figure 2.5: V to SWIR region: radiation components.[9]

In the electromagnetic window between SWIR to MWIR, beside solar radiation, also thermal radiation becomes a relevant component in spectral analysis. It is emitted by objects named *Lambertian* reflectors. Increasing wavelengths up to TIR region of the spectrum, direct solar radiation is not a factor compared to the self-emitted thermal radiation, exception for some objects named *Specular* reflectors. The *emissivity*, which is a function of wavelength, is an emission efficiency factor and it regulates the radiation emitted by a real object which is not a blackbody. Actually, real objects are not perfect absorbers or emitters. Thus, the energy collected by a remote sensing system is proportional to the solar effective region and to the thermal region.[8]

2.2 Spectral Signatures

The main scope of remote sensing is to identify objects, but, since spatial resolution of satellite-based sensing systems is too low to do it by shapes and spatial details, object identification is done by spectral measurements. In detail, in HSI remote sensing it is done with *spectral signatures*. The spectral signature of a material, in solar-reflective region, can be defined by its reflectance as a function of wavelength, measured at an appropriate spectral resolution. In TIR region of the electromagnetic spectrum, instead, signatures of interest are emissivity and temperature.[8]

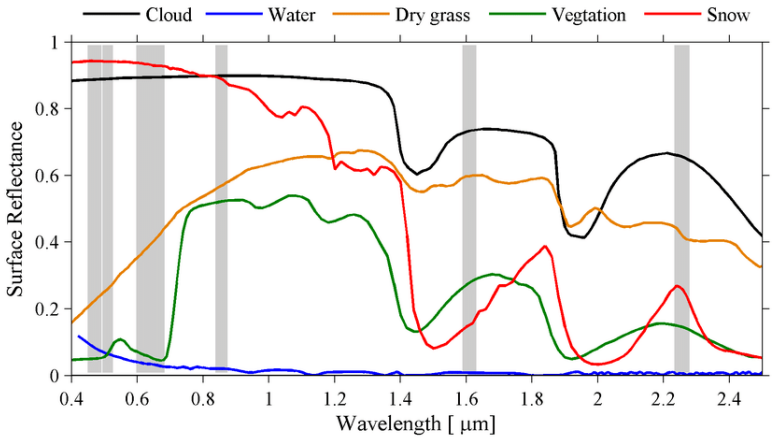


Figure 2.6: Example of typical surfaces' spectral signatures in in V and NIR regions of the electromagnetic spectrum.[10]

Each material is characterized by its unique spectral signature, described by spectral reflectance in V, NIR and SWIR regions, and by spectral emittance in LWIR region of the electromagnetic spectrum. Unfortunately, supposing to have a perfect spectral signature predetermined for each material is not realistic, but it is an ideal concept, difficult to be observed in actual applications. In fact, in a real case scenario there are many sources of variability that affect measured radiance of materials, affecting the performance of hyperspectral image exploitation, particularly in the case of target detection.[11]
There are more than one sources of target spectral variability. First of all, it can be the target material itself: it is very rare to observe in natural environment perfect pure materials, it is likely they will have some variability in their composition. This modification can find explanation due to different reasons. One of the most direct source of spectral variability is due to some chemical variation (e.g. oxidation or hydration), or due to morphology of the observed surface. If this is the case, it is called *intrinsic* target variability. Spectral signatures, in fact, depend on the reflectance properties of the solid material, that, in turns, is characterized by the refraction index and the extinction coefficient of the material, according to which it is possible to derive modeled reflectance signatures based on morphology (e.g. particle size or packing density). Consequently, obtaining overall spectral measurements over the multitude of possible mythologise is not viable, and there is the need of some models capable of generating simulated spectra. Moreover, for solid materials, reflectance depends on both the angle at which the surface is observed and the one from which it is illuminated. These models are evaluated according to the bidirectional reflectance distribution function (BRDF), which describes how light reflects off an opaque surface; it is a function of radiance and

irradiance, thus depends also on wavelength. Geometry of BRDF is described in Fig.2.7[11]

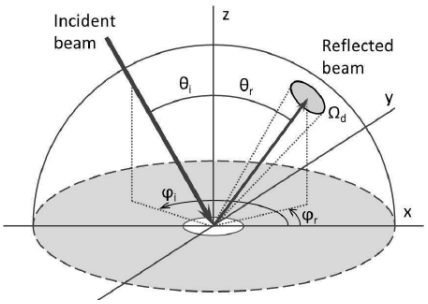


Figure 2.7: Geometry definition of BRDF.[12]

Regardless of the nature of the material, spectral variability can depend also on how much material there is relatively to pixel size, on concentration and thickness of the matter and, in the LWIR, also on temperature, since warmer materials emit more radiation and seem brighter to the sensor. These just described factors belong to case of *extrinsic* target variability. This type of variability can be described through several model depending on the observed object nature (i.e. solid, gas-phase, thin layer of powder, plumes or others). [11]
Target spectral variability not only depends on the target itself, but it is also caused by the surrounding environment, which is the case of *environmentally induced* target variability. This type of sources can be divided in four categories: variation can be caused by atmosphere, illumination, acquisition geometry or adjacent environment. Specifically adjacent environment source includes adjacent materials, nearby objects and obstacles, shadows and sensor viewing angle; acquisition geometry source, instead, rely on sun position and topography.[11] Main topics of atmospheric effects are absorption and scattering, which strongly affect spectral signatures. For this reason atmospheric correction and radiometric calibration is needed on acquired data.[3]
In conclusion, reading and interpreting hyperspectral images acquired through remote sensing is strongly related to spectral signatures, which are subject to variability. In fact, it is with spectral signatures that is possible to perform detection and classification of different materials in the observed environment. The point of studying this target variability is to be able to differentiate which distinctions are "good" ones-and so allow to differentiate natural versus man-made objects, target versus background, healthy vegetation versus dead one, etc- and which are the "bad" ones, i.e. what confuses the analysis, such as dry or humid atmosphere, sunlit versus shadows, etc. However, it is possible to characterize this variability through algorithms once the target variability model has been defined.

Chapter 3

Hyperspectral Images Compression

Hyperspectral images obtained by sensor are presented as data-cubes: three dimensional structure containing spatial information (x, y) at different spectral bands, i.e. wavelengths λs. They can be viewed as as many sub-band images as the acquisition bands, i.e. for each band there is the same number of pixels and the resolution (number of pixels) depends on the capability of the sensor. Size of one hyperspectral image depends on both resolution and number of acquisition bands which usually are in the range going from approximately 100 to 300 or more wavelengths. Considering $N_{pixels} = x \times y$ the number of pixels per sub-band image, N_{bit} the number of bits per pixel (e.g. 16), then: $size = N_{pixels} \times N_{bit} \times N_{bands}$. Size's order of magnitude is around hundreds or thousands of megabyte, hence reducing dimensions of collected data is fundamental for their storage.[13]
Another issue is limited transmission channel bandwidth, since required bandwidth is proportional to transmitted data size and it is the main factor in terms of application cost. Also the transmission time is a key parameter to be monitored, in fact it regulates calibration and information losses at data centers, once information is received from sensors. For all these reasons, data compression is fundamental stage in hyperspectral imaging processing. As in traditional image compression, there are many algorithms designed to reduce size. One parameter to be taken into account useful to this scope is redundancy; it in fact, in general, it allows compression algorithms to achieve better compression ratio (CR), i.e. the ratio between the number of bits needed to represent the original uncompressed image and the ones needed for the compressed version. Compression algorithms decorrelate redundancies if present and thus data size is reduced. This procedure is usually reversible, and original data can be reconstructed or approximated through inverse procedure.[13] Types of redundancy and their classification are represented

in Fig.3.1.

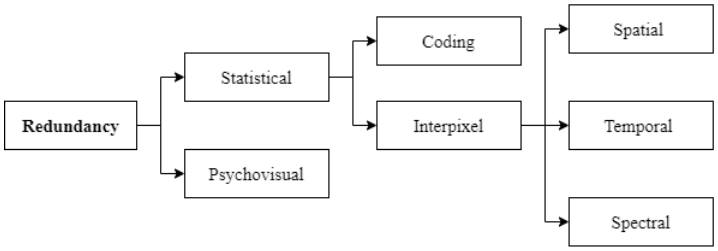


Figure 3.1: Redundancies' classification in image processing.

Statistical redundancy occurs due to similar intensity of pixels in neighborhood, except where illumination changes, while psychovisual is related to human perception: human brain distinguish different content on the visual scene and not all its components have the same importance for understanding it. Then, coding redundancy involves the usage of codes to reduce the amount of data used to represent the visual information. In traditional image processing interpixel spatial redundancy, is due to the correlation of neighboring pixels, since they are not statistical independent; in fact, value of current pixel can be predicted using its neighbors values. In HSI specifically, spatial redundancy occurs due to intra-band dependency that exist in spatial domain. Going on, interpixel temporal redundancy occurs when HSI of the same location is taken at different times, this, in fact, will generate a dependency in time domain for corresponding spectral and spatial pixels. In digital image processing, instead, it refers to successive frames in video sequence, indeed it is also called inter-frame redundancy. It is exploited when motion compensation is applied. Eventually, spectral redundancy occurs due to the dependency among pixels of different bands at the same spatial location.[13]

3.1 Compression Techniques

There are many techniques used and combined together to compress a hyperspectral image based on which HSI compression algorithms can be categorized, such as vector quantization, whose significant steps are training (codebook generation) and coding (vector matching), compressive sensing, widely used in real time application, in fact it includes different complexity for encoding (low) and decoding (high) procedures, tensor decomposition, learning based algorithms and many more. However, the most widely used can be identified in transforms and prediction

based techniques, since they exploit interpixel redundancy efficiently to reduce digital image size or data cube size in the case of hyperspectral or multispectral images.

3.1.1 Transforms

As in digital signal processing, also in HSI compression, transform based techniques act by transforming pixel values into a new domain, more suitable for quantization and coding, by applying a function to the three dimensional (x, y, λ) data-cube. Actually, most of the signal energy is contained in few coefficients in the transformed domain, for this reason it is possible to achieve compression: only significant values of the transformation result are kept. In the case of linear transforms, they can be interpreted as a decomposition of input data (i.e. vector if one dimensional, matrix if bi-dimensional, and so on) in terms of a basis set. The most known are Fourier Transform, used in its discrete version (DFT), Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT) and Karhunen-Loeve Transform (KLT). All of them are able to decorrelate data, both in spectral and spatial domain. These transformation are reversible, so that it is possible to recover original uncompressed data. They generate coefficients and then quantization is applied to remove factors that are close to zero. Bit streams are generated by encoding quantization output, which are then transmitted or stored. Even if they can vary, most common steps of algorithms that use transform based compression are shown in Fig.3.2 [13]

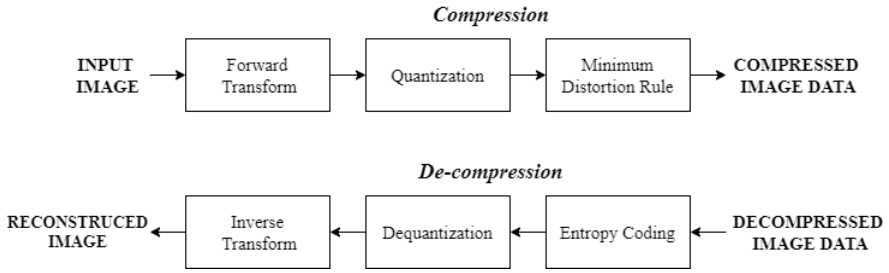


Figure 3.2: Common steps of transform based algorithms.

Starting from 1D case, it is possible to define a transform as a mathematical reversible function **T** that given a vector of N elements **x**, it returns **y** with the same number of elements, in formulas:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} T(n, i)x[i] \tag{3.1}$$

$$x[n] = \frac{1}{N} \sum_{i=0}^{N-1} T^{-1}(n, i)y[i] \tag{3.2}$$

In detail, DFT uses sine and cosine waves with increasing frequencies as basis functions. In the case of simple digital image with size $N \times N$, which in the case of HSI can be a sub-band image (i.e. spatial information (x, y) corresponding to one of the wavelengths), the transformation is two dimensional, 2D-DFT. It is obtained by multiplying the image $\mathbf{I}_N$ by the corresponding basis function summing the result (3.3).

$$F(k,l) = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} I(n,m) e^{-j2\pi\left(\frac{kn}{N} + \frac{lm}{N}\right)} \tag{3.3}$$

Inverse transform, instead, is 3.4. Its complexity is $O(N^2 \log_2 N)$ if Fourier transform is implemented in its fast version (FFT).[14]

$$I(n,m) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) e^{j2\pi\left(\frac{kn}{N} + \frac{lm}{N}\right)} \tag{3.4}$$

KLT, also called Hotelling Transform or Principal Component Analysis (PCA), instead, is based on eigenvector decomposition of covariance matrix. Considering $\mathbf{X} = [X_1, ..., X_N]$ a random process of length N, $\mathbf{A}$ a matrix of size $N \times N$ whose columns correspond to the eigenvectors of the covariance matrix $\mathbf{C}$ of $\mathbf{X}$ arranged in increasing eigenvalue order, and $m = [m_1, ..., m_N]$ the vector containing the mean values $m_i = E[X_i]$, then the KLT transform is defined as:

$$Y = A^T(X - m) \tag{3.5}$$

where $A^T A$ is the identity matrix I. $\mathbf{A}$ is the KLT matrix and it diagonalizes the covariance matrix for the input process. It is such that the elements in the transformed coefficients vector are uncorrelated, i.e. $\mathbf{Y}$ has no correlation. This transform minimizes the Mean Squared Error (MSE) between the original values and the selected k from the inverse transform result, as it packs most of the energy in the first k coefficients. The main problems with KLT are the requirement of training data, the calculation of eigenvalues and eigenvectors, which is computationally intensive (vector-matrix product is $O(N^2)$)and not very stable, and data adaptation to the transform, since the output $\mathbf{Y}$ depends on $\mathbf{C}$, which in turn depends on the training data, while, for example, Fourier transform basis functions are independent from the input image.[14]
Discrete Cosine Transform, instead, similarly to the Fourier transform, uses cosine waves as basis functions. It is applied to the images in its 2D version (separable multidimensional extension). As previously explained, in the case of hypercubes, transforms are used in their bi-dimensional version, so that they can be applied to the sub-band images (x, y) with dimensions $N \times N$ for each wavelength λ of the HSI. In formulas direct and inverse 2D-DCT is explained in 3.6 and 3.7, where $f(n, m)$ is the input original data and $\alpha[u] = \sqrt{\frac{u}{N}}$ and $\alpha[v] = \sqrt{\frac{v}{N}}$, with $u, v = 1, ..., N-1$.[14]

$$C[u,v] = \alpha[u]\alpha[v] \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f[n,m] cos\left(\frac{(2n+1)u\pi)}{2N}\right) cos\left(\frac{(2m+1)v\pi)}{2N}\right) \quad (3.6)$$

$$\tilde{f}[n,m] = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha[u]\alpha[v]C[u,v] cos\left(\frac{(2n+1)u\pi)}{2N}\right) cos\left(\frac{(2m+1)v\pi)}{2N}\right) \quad (3.7)$$

DCT transform is used in JPEG, since it is very simple to be implemented in hardware and solve the discontinuity problem of the DFT implementation (see Fig.3.3). DCT in fact uses only odd numbers as argument of the the cosine basis, doing a "mirror copy" and eliminating the discontinuity between two consecutive periods. In digital image processing this means that lines can be approximated through fewer coefficients.

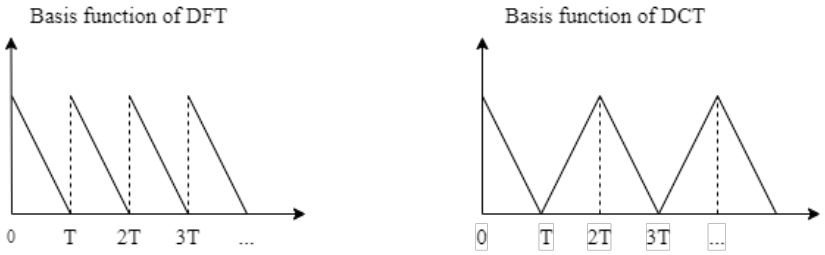


Figure 3.3: DFT and DCT periodicity.

Since 2D-DCT is implemented as a horizontal-vertical separable transform, it is very good at compacting the energy of horizontal and vertical discontinuities, but the need of transforms that can adapt to specific directions of the image feature led to Directional DCT implementation. Instead of scanning by rows and then by columns, it acts choosing an angle and then diagonalizing. The ability of choosing a good angle which is optimal directly on the image leads to maximum coding efficiency, even if this value has to communicated also to the decoder, increasing the overhead. Steps of Mode 3 Directional DCT with 45° angle are showed in Fig.3.4.[14]

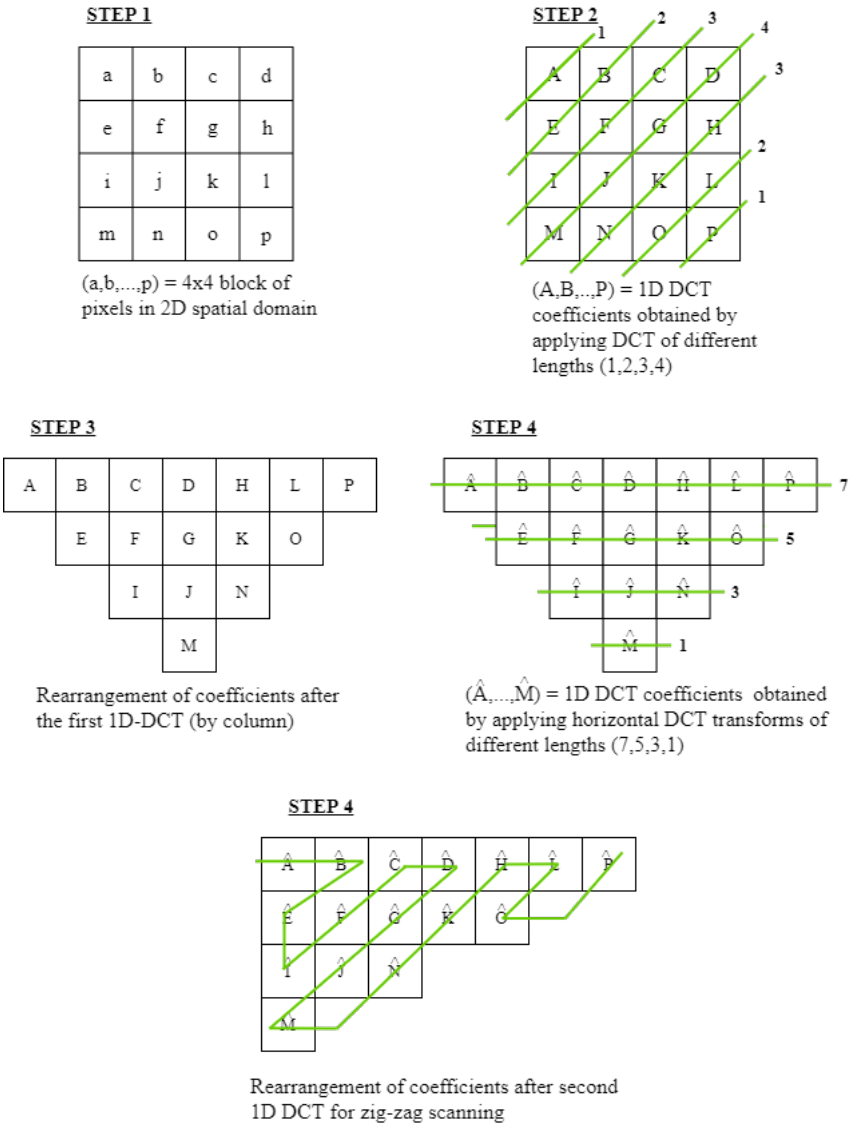


Figure 3.4: Steps of Mode-3 Directional DCT.

Lastly, another very used technique is Discrete Wavelet Transform. Wavelet expansion, in fact, allows to more accurate local description and separation of the signal characteristics. Its coefficients represents a local component itself, being easier to interpret. Wavelet transform based techniques, in general, may allow to a separation of signal's components that overlap in time and frequency. Wavelets are adaptable and adjustable, and consequently ideal for adaptive systems. Moreover, their generation and the calculation of DWT is well matched to digital computer: all the operation (additions and multiplications) are very easily implementable in electronics. The DWT is rarely used in its mathematical form, while it is most

common to use its filter bank implementation. A filter bank is a structure that decomposes a signal into a collection of sub-signals that convey salient features of the original one and are sufficient to reconstruct it. In signal compression, the *analysis* filters (high-pass filters) are used to filter the original signal, which is then down-sampled by a factor M to give a sub-band signal. The original one is recovered by up-sampling and filtering with *synthesis* filters. Perfect reconstruction is possible by wisely choosing the design of filters.[15]

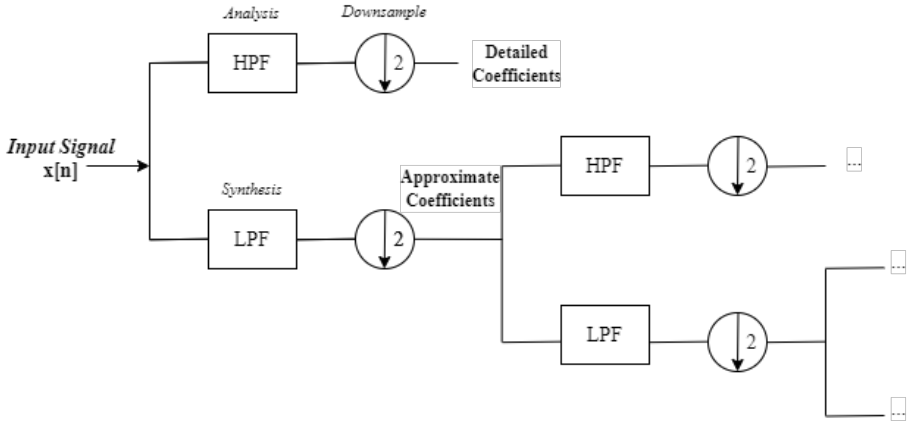


Figure 3.5: DWT Filterbanks.

The high-pass filter *HPF*, at each level i, produces detailed information D_i, while the low-pass *LPF* associated with scaling function, produces the approximate information A_i. Filters' coefficients are chosen according to the characteristics of the input signal, coherently with the available wavelets functions (Biorthogonal, Daubechies, Coifilet, etc.) and they have to guarantee the perfect reconstruction (PR). In this sense they have application dependent properties as, for example, frequency selectivity. In the case of image compression, 2D-DWT is used and, furthermore, in the case of hyperspectral images, 2D-DWT is applied to spatial content of each spectral band of the datacube. Discrete Wavelet Transform is used in the JPEG 2000 standard instead of the DCT to perform decomposition of the image. Fig.3.6 shows how the 2D transform is obtained by applying the 1D transform first rows-wise and then columns-wise (at each iteration 4 sub-bands are obtained). All four final components (LL, LH, HL and HH) will have same final size of $\frac{N}{2} \times \frac{N}{2}$. Similarly to the DCT, DWT compact most of the signal energy into the lower frequency sub-bands, and most of the coefficients in higher sub-bands are small or nearly zero.[16]

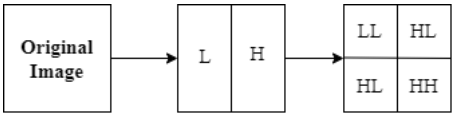


Figure 3.6: 2D-DWT: Sub-band decomposition of $N \times N$ image

Transform based techniques have many advantages, including high compression performance, global optimum solution, fault tolerance mechanism. Thanks to fast calculation, they can be adopted in on-board compression and data centers. Main disadvantage of this approach is high computational time, caused by computations included in transform procedure (multiplications, transpose, inverse matrix, etc.). However, it can be decreased with parallelism.[13]

3.1.2 Prediction Based Techniques

As an alternative to transform based algorithms, compression can be achieved by using prediction techniques. In this case, pixel value is predicted through mathematical methods based on the value of the previous pixels and both spectral and spatial correlation are exploited and removed. In HSI, prediction is applied mostly on the spectral domain through filtering. Most used filter functions are Recursive Least Squares (RLS) and Least Mean Squares (LMS) filter. Prediction based methods are easy to implement on HSIs and their steps rely on what depicted in Fig.3.7. [13]

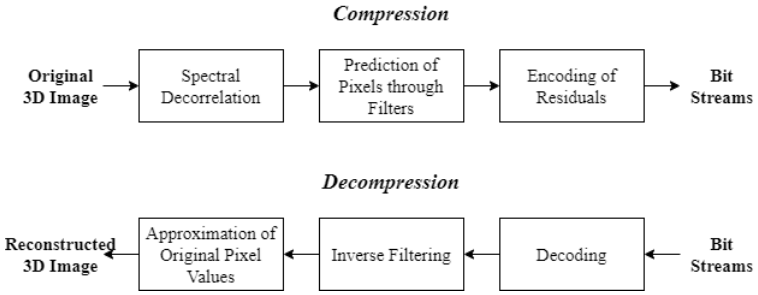


Figure 3.7: Common steps of prediction based algorithms.

While spectral decorrelation is performed for all of the bands of the HSI, the prediction step in Fig.3.7 acts only on *p-1* bands, where *p* is the number of prediction bands used to evaluate the pixel estimate through mathematical operations. This can be done by using a weight matrix, which is generated by a filter function that varies according to the implemented algorithm. After image is predicted, it is subtracted to the original one, generating the residual, which, on turn, is encoded. Usually, Golomb or entropy coders are adopted.[13]

Going in detail, the adaptive Least Mean Square filter mechanism relies on the Stochastic Gradient Descent (SGD) algorithm, whose scope is to determine the gradient (i.e. the direction) from one adaptation cycle to the next. The LMS algorithm computational complexity scales linearly with the dimensionality of the FIR filter on which it operates, in this sense it is said to be simple. Moreover, it does not require knowledge of statistical characteristics on the environment on which it operates and it is robust in each single realization of the algorithm.[17]

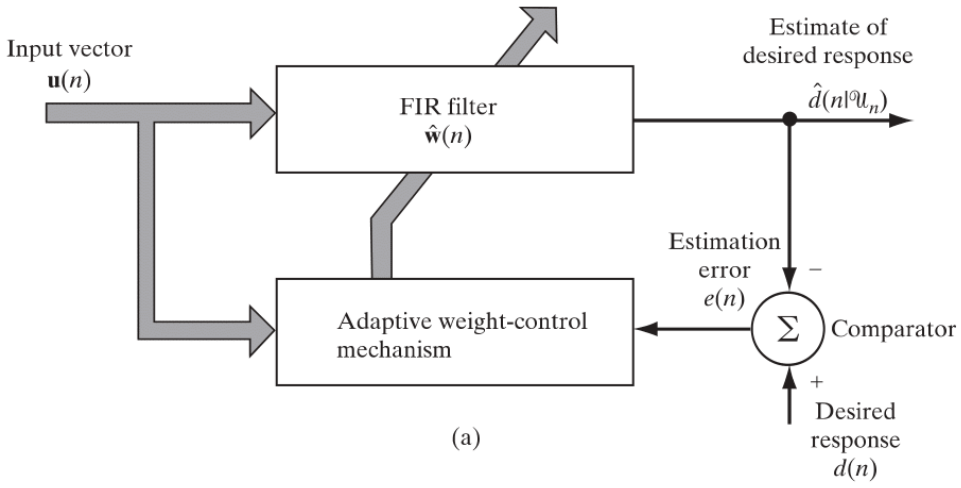


Figure 3.8: LMS algorithm block diagram.[17]

The block diagram depicted in Fig.3.8 is composed by the three components of the LMS algorithm, i.e. a *FIR filter* that produces an estimate of the desired response, a *comparator*, which subtracts the estimate from the desired response producing a estimate of the error, and an *adaptive weight-control mechanism* that regulates the adjustments applied to the tap weights of the FIR filter by exploiting information contained in the estimation of the error. Table 3.1[17] summarizes how the LMS algorithm works.[17]

Summary of the LMS Algorithm
Parameters:
M = number of taps (i.e. filter length) μ = step-size parameter $0 < \mu < \frac{2}{\lambda_{max}}$, with λ_{max} maximum value of correlation matrix between tap inputs *u(n)* and filter length.
Initialization:
If poor knowledge of the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select and appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(n) = \mathbf{0}$.
Data:
Given: $\mathbf{u}(n)$ M-by-1 tap-input vector at time n $\mathbf{u}(n) = [u((n), u(n-1)), ..., u(n-M+1)]^T$ *d(n)* = desired response at time n. ***To be computed:*** $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time $n+1$.
Computation:
For n = 0,1,2.... compute $e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)$ $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e*(n)$.

Table 3.1

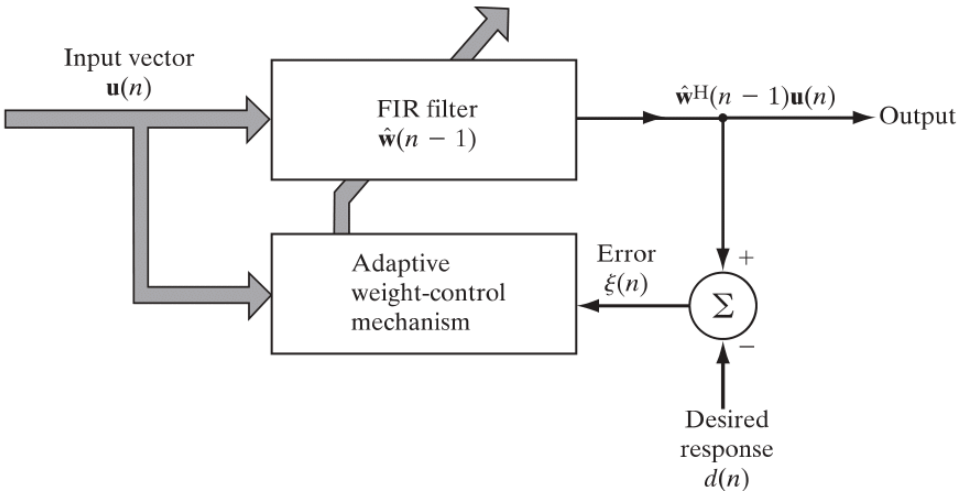


Figure 3.9: RLS algorithm block diagram.[17]

In terms of convergence, a faster alternative to the simple LMS algorithm is the Recursive Least Squares (RLS) one, which whitens the input data by using the inverse correlation matrix of the data, assumed to be zero mean. This improvement is obtained at expense of an increase in computational complexity. In particular, the step size parameter μ in the LMS algorithm, here is replaced by the inverse of the correlation matrix of the input vector. RLS algorithm is composed by two main stages, *filtering* process and *adaptation* process, and its computation follows a square law. In Fig.3.9 is presented the block diagram representation of the RLS algorithm. It requires to specify to quantities, i.e. the initial weight vector, which is commonly at first equal to the all-zeros vector, and the initial correlation matrix, which depends on a regularization parameter δ, assigned according to the signal-to-noise ratio (SNR). In Tab.3.2 steps of the RLS algorithm are summarized.[17]

Summary of the RLS Algorithm
Initialize the algorithm by setting:
$\hat{\mathbf{w}}(0) = \mathbf{0}$, $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, and δ equal to small positive constant for high SNR and large positive constant for low SNR
For each instant of time n=1,2,... compute:
$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H\mathbf{P}(n-1)\mathbf{u}(n)}$ $\xi(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$, $\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$, and $\mathbf{P}(n) = -\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{P}(n-1)$.

Table 3.2

It is important to point out that, in the context of adaptive filtering algorithms, there is a trade-off between deterministic robustness in the face of uncertainties on the one hand and statistical efficiency on the other. The LMS algorithm, being model-independent in statistical terms, provides a suboptimal solution, but, by assign a small enough value to its step-size parameter, it is possible to ensure robustness. The loss to do so is in terms of statistical efficiency, i.e. long rate of convergence. The RLS algorithm, instead, is model-dependent and its derivation is based on the method of least squares, which is pivoted on a multiple linear regression

model for describing the desired response. In this sense, it is optimal from the statistical efficiency (faster convergence rate with respect to LMS algorithm by more than one order of magnitude), but the price to pay is robustness. Consequently, resolution of this trade-off is possible only by considering the practical realities of the application of interest.[17]
Main advantages of prediction algorithms over transforms ones are support for higher bit rates, low complexity and better performances on average. If quantizer is used, near lossless compression can be achieved. Main problem of this approach is error propagation, due to the recursive filters used to predict pixel values. Moreover also low performance is one of the main drawbacks of prediction techniques, as well as poor error tolerance. One solution can be using hybrid algorithms, which include the employment of different techniques in order to overcome the just explained problems.[13]

3.2 State of the Art

As well as for typical digital image processing, also in hyperspectral imaging compression techniques can be classified as lossless, i.e. reducing data size without losing information, and lossy, which on the contrary lowers data size by discarding less significant information. In lossy techniques how much information can be lost is usually regulated by some parameters that set a threshold on the amount of discarded data. Once motivations for reducing amount of data have been explained, it is fundamental to understand that there is not a universal method that is suited for all scenario. Hence, many solutions have been developed, according to available techniques that can be combined together (e.g. transform coding, adaptive filtering, vector quantization, etc.). This section provides a review on the state of the art of compression algorithms for HSI.
Algorithms reported in Tab.3.3 have been selected from reviews done by Dua et al.[13] and Babu et al.[18].

Table 3.3

Algorithm	**Technique and Results**	**Future research directions**
DWT-TD[2]	Transform (2D-DWT) and Tensor (Tucker Decomposition) based. Better pixel-based results for classification accuracy, with the drawback of computational load due to TD.	Decrease core tensor size to decrease the load.

Table 3.3

Algorithm	**Technique and Results**	**Future research directions**
PCA-DCT[19]	Combination of machine learning and DCT (transform based); PCA is applied to find the features and DCT is used to compress. High CR but selection of the number of features is not considered.	Data decorrelation in spatial domain through different transformation techniques.
IKLT-IDWT[20]	Integer based method. Eigen-matrix decomposition is applied. Invertible integer KLT and integer DWT are cascaded to spatially decorrelate image data. Three different wavelet-based coding are investigated. Ideal energy compaction but not suitable for on-board or real-time compression.	Parallelization to reduce compression time.
Folded PCA[21]	PCA combined with JPEG2000 (to further compression). Covariance matrix is calculated by folding the spectral vector into a matrix and eigenvectors are used to represent features of the entire image. Best compression and classification with number of principal components=40. Not suitable for many applications.	Parallelization.
RLS filter[1]	Statistical and prediction based: least square optimization and entropy coding. Adaptive edge-based prediction algorithm by using correlation among pixels. Worsening of the performances in terms of CR for low number of prediction bands.	Reduce computational time.

Table 3.3

Algorithm	**Technique and Results**	**Future research directions**
CCSDS-123.0-B Enhancement[22]	Enhancement of the standard proposed by the consultative committee for space data systems. Introduction of specific extensions (constant SNR, rate control and hybrid coding). User control over accepted losses with low complexity algorithm. It is application friendly but hybrid encoding could provide worse results in some cases.	Hardware implementation.

Chapter 4

Proposed Algorithms

Once the basis of compression techniques have been explained and the state of the art has been investigated, some algorithms have been studied in detail and then implemented in software to test their performances on a real dataset of hyperspectral images. This chapter explains how they works in depth.

4.1 PNG and JPEG

As already explained, it is possible to see the hyperspectral image as a series of consecutive images through frequency spectrum. As a consequence, the starting point for this survey was using traditional digital image compression standards, specifically PNG and JPEG for reducing size of each of them, and so the one of the hypercube.

Portable Network Graphics (PNG) is an image format, defined in W3C recommendation for image coding. It is also standardized as ISO 15948-2004. PNG is a fully lossless compression method, patent-free, it supports palette based images, i.e. color mapped images, and it provides support for transparency. It is based on prediction and entropy coding, as a combination of LZ77 and Huffman coding, called *deflate* algorithm. Prediction is applied first, transforming data through filtering. There are five types of filters, collectively referred to as "filter method 0". The simplest filter is the *None* filter, that implements the trivial case of no filtering; the second one is the *Sub* followed by *Up* filter, which are conceptually similar. Both are differencing filters, Sub operates on the corresponding byte of the pixel to the left, while Up on the above, i.e. on the corresponding byte of the pixel on the previous row of the image. Problems using this two filters are avoided filling with zero bytes the region outside the image. The fourth filter type is *Average*, which is a combination of the previous two. Last type is called *Paeth* filter and it is the

most complex one. Its basic idea is to make an initial estimate of the current byte based on a linear function of the pixel of the above, left and upper-left positions with respect to the current one.[23]
The compression engine, instead is deflate. It is comparable to or even faster than LZW in encoding as well as in decoding. Specifically, it uses a sliding window of up to 32 KB with a Huffman encoder on the back end. Encoding procedure relies on the longest matching string in the 32-KB window immediately prior to the current position, storing that as a pointer and length, incrementing the current position and consequently the window. Deflate limits the maximum match length to be between 3 and 258 bytes and, as a consequence, the maximum achievable compression ratio is 1032:1 (using 1 bit to encode the distance pointer and one bit to encode the maximum length, i.e. 258 bytes).[23]
Joint Photographic Experts Group (JPEG) is a ISO/ITU joint committee that defined the first standard for continuous-tone images. The standard specifies the decoding process, while the encoding has not a mandatory procedure to follow, allowing competition among different implementers. However, non-normative guidelines about the compression process are provided in the standard, including extensions for progressive and hierarchical coding.[14][24]

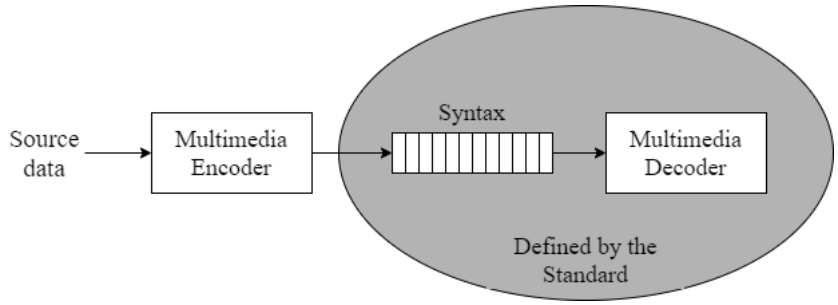


Figure 4.1: Standardization in JPEG.

JPEG comes to the need of increasing compression ratio and it is a lossy technique. It was developed for digital images and primarily for digital photography. This standardized image compression scheme, in fact, works on both full-color or grayscale images, and foresees also a separate lossless mode. In Fig.4.2 is depicted the basic steps of JPEG standard. Indeed, this standard, relies on four basic passages: color space decomposition, application of 2D-DCT (see Chapter 3), quantization and entropy coding. Moreover, JPEG is a symmetric codec, in the sense that the steps needed for decoding are the exact inverse of the ones used for encoding procedure.[14][24]

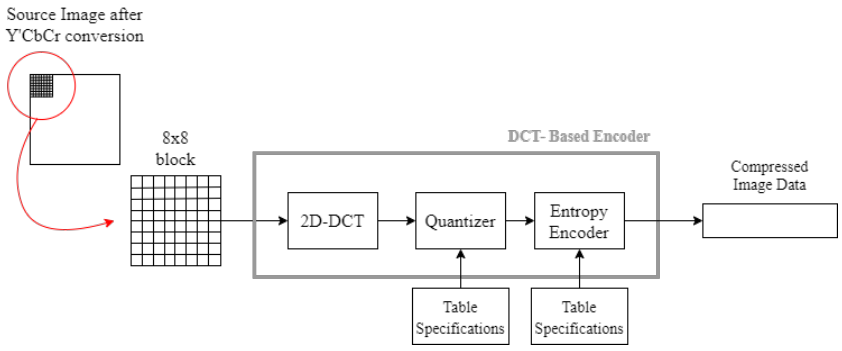


Figure 4.2: JPEG Encoder Block Diagram.

Unlike PNG, JPEG does not use the palette color system (red, green and blue - RGB), but the Y'CbCr, which is very similar to YUV. The difference between the two is an offset, introduced in order to have always positive values, since the domain of interest is digital. YUV system can be obtained from RGB one by means of a linear transform: Y is the luminance, which contains most of the information, i.e. the one better perceived by the human visual system. It contains the corresponding grayscale image. U and V, instead, are the chrominance components, which add color information to the luminance one. At this step, some information can be lost, due to subsampling of the chrominance components (see Fig.4.3). In fact, it can be chosen 4:4:4 (no subsampling), 4:2:2 and 4:1:1 or 4:2:0 coding (Cb and Cr subsampled by 2), depending on the application requirements. However, in order to avoid aliasing, most of the time it is necessary to filter the image before subsampling.[14][24]

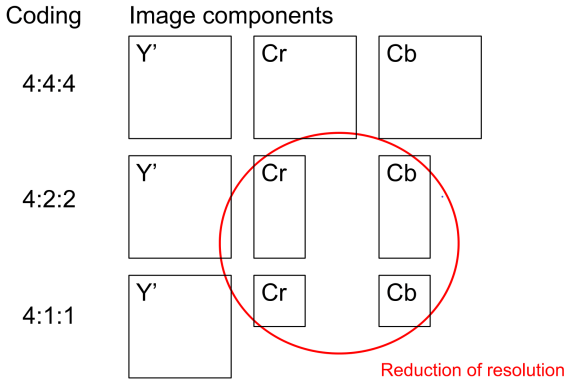


Figure 4.3: Y'CbCr Subsampling.[24]

Next step is dividing the image in blocks of 8×8 pixels and applying the two dimensional discrete cosine transform to them. This passage decorrelates well the

coefficients, avoiding the need of prediction. Then, a weighted scalar quantization is applied to each transformed coefficient in every block. Usually, quantization parameter is chosen to adjust the compression ratio: the lower it is, the less will be the significant coefficients to code; on the other hand, lowering the quantization parameter will worsening the image quality, so that it is a trade-off between how much compression is achieved and the quality of the compressed image. At this stage less important details are discarded and, in order to reduce the number of bits to code the data, other techniques are used. Final step is entropy coding: first of all coefficients are re-ordered from 2D to 1D in a *zig-zag* fashion, leading to many sequences of consecutive zeros. The resulting scanned coefficients are then encoded with Run Length Encoding (RLE), as sequence of "couples" of symbols, i.e. Symbol 1 = (RUN LENGTH, SIZE) containing the number of zero samples preceding the current sample and the number of quantization bits for the current sample, and Symbol 2 = (AMPLITUDE), which is the quantized sample value. Then, usually Huffman coding is applied, even if JPEG standard allows also the use of arithmetic encoder, but it is rarely used due to royalties problems and it is slower with respect to Huffman coding, and ,even if it is more efficient mathematically speaking, it achieve poor advantage on CR compared to Huffman.[14][24]
After JPEG, JPEG-2000 was developed. The main difference between the two is the type of transform: instead of the two dimensional DCT on 8×8 blocks, in fact, JPEG-2000 uses the 2D-DWT on larger blocks (64×64). Its complexity is an order of magnitude higher than the JPEG and there are many additional features with respect to previous standard, such as different quality for different areas (region of interest-ROI) and re-synchronization codes for transmission over noisy channels. Other evolution of first JPEG standard are available, as JPEG-XR, which allows to achieve better compression, JPEG-LS (lossless/near-lossless) and JPEG-XT, which adds bit depth and high dynamic range.[24]
Focusing on JPEG-LS, it provides both lossless and near-lossless modes, the difference between the two consist in the value assigned to the NEAR parameter, that is the maximum absolute pixel error tolerated in the near lossless; if NEAR=0, then lossless mode is selected. In Fig.4.4 is presented the block diagram of JPEG-LS encoding, made up by context modeling, mode determination (run or regular), context based prediction, with a context-dependent error correction term, error quantization and entropy coding.[23]

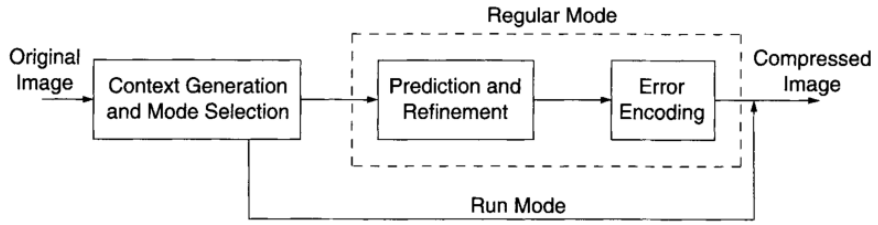


Figure 4.4: JPEG-LS Block Diagram.

In the first block, context modeling, a set gradients is determined from reconstructed pixels adjacent to the current one: if all the local differences are zero (lossless compression) or less than NEAR value (near-lossless compression), then it is selected *run mode*, otherwise it is used the *regular mode*. If it is the latter case, an initial prediction is made based on an edge detecting predictor, which uses the values of the neighbour reconstructed pixels. Then, this rough prediction is refined in order to decrease the error bias associated with a given context. Next step consist of mapping and quantizing the output error. Eventually, entropy coding is applied.[23]

4.2 Recursive Least Square Filter for HSI Compression

Jinwei Song et al.[1] developed an algorithm capable of compressing lossless hyperspectral images with the use of RLS filter.
As already explained, HSIs have strong correlation on both spatial and spectral dimension. First step of the algorithm consist of calculating the average value of four neighbour $s_z^{NW}(t)$, $s_z^N(t)$, $s_z^{NE}(t)$ and $s_z^W(t)$ of the current one $s_z(t)$, i.e. calculating the *local mean* $\tilde{s}_z(t)$ as defined in 4.1 (see Fig.4.5).

$$\tilde{s}_z(t) = \left(s_z^{NW}(t) + s_z^N(t) + s_z^{NE}(t) + s_z^W(t)\right)/4 \tag{4.1}$$

In particular, $t = Wy+x$, with $W \times H$ size of the current band image (width×height) and (x, y) coordinates of the current pixel.[1]

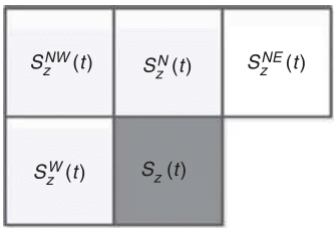


Figure 4.5: Neighbour pixels used for calculating local mean.[1]

Next step, is subtracting the calculated local mean by the current pixel, in order to eliminate correlation in the current band image. The result is the local difference, defined as $d_z(t) = s_z(t) - \tilde{s}_z(t)$. For the first band $d_z(t)$ is sent to the arithmetic encoder directly, while correlation in the other bands is eliminated by using a RLS filter. The adopted encoder is the adaptive arithmetic one (AAC).[1] How the algorithm works step by step is explained in Tab.4.1, where the expectation signal of the RLS filter is $d_z(t)$ and the input vector is composed as $\mathbf{d}_z(t) = [d_{z-1}(t), d_{z-2}(t), ..., d_{z-p}(t)]$, with p is the number of prediction bands used to predict the current band. The weight vector of the filter is $\mathbf{w}(t) = [w_1(t), w_2(t), ..., w_p(t)]$.[1] This procedure keeps the complexity relatively low and it is suitable for real time compression on satellites.[1]

Proposed Algorithm

STEP 1: Let $\mathbf{P}(0) = \delta \mathbf{I}_p$, the weight vector $\mathbf{w}(0) = [0]$ and $t = 1$, where $\delta = 1e - 4$ and $\mathbf{I}_p$ identity matrix of order p.

STEP 2: Calculate $d_z(t)$ and $d_{z-i}(t)$ with $i = 1, ..., p$. Form the input vector as $\mathbf{d}_z(t) = [d_{z-1}(t), d_{z-2}(t), ..., d_{z-p}(t)]$.

STEP 3: Calculate the prediction residual $e_z(t)$ as:

$$e_z(t) = d_z(t) - \lfloor \mathbf{d}_z(t)\mathbf{w}^T(t-1) \rfloor \tag{4.2}$$

STEP 4: Let:

$$\mathbf{k}^T(t) = \frac{\mathbf{P}(t-1)\mathbf{d}_z^T(t)}{1 + \mathbf{d}_z(t)\mathbf{P}(t-1)\mathbf{d}_z^T(t)} \tag{4.3}$$

and

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{k}^T(t)\mathbf{d}_z(t)\mathbf{P}(t-1) \tag{4.4}$$

STEP 5: Update $\mathbf{w}(t)$ by

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{k}(t)e_z(t) \tag{4.5}$$

STEP 6: Send $e_z(t)$ to the encoder (AAC).

STEP 7: Increment t, i.e. $t = t + 1$

STEP 8: If t is less or equal than $W \times H$, *GO TO STEP 2*, otherwise end prediction procedure of current band prediction.

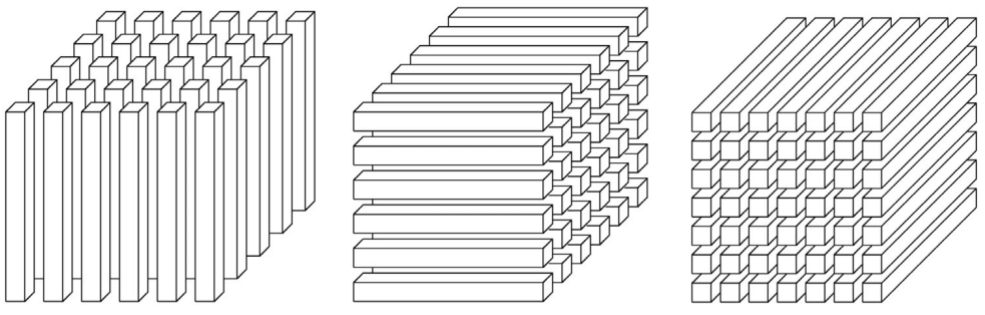
Table 4.1

4.3 Discrete Wavelet Transform and Tucker Decomposition Algorithm

Last investigated HSI compression technique is proposed by Karami et al. in [2]. It relies on 2D-DWT and Tucker Decomposition, since data are represented in tensor notation. In order to better understand how it works a small digression on tensor theory is needed.

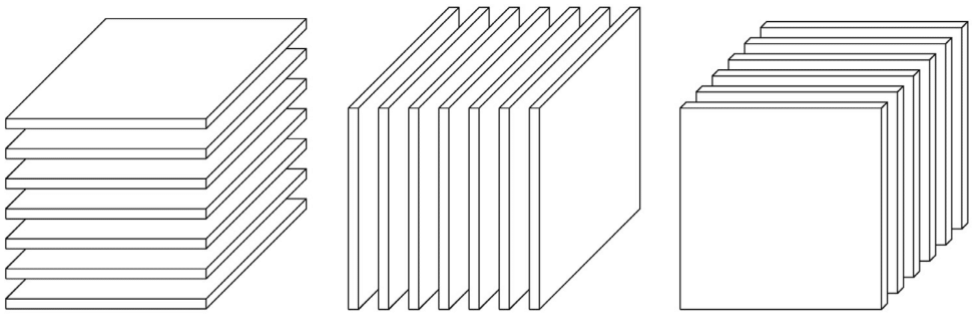
4.3.1 Tensors and Tensor Notation

In mathematics, a *tensor* is a multidimensional array. The *order*, or *way* or *mode* is the number of dimensions. Tensors of order one are vectors and are denoted by boldface lowercase letters, e.g. $\mathbf{x}$ and the ith entry of a vector $\mathbf{x}$ is denoted by a_i; tensors of order 2 are matrices and are denoted by boldface capital letter, e.g. $\mathbf{X}$ and the element (i, j) of $\mathbf{X}$ is denoted by x_{ij}; tensors with order grater or equal to 3 are *higher-order* tensors and are denoted by boldface Euler script letters, e.g. $\boldsymbol{\mathcal{X}}$, and an element (i, j, k) of a third-order tensor $\boldsymbol{\mathcal{X}}$ is denoted by x_{ijk}. The nth element in a sequence is denoted by a superscript in a parentheses, e.g. $\mathbf{A}^{(n)}$ denotes the nth matrix in a sequence. It is possible to fix a subset of indices and form a subarray, for example in matrices with notation $\mathbf{x}_{i,:}$ it is denoted the ith row of matrix $\mathbf{A}$. If notation is compact, e.g. $\mathbf{x}_j$, it is usually referred to as the jth column of matrix $\mathbf{A}$.[25]
Fibers are the higher-order analogue rows and columns, and it is defined by fixing every index but one. *Slices*, instead, are two-dimensional section of a tensor, defined by fixing all but two indices. In Fig.4.6 and Fig.4.7 are presented fibers $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$ and $\mathbf{x}_{ij:}$, and slices $\mathbf{X}_{:jk}$, $\mathbf{X}_{i:k}$ and $\mathbf{X}_{ij:}$ of a third order tensor $\boldsymbol{\mathcal{X}}$.[25]



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure 4.6: Fibers of third-order tensor.[25]



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Figure 4.7: Slices of third-order tensor.[25]

All the subsequent operations are defined according to [25].
The *norm* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is defined as the square root of the sum of all its elements and it is defined in (4.6). It corresponds to the Forbenius norm of a matrix.

$$\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2} \tag{4.6}$$

Another important operation to be defined is the *inner product*: given two tensor with the same size $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ it is the sum of the products of their entries, in formulas:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N} \tag{4.7}$$

The *n-mode product*, instead, is the product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ and it is denoted by $\mathcal{X} \times_n \mathbf{U}$. It is defined as in 4.8.

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_N} x_{i_1 i_2 \ldots i_N} u_{j i_n} \tag{4.8}$$

The *Kronecker product* is defined between two matrices $\mathbf{U} \in \mathbb{R}^{I \times J}$ and $\mathbf{V} \in \mathbb{R}^{K \times L}$, it is equal to:

$$\mathbf{U} \otimes \mathbf{V} = [\mathbf{u}_1 \otimes \mathbf{v}_1 \quad \mathbf{u}_1 \otimes \mathbf{v}_2 \quad \mathbf{u}_1 \otimes \mathbf{v}_3 \ \ldots \ \mathbf{u}_J \otimes \mathbf{v}_{L-1} \quad \mathbf{u}_J \otimes \mathbf{v}_{L-1}] \tag{4.9}$$

Other two products defined in tensor notation are the *Khatri-Rao product* and the *Hadamard product*. The first one is the equivalent columnwise of the Kronecker product, denoted with $\odot$. Notice that in the case of vectors Kronecker and Khatri-Rao products coincide. The Hadamard product, instead is the elementwise matrix product and it is possible for matrices $\mathbf{U}$ and $\mathbf{V}$ that have the same size. It is

denoted by $\mathbf{U} * \mathbf{V}$.[25]

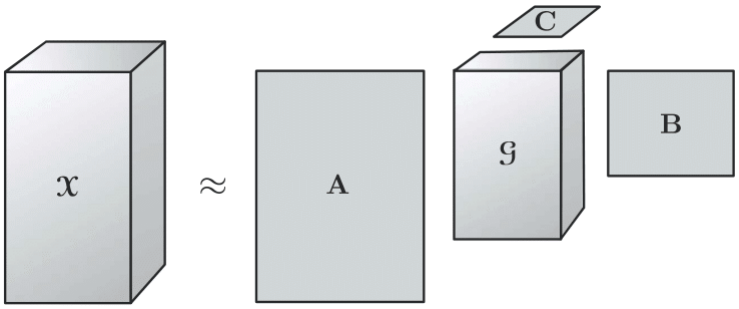


Figure 4.8: Tucker decomposition of a three-way array.[25]

It is possible to decompose a tensor according to two main methods: the CANDECOMP/PARAFAC, better known as ***CP***, which factorizes a tensor into a finite sum of component rank-one tensors, and the ***Tucker*** decomposition, which is a form of higher-order PCA. Tucker decomposition decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode, so that, for example, a three-way tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be written as:

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r = [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{4.10}$$

where, $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices, usually orthogonal, while $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is the so-called *core tensor*, which ca be seen as a compressed version of $\mathcal{X}$. Tucker tensor decomposition is illustrated in Fig.4.8. Most of the time factor matrices are columnwise orthonormal. There are three main methods for computing the tucker decomposition of a tensor: higher-order SVD (HOSVD), Alternating Least Squares (ALS) and higher-order orthogonal iteration (HOOI).[25]

4.3.2 Algorithm

In their work [2], Karami et al. relies on the fact that some compression methods consider HSIs as three-dimensional data, so that they could be presented as three-dimensional tensors, with two dimensions for spatial information and one for spectral. By doing so, spatial and spectral correlation of a HSI are taken into consideration simultaneously, and not alternatively as in the case of other compression techniques. The proposed algorithm is an hybrid scheme based on

DWT and Tucker Decomposition (TD).[2]
In the first step two dimensional discrete Wavelet transform is applied to each band of the HSIs, so that four sub-band images are obtained: the ones produced by the high-pass filtering of the discrete time domain, i.e. horizontal (H), vertical (V) and diagonal (D) information, and the approximate information (A) produced with the low pass filtering. The 2DWT of function $f(x, y)$ with size I_1 and I_2 can be shown as:

$$W_\phi(i_1, i_2) = \frac{1}{\sqrt{I_1 I_2}} \sum_{y=0}^{I_2-1} \sum_{x=0}^{I_1-1} f(x, y)\phi_{i_1,i_2}(x, y) \tag{4.11}$$

$$W_\psi^k(j, i_1, i_2) = \frac{1}{\sqrt{I_1 I_2}} \sum_{y=0}^{I_2-1} \sum_{x=0}^{I_1-1} f(x, y)\psi_{j,i_1,i_2}^k(x, y) \tag{4.12}$$

$$k = \{H, D, V\} \tag{4.13}$$

$$\begin{aligned}\phi_{i_1,i_2}(x, y) = \phi(x - i_1, y - i_2) = \sum_m h_\phi(m - 2i_1)\sqrt{2}\phi(2x - m) \\ \times \sum_n h_\phi(n - 2i_1)\sqrt{2}\phi(2x - n)\end{aligned} \tag{4.14}$$

$$\begin{aligned}\psi_{j,i_1,i_2}^H(x, y) = 2^{\frac{j}{2}}\psi(2^j x - i_1, 2^j y - i_2) = 2^{\frac{j}{2}} \sum_m h_\psi(m - 2i_1)\sqrt{2}\psi(2^{j+}x - m) \\ \times \sum_m h_\psi(n - 2i_1)\sqrt{2}\psi(2^{j+}x - n)\end{aligned} \tag{4.15}$$

h_ϕ and h_ψ are the (9/7) biorthogonal wavelet filters whose coefficients are reported in Tab.4.2, ϕ is the scaling function, $W_\phi(i_1, i_2)$ coefficients define an approximation of $f(x, y)$ and $W_\psi(j, i_1, i_2)$ instead add horizontal, vertical and diagonal details. Usually, $i_1 = i_2 = 2^j$ with j = 0,1..., $J - 1$.[2]

n	$h_\phi[n]$	$h_\psi[n]$
0	0.7885	0.8527
±1	0.4181	0.3774
±2	-0.04069	-0.111
±3	-0.06454	-0.02385
±4	-	0.03783

Table 4.2

The inverse 2DWT is formulated as in 4.16

$$\begin{aligned} f(x,y) &= \frac{1}{\sqrt{I_1 I_2}} \sum_{i_2} \sum_{i_1} W_\phi(i_1,i_2)\phi_{i_1,i_2}(x,y) \\ &+ \frac{1}{\sqrt{I_1 I_2}} \sum_{k=\{H,D,V\}} \sum_{j=0}^{J-1} \sum_{i_2} \sum_{i_1} W_\psi^k(i_1,i_2)\psi_{j,i_1,i_2}^k(x,y) \end{aligned} \tag{4.16}$$

In lossy compression the detailed sub-bands, i.e. H,V and D, are ignored and the preserved information is only the detailed one (A). By doing so, in fact, higher compression ratio is achieved.[2]
In the second step, instead, the TD algorithm is applies to the four (if lossless case) results of the 2DWT. For each of the four tensors the size of the core tensor in the Tucker decomposition is selected manually and, considering that most of the energy is contained in the lowest frequency components (tensor A), its dimensions are higher then those of other tensors (H,V and D).[2]
In the third step the four core tensors and the corresponding factor matrices have to be transmitted. Since most of the elements of the core tensors are nearly zero, the bit-plane coding procedure is used to transmit them. This procedure is composed by two passes: the significant pass an the refinement. The first one identifies a significant element according to a certain threshold value among the absolute values of the core elements; then, the significant elements have to be encoded and preserved for the next bitplane procedure iteration. Refinement, instead, consist of dividing in half the threshold and repeat the process. This approach is repeated until the energy of the encoded elements is equal or higher than 99.5% of that of the original tensor.[2]
Last step is encoding: there are many possible approaches, one of the most used for lossless entropy coding is using the adaptive arithmetic encoder (AAC). Unlike Huffman encoding, it does not require the transmission of a codebook, so that it achieves higher compression. AAC computes the cumulative distribution function (CDF) of the probability of a sequence of symbols and the result, i.e. a numerical value, is represented in binary code. To reconstruct the original HSI, beside the core tensor, also the factor matrices are needed. They are normalized so that their elements are in the range [0,1] and then uniform quantization is used to transfer the 12 factor matrices (in the case of lossless algorithm implementation).[2]
In fifth step the transmitted data are decoded and in the sixth one the inverse 2DWT is applied to reconstruct the images.[2]
In Tab.4.3 and Tab.4.4 it is summarized the just explained algorithm.[2]

Proposed Algorithm
Input: Original HSIs $\underline{\mathbf{X}}$ with size $I_1 \times I_2 \times I_3$
STEP 1: Apply the 2DWT to each spectral band to obtain 4 sub-images, i.e. approximate, diagonal, vertical and horizontal tensors.
STEP 2: Apply Tucker Decomposition algorithm to the four tensors individually. Each tensor has size $(\frac{I_1}{2} \times \frac{I_2}{2} \times I_3)$.

TD
Input: -Tensor $\underline{\mathbf{Y}}$ of size $(\frac{I_1}{2} \times \frac{I_2}{2} \times I_3)$. -1$\leq J_1 \leq \frac{I_1}{2}$, 1$\leq J_2 \leq \frac{I_2}{2}$, 1$\leq J_3 \leq \frac{I_3}{2}$ ***Output:*** 3 factors $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ and core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$
STEP 3: Bit-plane coding procedure repeated until a stopping criterion is met.
STEP 4: Quantize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ and encode core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ using AAC.

Table 4.3

Reconstructed Algorithm
STEP 5: Decode the elements $\underline{\hat{\mathbf{Y}}} = \underline{\hat{\mathbf{G}}} \times \{\underline{\hat{\mathbf{A}}}\}$
STEP 6: Calculate the inverse 2DWT to reconstruct the images $\underline{\hat{\mathbf{X}}}$

Table 4.4

Considering that the chosen wavelet is the (9/7) wavelet with one level of decomposition and being I_A the average number of pixels of the approximate core tensor, and J_A the average of the dimensions $(\frac{I_1}{2} \times \frac{I_2}{2} \times I_3)$ of the approximate core tensor, the total computational complexity of the proposed algorithm is of order $4 \times O(J_A I_A^3) + O(9 I_1 I_2 I_3/7)$.[2]

Chapter 5

Experimental Results

This chapter illustrates the implementation and performances of each investigation proposed in the previous chapter, making a discussion among comparable results. Numerical results are presented and effects of compression on spectral signatures are examined.

5.1 Data Set and Implementation Choices

Proposed experiments aim to reduce the size of the original hyperspectral image without affecting spectral signatures. In particular, tests have been done on four hyperspectral images on which ice detection on metallic surfaces has to be performed. Specification on the data set can be found in Tab.5.1. Each file of the proposed data set has size 389 [MB], saved with ENVI image format.

File	**Presence Of Ice**	**Indoor/Outdoor**
160504-SG_30ms_100_RC.dat	no ice	outdoor
161337-G_30ms_100_RC.dat	ice	outdoor
noice_Luce_1.dat	no ice	indoor
ice_Luce_5.dat	ice	indoor

Table 5.1

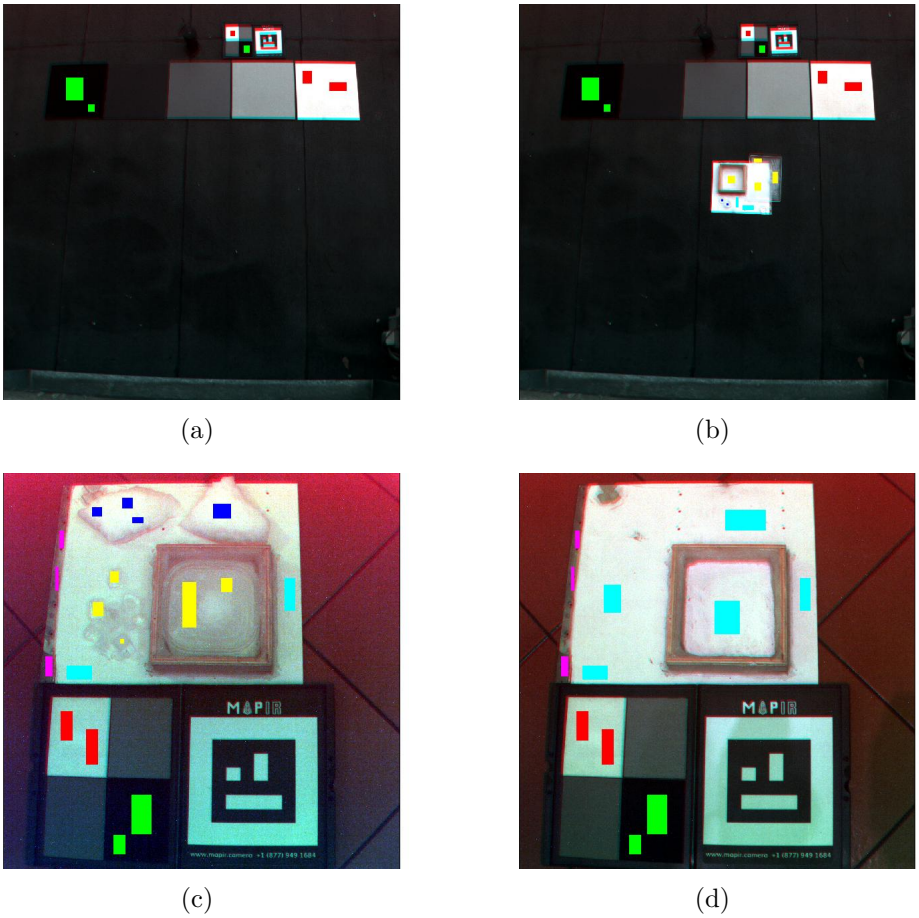
Figure 5.1: Senop Rikola Hyperspectral Camera.

Data have been acquired with Senop Rikola hyperspectral camera (Fig.5.1) whose sensor specifications are reported in Tab.5.2.

Specifics	**Senop Rikola**
Lens Optics	H 36.5°, V 36.5°
Spectral Range	500-900 nm
Spectral Channels	380
Spectral Resolution	1 nm
Shutter Type	Global
Focal Length	9 mm
Image Resolution	1010× 1010 pixels
Pixel Size	5.5μm
Weight	720 g
Dimensions	172.7×89×77 mm
Cost	≈60 000 €

Table 5.2

Examined data, specifically *161337-G_30ms_100_RC.dat* and *ice_Luce_5.dat*, include two types of ice: white and transparent, the first similar to rime and the second similar to clear ice. Thus, differentiation in testing spectral signatures must be done, due to spectral variability of solid materials (see Chapter2).
All four images are composed by 100 bands covering the spectral range form 505 to 903 nm with a theoretical band step of 4 nm. Each image resolution is 1010×1010 pixels. Data used are radiometrically calibrated through Empirical Line Calibration tool of ENVI Software[26].



(a) (b)

(c) (d)

Figure 5.2: Original Hyperspectral Images ROIs.

In Fig.5.2 are shown the four tested images described in Tab.5.1. Selection of regions of interest have been done manually using ENVI 5.2 [27], ROIs' legend can be found by looking at Tab.5.3

Clear Ice
Rime Ice
Metal
Lacquered Metal
White Reference Panel
Black Reference Panel

Table 5.3: ROIs' Legend

In order to validate the proposed implemented algorithms, tests have been done

on larger data set of images, the ones described in Tab.5.1 contains one of each. Compression techniques have been implemented using MATLAB and its Hyperspectral Image Toolbox[28].

5.2 PNG and JPEG

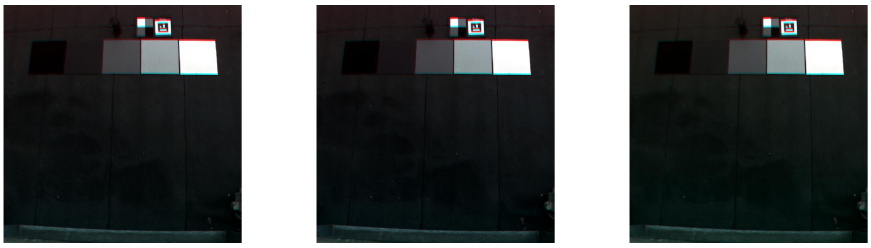
First attempt consist of applying digital image compression standards to each per-band image of the hyperspectral data cube. This has been done by extracting from the input file one image of 1010×1010 pixel for each of the 100 bands, which is then saved as png or jpeg file. According to original values, 16 bits are needed to represent properly original information. The result is a collection of standard per-band bi-dimensional images. For each folder containing per-band images of the same hyperspectral one, there is a metadata file, since information (e.g. wavelengths) are necessary for the inverse procedure, and so for the hypercube reconstruction.
For what concerns lossless procedures, to reduce noise, and so increasing the compression ratio, a bi-dimensional Wiener filter has been used. It filters the per-band image using a pixel-wise adaptive low pass Wiener filter. It acts on a neighborhood of size m×n to estimate the local image mean and standard deviation. A good compromise between execution time and performance is using one of size 3×3. The additive noise power is assumed to be Gaussian.
The objective of these experiments is to achieve a reasonable compression ratio without disrupting spectral signatures. In Tab.5.4 are reported numerical results for lossless compression of the four hyperspectral images described in previous section.

FILE	**PNG**		**JPEG**	
	Noisy	**Filtered**	**Noisy**	**Filtered**
160504-SG_30ms_100_RC.dat	35.4 MB	19.3 MB	29.5 MB	18.1 MB
161337-G_30ms_100_RC.dat	36 MB	20 MB	29.7 MB	18.5 MB
noice_Luce_1.dat	77.9 MB	51.6 MB	64 MB	43.3 MB
ice_Luce_5.dat	87.6 MB	59.9 MB	71.8 MB	50.3 MB

Table 5.4

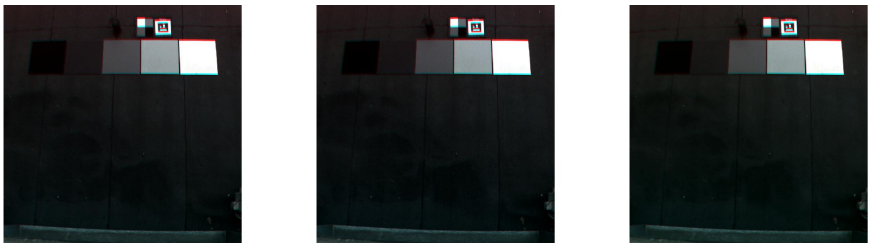
Results show that if Wiener filtering is not applied, and consequently the per-band image is noisy, JPEG in its lossless implementation performs 21% better then PNG. If, instead, noise is filtered out from each per-band image, on average, there is 13% of achievement. However, improvement obtained through noise filtering depends on data and spatial correlation among pixels. In fact, for outdoor hyperspectral images that have similar spatial content (160504-SG_30ms_100_RC.dat

and 161337-G_30ms_100_RC.dat), size of compressed data with JPEG is only 7% better than PNG, while for the indoor (noice_Luce_1.dat and ice_Luce_5.dat) it is about 19%.



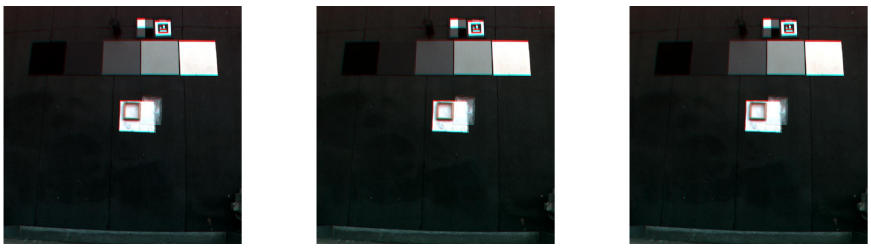
(a) Original (b) Noisy PNG (c) Filtered PNG

Figure 5.3: 160504-SG_30ms_RC.dat: PNG.



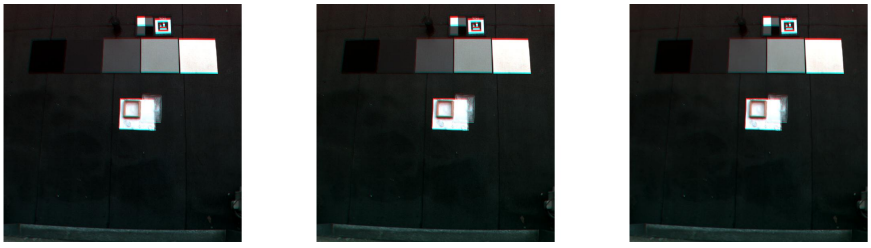
(a) Original (b) Noisy JPEG (c) Filtered JPEG

Figure 5.4: 160504-SG_30ms_RC.dat: JPEG.



(a) Original (b) Noisy PNG (c) Filtered PNG

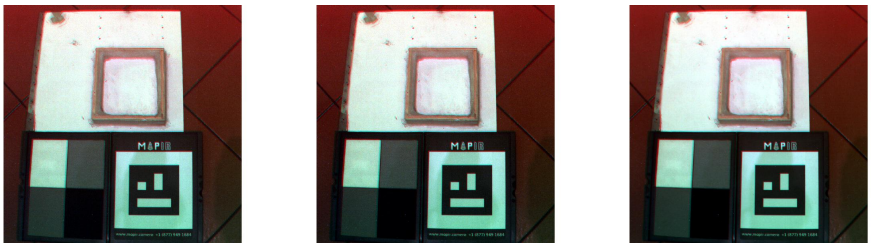
Figure 5.5: 161337-G_30ms_100_RC.dat: PNG.



(a) Original (b) Noisy JPEG (c) Filtered JPEG

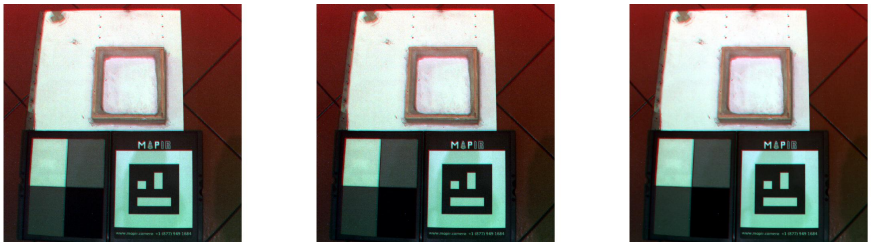
Figure 5.6: 161337-G_30ms_100_RC.dat: JPEG.

From Fig.5.3 to Fig.5.10 are displayed spatial content of each new reconstructed hypercube. There are no perceptual differences between the original and the two lossless techniques, PNG and JPEG, both with and without noise filtering.



(a) Original (b) Noisy PNG (c) Filtered PNG

Figure 5.7: noice_Luce_1.dat: PNG.

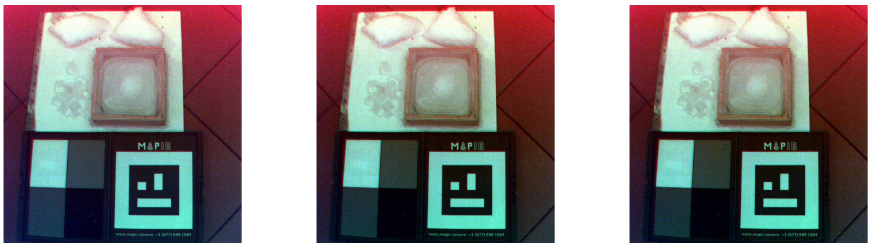


(a) Original (b) Noisy JPEG (c) Filtered JPEG

Figure 5.8: noice_Luce_1.dat: JPEG.

(a) Original (b) Noisy PNG (c) Filtered PNG

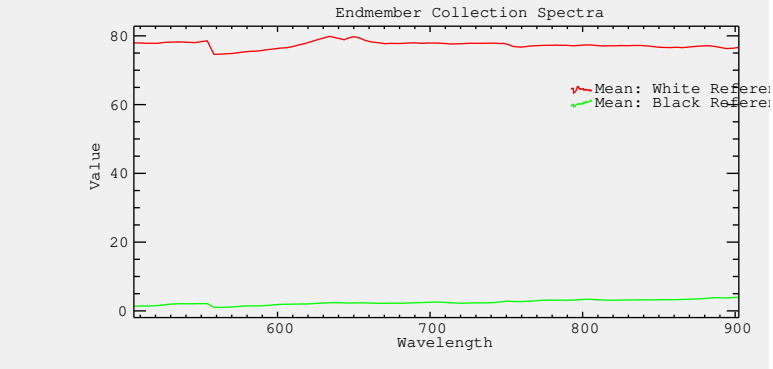
Figure 5.9: ice_Luce_5.dat: PNG.



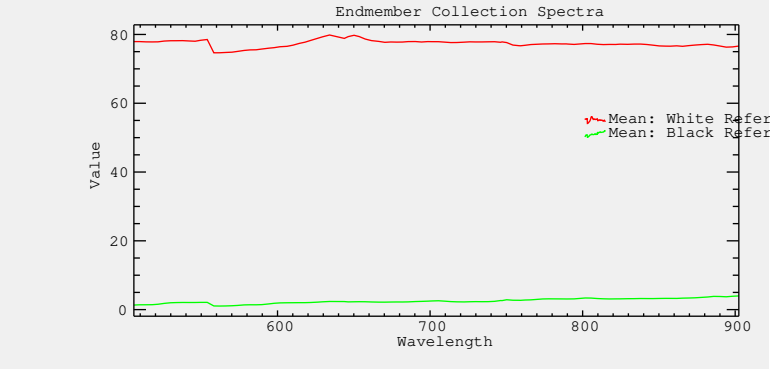
(a) Original (b) Noisy JPEG (c) Filtered JPEG

Figure 5.10: ice_Luce_5.dat: JPEG.

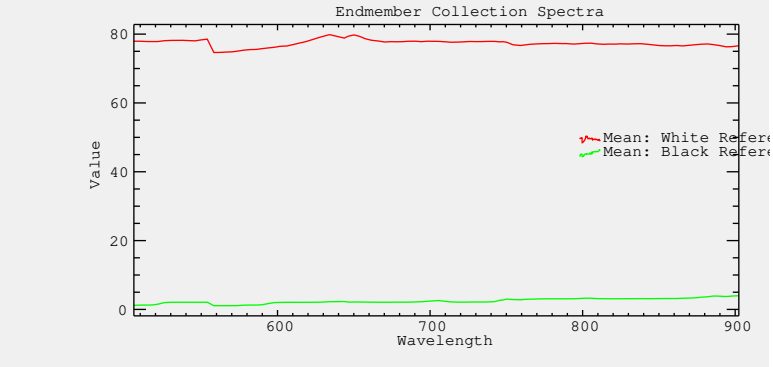
The spectrum of any pixel of these HSIs is a mixture, and the objective of spectral analysis is to extract the spectra of individual materials. In fact, a ground resolution element contains several materials and all of them contribute to the individual pixel spectrum measured by the sensor. As result there is a composite or mixed spectrum, and the target is to identify the "pure" spectra that contribute to this mixture. Such pure spectra are called *endmembers*.[3] Consequently, another important aspect to be analysed is how these manipulations affect endmembers extraction. For hyperspectral analysis it is fundamental to not disrupt spectral signature of reconstructed hypercubes, otherwise classification and other steps could not be possible. Endmember extraction has be done using the same ROIs defined in Fig.5.2 (see Tab.5.3 for legend) with ENVI Endmember Collection Tool [29].
From Fig.5.11 to Fig.5.18 it can be observed how endmembers are preserved from considerable distortion for all hypercubes of the data set. Their extraction leads to almost same results both in case of PNG and lossless JPEG. Specifically, minor changes are present in the mean of Black Reference Panel ROI in indoor file with and without ice (Fig.5.15 to 5.18).



(a) Original

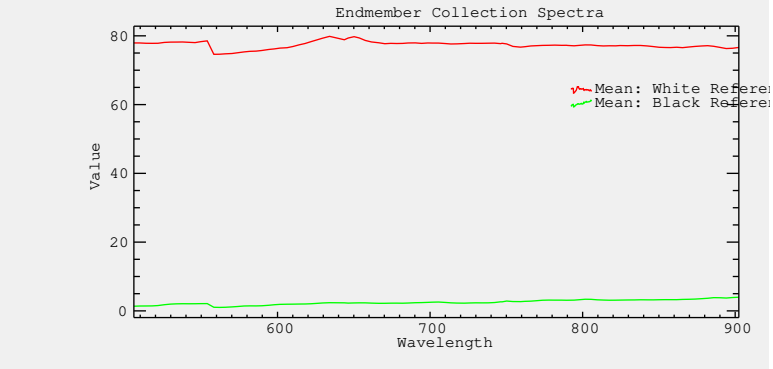


(b) Noisy PNG

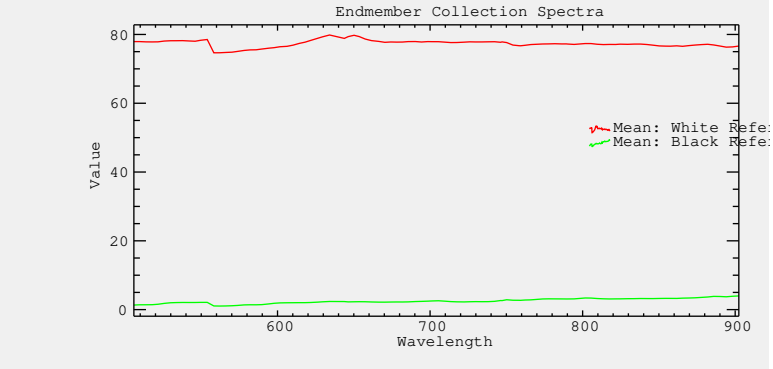


(c) Filtered PNG

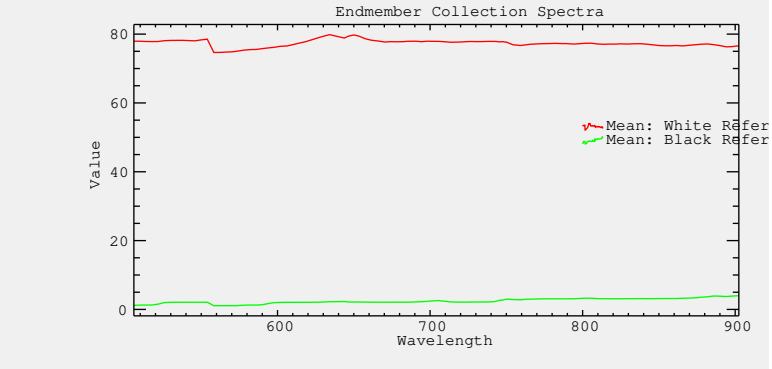
Figure 5.11: 160504-SG_30ms_RC.dat: PNG.



(a) Original

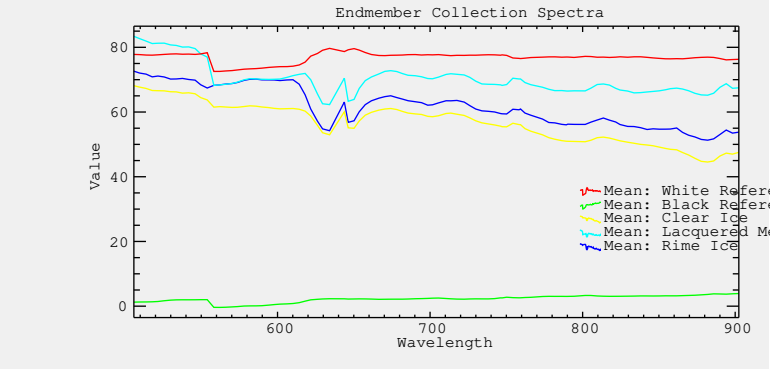


(b) Noisy JPEG

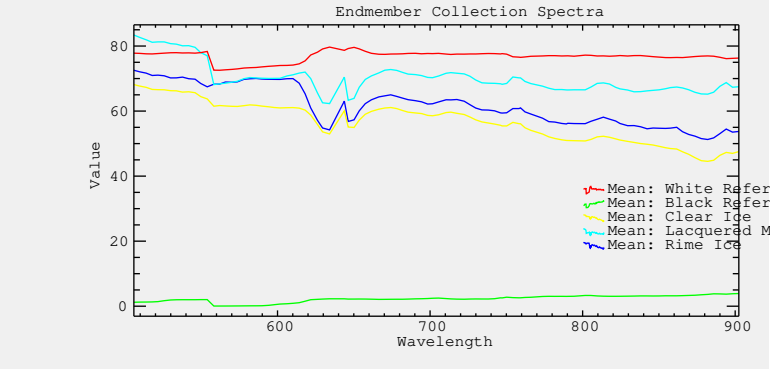


(c) Filtered JPEG

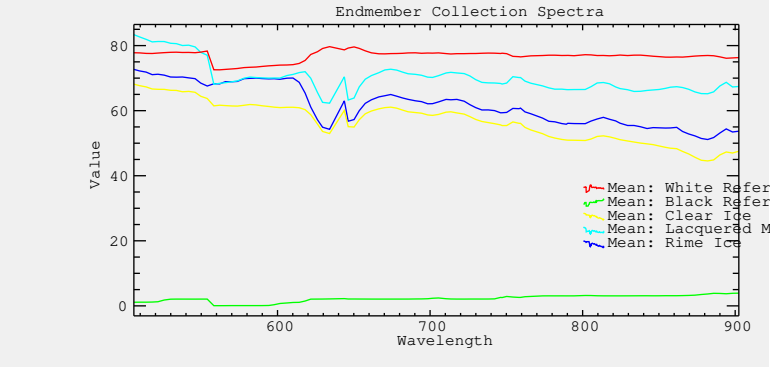
Figure 5.12: 160504-SG_30ms_RC.dat: JPEG.



(a) Original

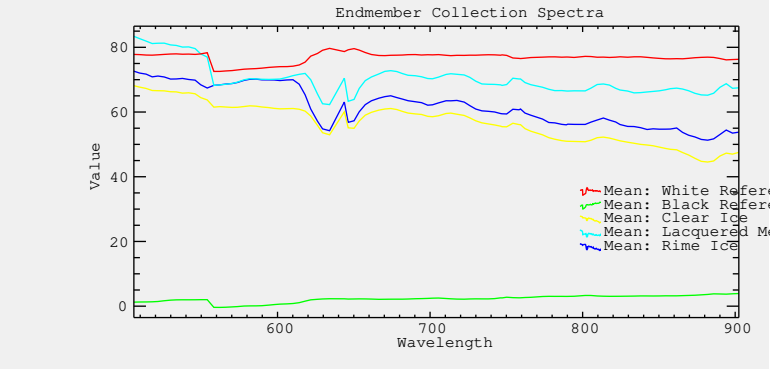


(b) Noisy PNG

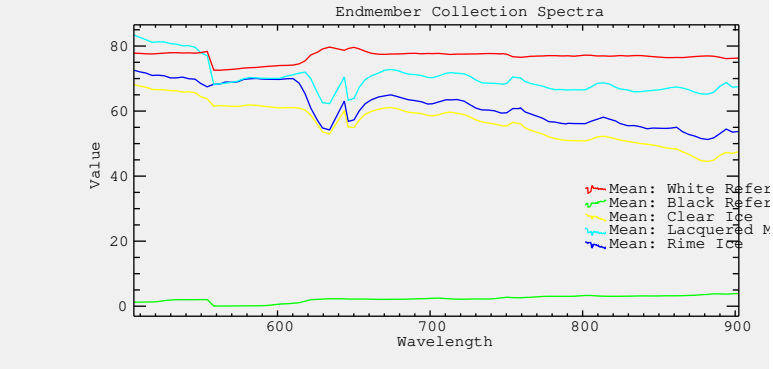


(c) Filtered PNG

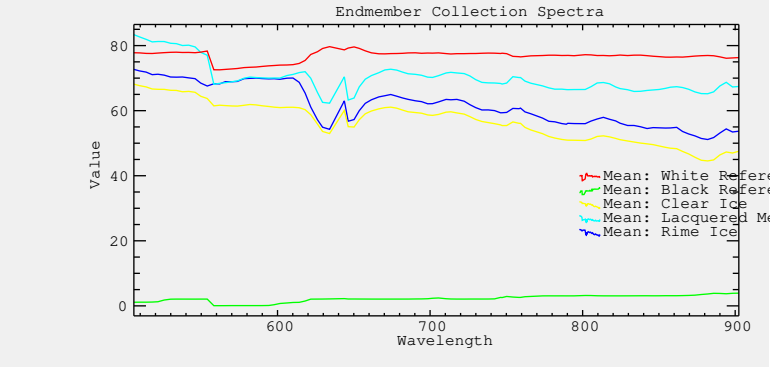
Figure 5.13: 161337-G_30ms_100_RC.dat: PNG.



(a) Original

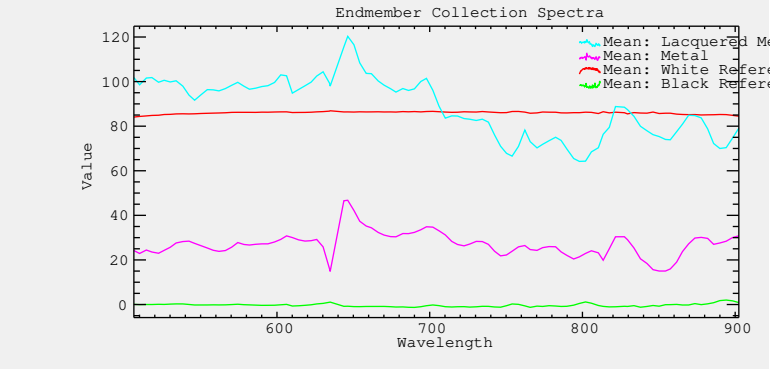


(b) Noisy JPEG

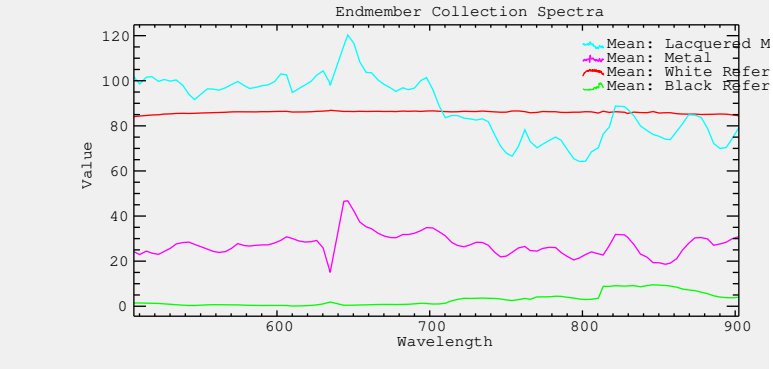


(c) Filtered JPEG

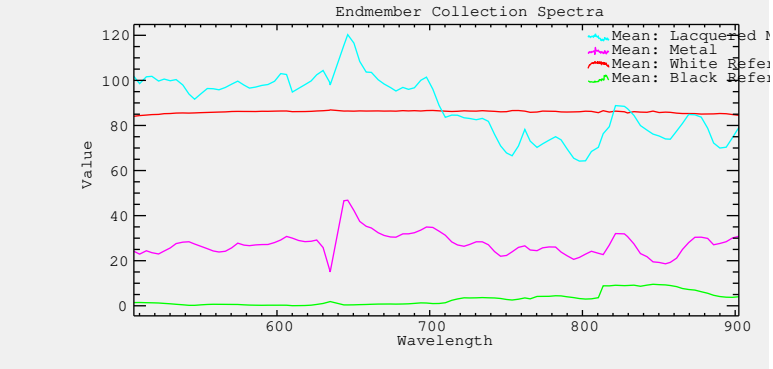
Figure 5.14: 161337-G_30ms_100_RC.dat: JPEG.



(a) Original



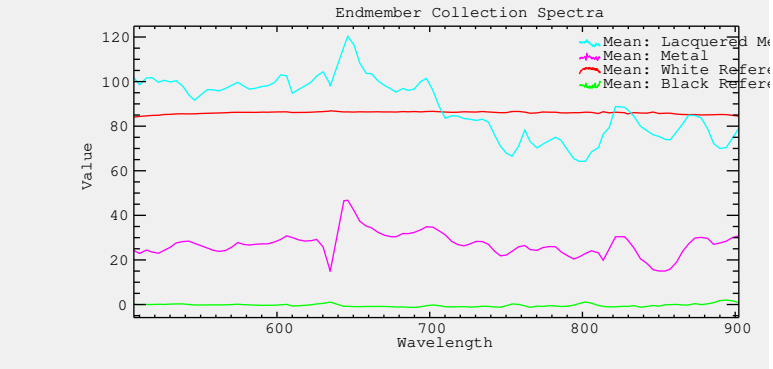
(b) Noisy PNG



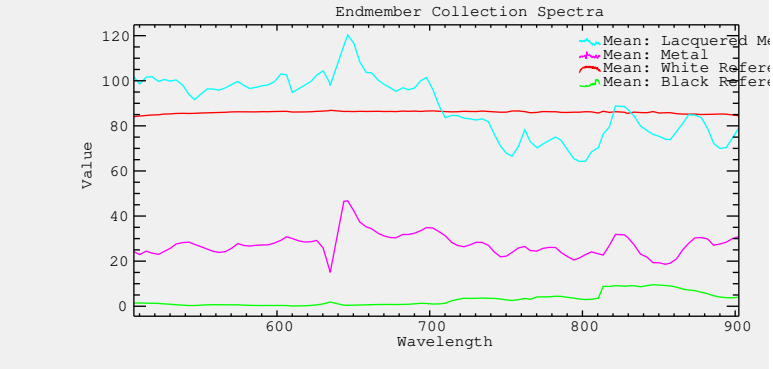
(c) Filtered PNG

Figure 5.15: noice_Luce_1.dat: PNG.

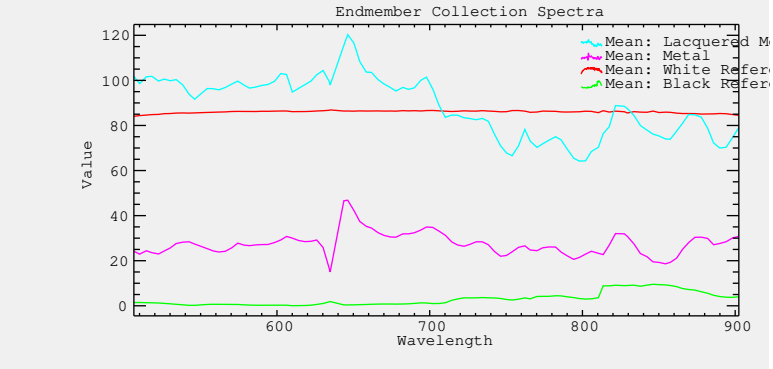
Also in spectral analysis, noise filtering does not affect results.



(a) Original

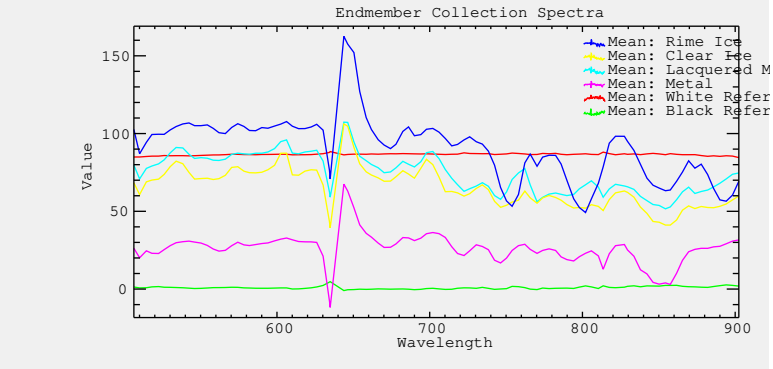


(b) Noisy JPEG

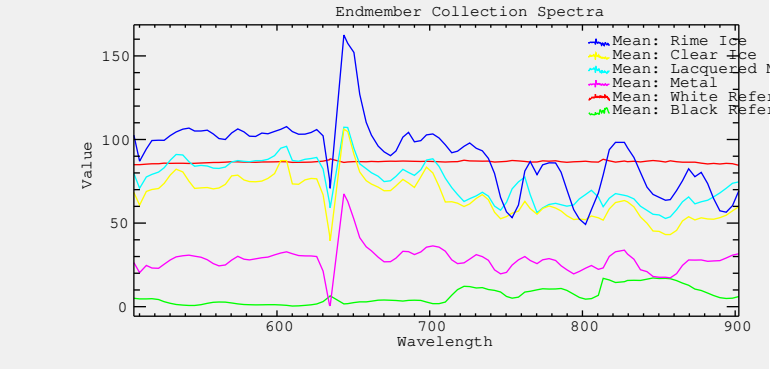


(c) Filtered JPEG

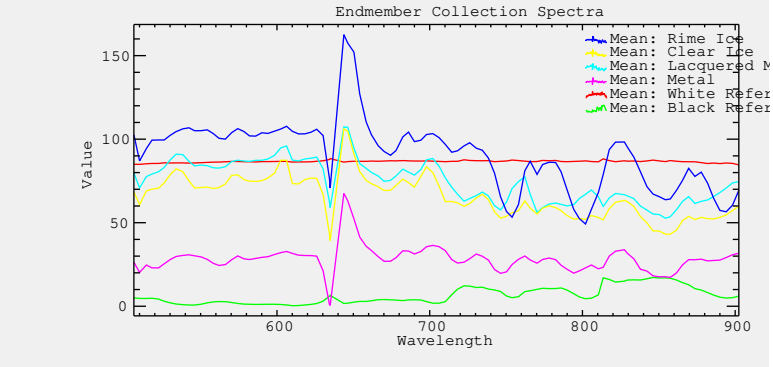
Figure 5.16: noice_Luce_1.dat: JPEG.



(a) Original

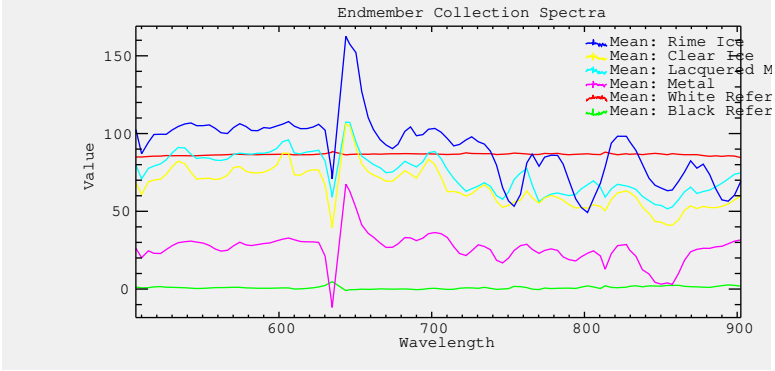


(b) Noisy PNG

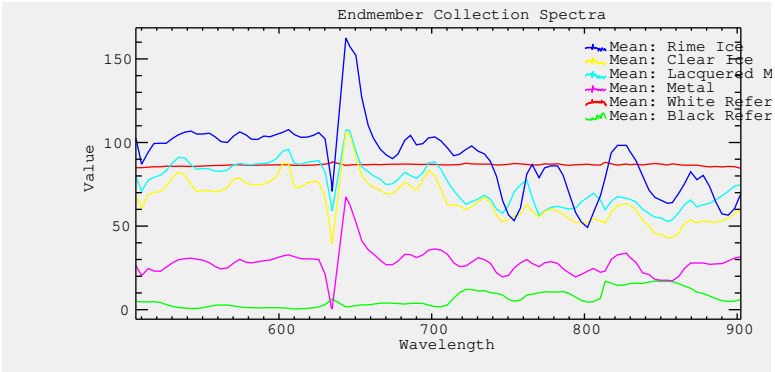


(c) Filtered PNG

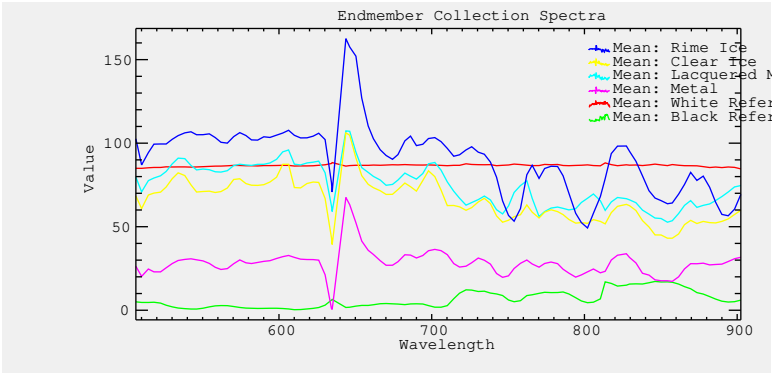
Figure 5.17: ice_Luce_5.dat: PNG.



(a) Original



(b) Noisy JPEG

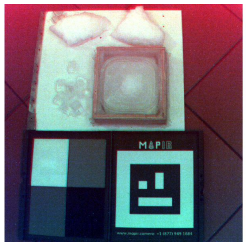


(c) Filtered JPEG

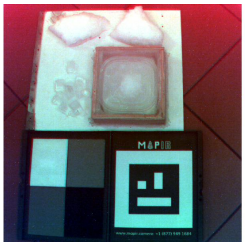
Figure 5.18: ice_Luce_5.dat: JPEG.

When it comes to lossy JPEG, by changing the quality parameter for the compression, i.e. the Quality Factor (QF), it is possible to decrease the size likely to 6 MB (for QF=15). Using Matlab `imwrite()` function it is possible to implement lossy JPEG by changing QF from 100 (lossless) to at maximum 0. Unfortunately,

lossy JPEG supports only up to 12 bits which are not enough to correctly represent original data. 16 bit can be used only for lossless JPEG implementation. In fact, even if spatial information could seems poorly changed, spectral signatures are strongly corrupted, and so endmembers extraction is altered.

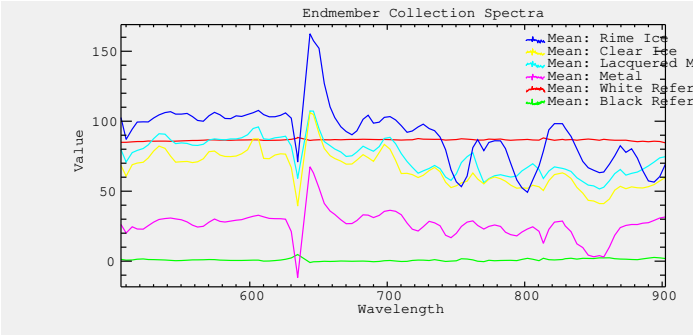


(a) Original

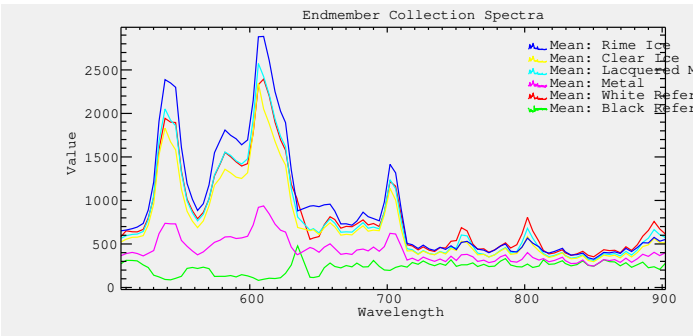


(b) JPEG QF 60 Filtered

Figure 5.19: ice_Luce_5.dat



(a) Original Endmembers



(b) Endmembers JPEG QF=60

Figure 5.20: Endmembers ice_Luce_5.dat

5.3 Lossless Compression through RLS filter

Another tested algorithm involves the usage of a recursive least square filter, as suggested by Song et al. in [1]. The view is to remove spatial correlation through local mean calculation among adjacent pixels. For the first band it is subtracted to the pixel value and then sent to the encoder, while for successive ones it is sent to the RLS filter. Local mean and RLS filter have been implemented from scratch, whereas `arithenco()` MATLAB function have been used as encoder.

5.3.1 Implementation

Emulating what specified in [1], local mean is defined as in 5.1, where $s_z(x, y)$ is the value of the current pixel in the current band, (x, y) the coordinates of the pixel, and $s_z^{NW}(x, y)$,$s_z^{N}(x, y)$,$s_z^{NE}(x, y)$ and $s_z^{W}(x, y)$ are the neighbour pixels in north-west, north, north-est and west positions respectively.

$$\tilde{s}_z(x, y) = \left(\frac{s_z^{NW}(x, y) + s_z^{N}(x, y) + s_z^{NE}(x, y) + s_z^{W}(x, y)}{4} \right) \tag{5.1}$$

Steps have been implemented in MATLAB taking as reference what described in [1].

1. Load the hypercube and extract the 1010×1010 pixels current band image.

2. Calculate local mean through `local_mean = LocalMean(image)`, which evaluates the local mean of each pixel for the current band image.

3. Calculate difference between each (x, y) of the pixel values and its corresponding local mean $d_z(x, y) = s_z(x, y) - \tilde{s}_z(x, y)$. If the current band is the first it is encoded with `arithenco()` function; otherwise, $d_z(x, y)$ is sent to the RLS filter.

4. Filter: RLS filter takes as input $d_z(x, y)$and a vector $\overline{u}$ which contains the differences $d_{z-i}(x, y)$, with $i = 1...p$, and p the number of prediction bands used. The output is the vector $\overline{e}$ which is the difference between $d_z(x, y)$ (desired response) and $\overline{u}$ (vector of buffered input samples) weighted by the filter taps, which recursively adjust.

5. Send $\overline{e}$ to the arithmetic encoder. Repeat the routine for all band images.

The most important parameter is p, i.e. the number of prediction bands used. The higher it is the better will be the performance in terms of compression. In this

experiment 2, 4, 6 and 8 bands have been used for each of the one files of the dataset described in 5.1. Results obtained are reported in Fig.5.21.

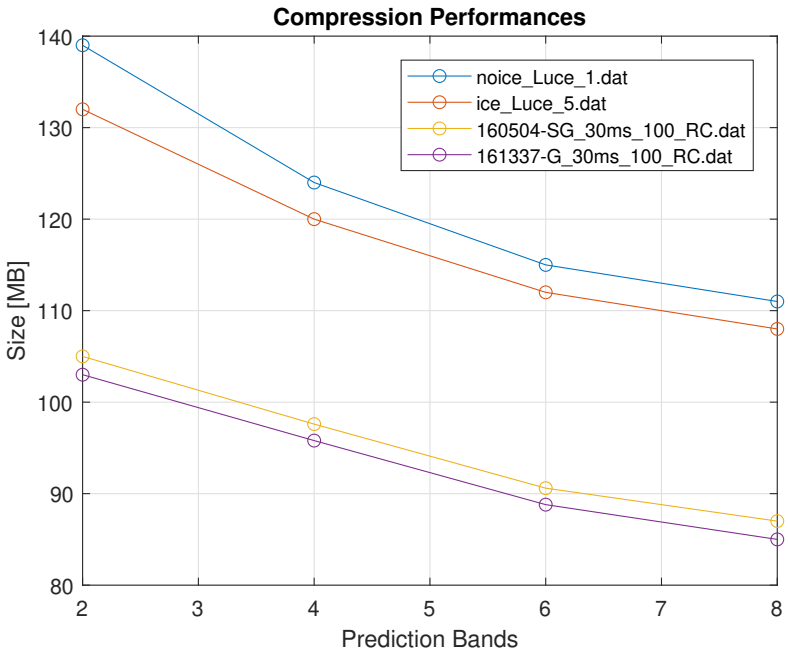


Figure 5.21: RLS algorithm compression performances.

Going from 2 to 4 prediction band there is, on average, an improvement in compression size of 10%, from 4 to 6 of almost 7% and form 6 to 8 of 3.6%. Thus, simulation results coincide with expected ones.
Being lossless, the reconstructed hypercube spectral signatures are not corrupted, but experiments have been limited, since performances are worst compared to other algorithms analysed in this study.
One solution to poor compression results could be implementing and efficient adaptive arithmetic encoder, which should lead to better results.

5.4 Two Dimensional DWT and Tucker Decomposition

According to the state of the art, it is possible to efficiently reduce original size of an HSI by exploiting both spectral and spatial correlation, by using two dimensional discrete Wavelet transform and Tucker Decomposition combined. Taking as reference solution proposed by Karami et al. in [2], lossy solution has been implemented.
This algorithm is based on data representation in tensor notation. In order to manipulate tensor-type data and to use related operations, Tensor Toolbox for MATLAB developed by Sandia National Labs [30] has been deployed.

5.4.1 Implementation

Here follows implementation steps for **encoding**:

1. Load the hyperspectral image and extract three dimension of the datacube: I_1, I_2 and I_3.

2. Apply two-dimensional discrete Wavelet transform to each band of the datacube. The output of this operation are four three dimensional variables (see Fig.5.22): **A** (approximate), **H** (horizontal),**V** (vertical) and **D** (diagonal), each of them with size $\frac{I_1}{2} \times \frac{I_2}{2} \times I_3$. Since lossy version of the algorithm is studied, only **A** component is considered from now on.

3. Convert **A** to a third order tensor $\underline{\mathbf{A}}$

4. Apply Tucker Decomposition to tensor $\underline{\mathbf{A}}$ of size $\frac{I_1}{2} \times \frac{I_2}{2} \times I_3$. Output will be three n-mode matrix in Tucker model $\mathbf{U}^{(n)} \in \mathbf{R}^{I_n \times J_n}$ and a core tensor $\underline{\mathbf{G}} \in \mathbf{R}^{J_1 \times J_2 \times J_3}$, see Fig.5.23

5. Write in binary file core tensor $\underline{\mathbf{G}}$ and three factors

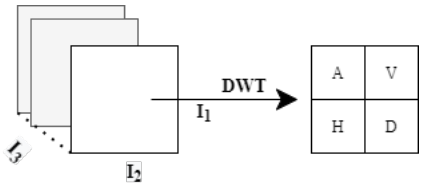


Figure 5.22: 2D Discrete Wavelet Transform decomposition.

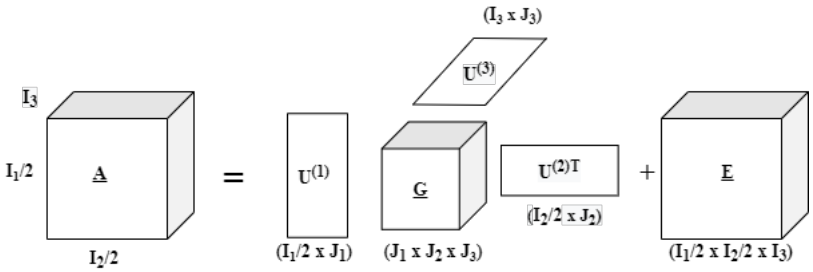


Figure 5.23: Implementation Step 4: Third-order Tucker decomposition.

For **decoding**, instead:

1. Read from file the core tensor $\hat{\underline{\mathbf{G}}} \in \mathbf{R}^{J_1 \times J_2 \times J_3}$ and $\hat{\mathbf{U}}^{(\mathbf{n})} \in \mathbf{R}^{I_n \times J_n}$
2. Reconstruct approximate tensor with n-mode product:
   $\hat{\underline{\mathbf{A}}} = \hat{\underline{\mathbf{G}}} \times_1 \hat{\mathbf{U}}^{(\mathbf{1})} \times_2 \hat{\mathbf{U}}^{(\mathbf{2})} \times_3 \hat{\mathbf{U}}^{(\mathbf{3})} = \hat{\underline{\mathbf{G}}} \times \{\hat{\mathbf{U}}\}$
3. Calculate the inverse two-dimensional discrete Wavelet transform and reconstruct the datacube.

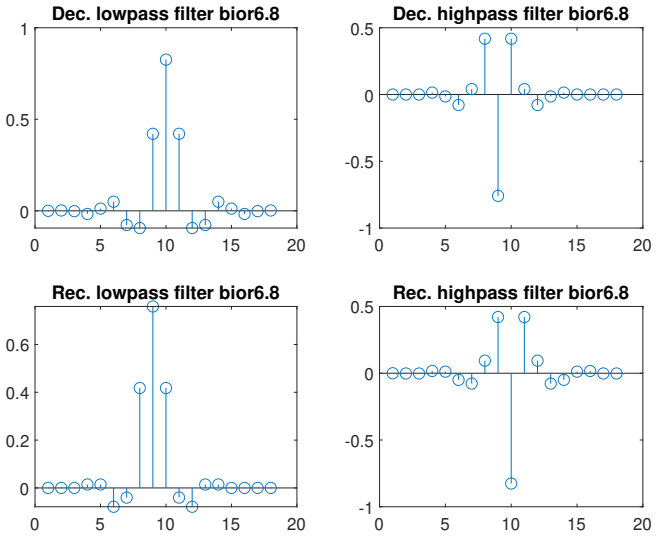


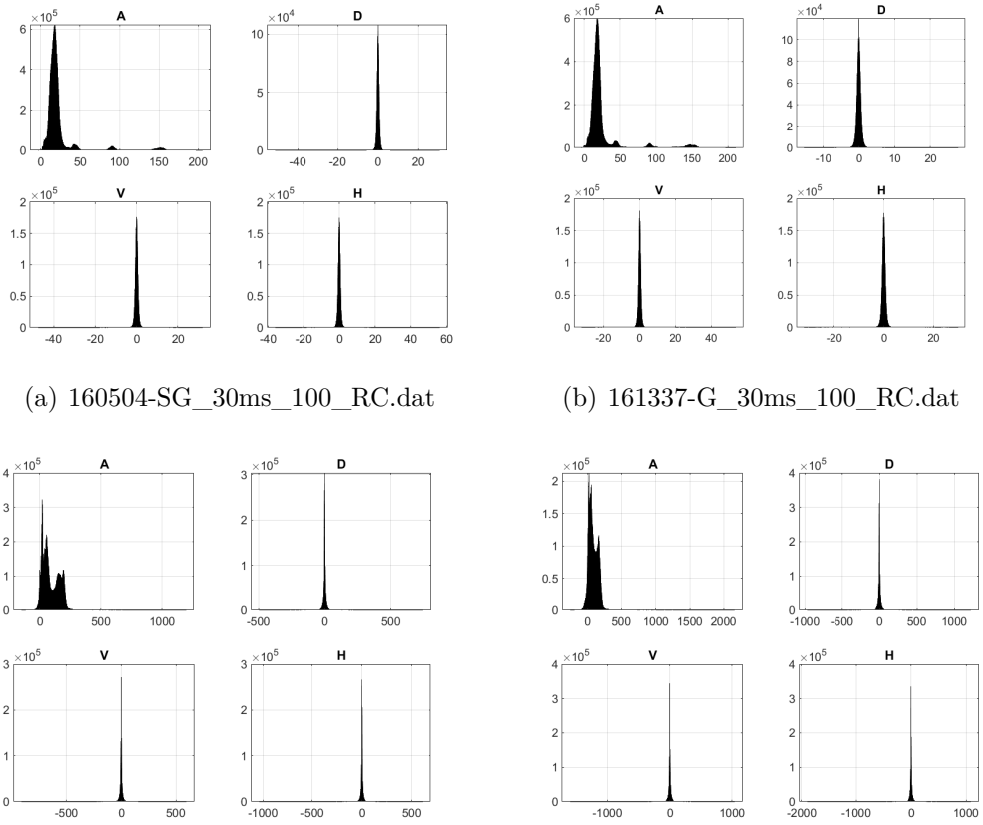
Figure 5.24: Impulse response of LP and HP filters.

Focusing on two-dimensional DWT, it has been used MATLAB function `[cA,cH,cV,cD] = dwt2(X,LoD,HiD)`, which computes the single-level 2-D DWT using the wavelet decomposition lowpass filter `LoD` and highpass filter `HiD`. The

chosen wavelet for this application is the biorthogonal. More precisely, considering SNR defined in (5.2)[2], the chosen one is with 6 synthesis filters and 8 analysis filter, i.e. `'bior6.8'`(Fig.5.24); it, in fact, maximize the SNR for all the four HSI in the dataset described before.

$$SNR_{dB} = 10log_{10}\left(\frac{\|\underline{\mathbf{A}}\|^2}{\|\underline{\mathbf{A}} - \underline{\hat{\mathbf{A}}}\|^2}\right) \tag{5.2}$$

It is important to underline the proposed algorithm is in its lossy version, i.e. only approximate component of the four produced by 2D DWT is used for data representation. Indeed, as it can be seen in Fig.5.25, horizontal (H), vertical (V) and diagonal (D) components' data are distributed around zero, while A contains most of the information.



(a) 160504-SG_30ms_100_RC.dat

(b) 161337-G_30ms_100_RC.dat

(c) noice_Luce_1.dat

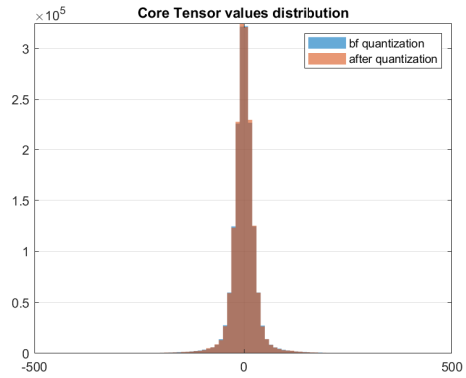
(d) ice_Luce_5.dat

Figure 5.25: Sub-band Images Values Distributions.

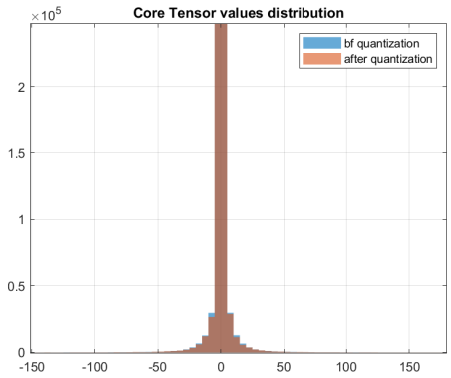
Tensor conversion is done efficiently with `tensor()` function, available in Tensor-Toolbox[30]. Once data are in tensor notation, it is possible to apply Tucker Decomposition to them.
Specifically, it has been used `T = tucker_als(X,R,'param',value,...)` from the same toolbox, which implements Tucker Decomposition in its alternating least square version. It computes the best rank approximation of input tensor `X` (i.e. $\underline{\mathbf{A}}$ in Fig.5.23), according to the dimension specified in vector `R`.
It works as follows:

1. Extract number of dimension and norm of input tensor `X`, i.e. $\underline{\mathbf{A}}$.
2. Set up and error checking on initial guess for $\mathbf{U}$. Initialization is done by computing an orthonormal basis for the dominant dimension in `R`
3. Main loop: iterate until convergence
   I. Considering $n = 1...N$, iterate over all N modes of the tensor. In this case N=3
   II. Evaluate $\mathbf{U^{(n)}}$ and core tensor through current approximation
   III. Compute the error as the difference between current and old fit and check for convergence. Fit is calculated as $fit = 1 - \frac{\sqrt{\|X\|^2 - \|Core\|^2}}{\|X\|^2}$

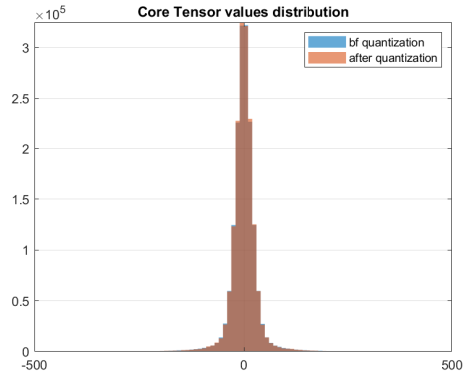
As parameters, tolerance has been set at $1.0e-4$ and maximum number of iteration at 50. However, convergence is met within third iteration.
Also quantization effects have been investigated. Since $\mathbf{U^{(n)}}$ and core tensor $\underline{\mathbf{G}}$ values are not uniformly distributed, Max-Lloyd quantizer have been chosen. For the proposed dataset minimum number of bits needed to have good approximation is 4, so that the quantizer has $2^{n_{bit}}$=2^4=16 levels.
For reconstructing the hypercube $\underline{\hat{\mathbf{A}}}$ from compressed data, n-mode product between core tensor $\underline{\hat{\mathbf{G}}}$ and $\{\hat{\mathbf{U}}\}$ `ttm()` function form Tensor Toolbox[30] has been used.



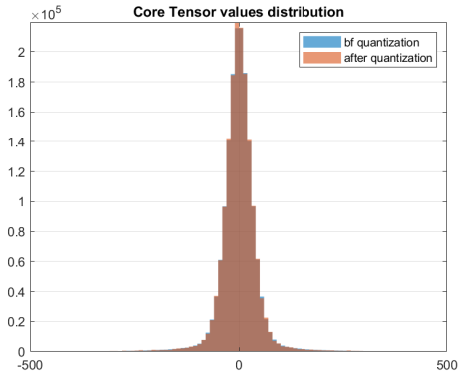
(a) 160504-SG_30ms_100_RC.dat



(b) 161337-G_30ms_100_RC.dat

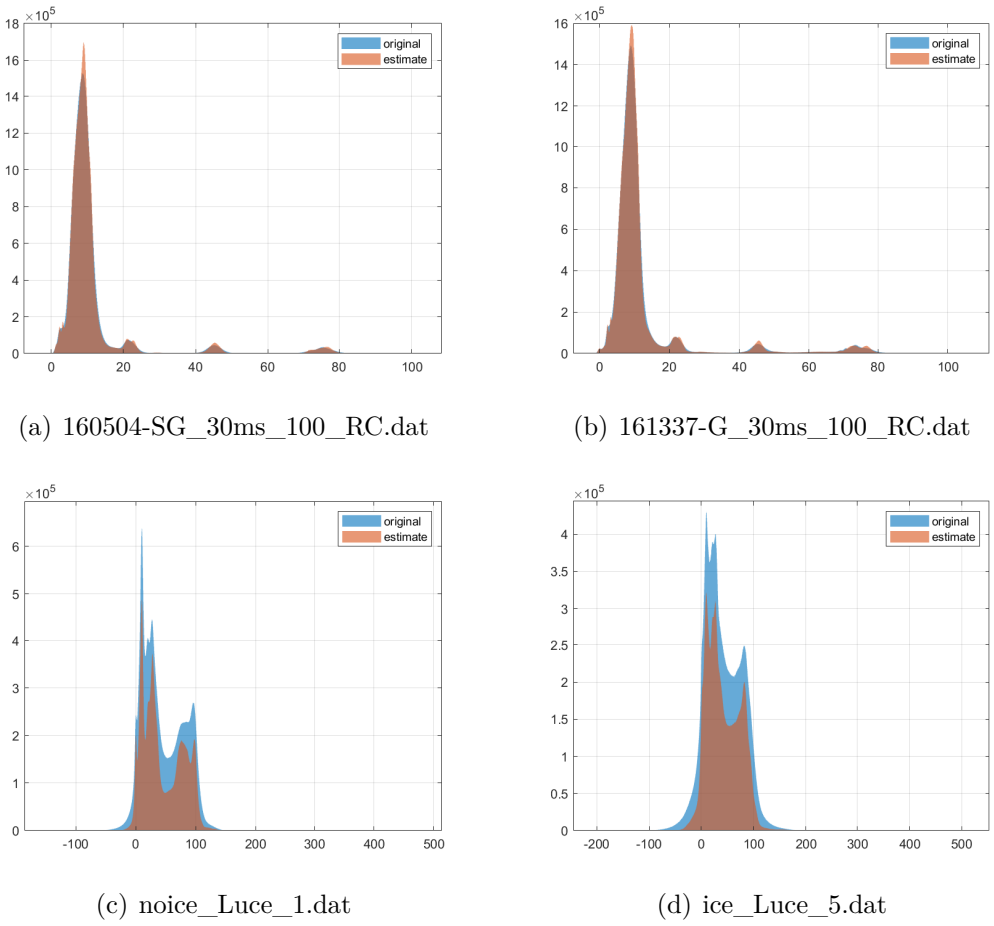


(c) noice_Luce_1.dat



(d) ice_Luce_5.dat

Figure 5.26: Core Tensor values before and after quantization.



(a) 160504-SG_30ms_100_RC.dat

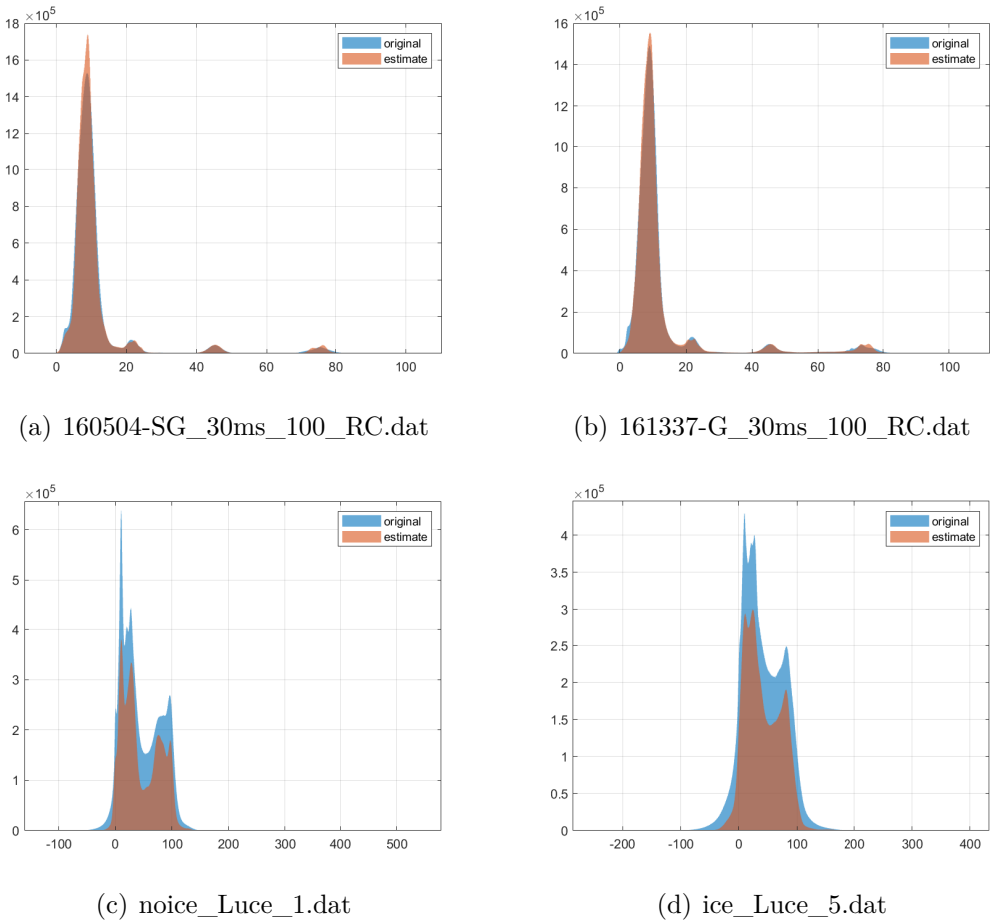
(b) 161337-G_30ms_100_RC.dat

(c) noice_Luce_1.dat

(d) ice_Luce_5.dat

Figure 5.27: Difference between original and reconstructed HSI data distribution.

To be aware of the loss between original hypercube and reconstructed one see Fig.5.27 and Fig.5.28. May be found that even if they have not the same values, their shaping distribution is preserved. It can be noticed that, even if quantization is applied, it does not corrupt the reconstructed (estimated) hypercube. This behaviour translates in the same size of compressed data in both cases.
New size of compressed information needed for the hypercubes reconstruction is **14.5 [MB]** for all of the four files specified in the dataset. Notice that uncompressed datacube in ENVI format have size 389 [MB], as specified in previous section.



(a) 160504-SG_30ms_100_RC.dat

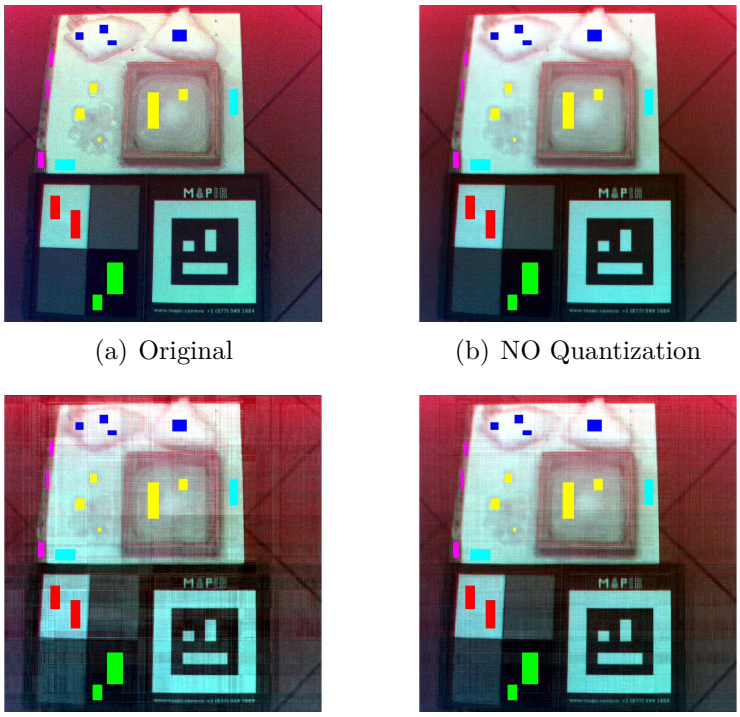
(b) 161337-G_30ms_100_RC.dat

(c) noice_Luce_1.dat

(d) ice_Luce_5.dat

Figure 5.28: Difference between original and reconstructed hsi data distribution with quantization.

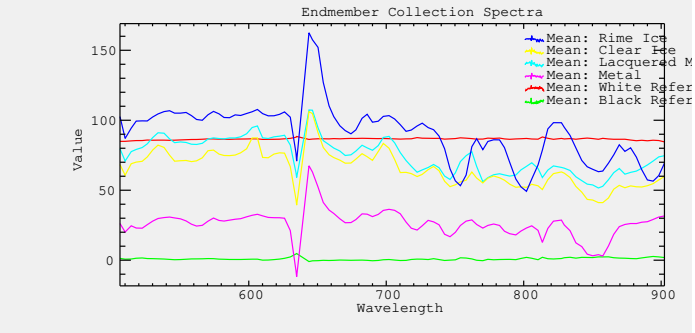
Turning to spectral signatures and endmembers' extraction relatively to the ROIs defined in section 5.1, since the algorithm is lossy, it is expected to observe some difference between those of the original hypercube and the reconstructed ones.



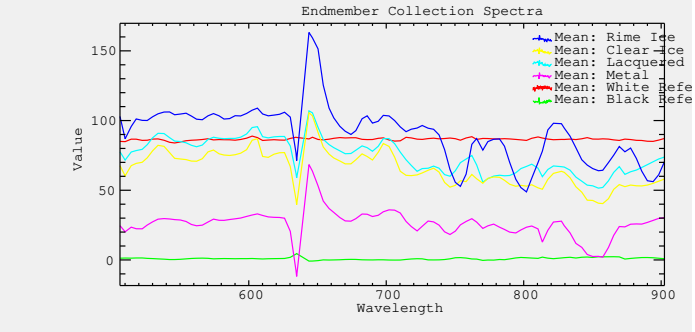
(a) Original (b) NO Quantization

(c) Quantization with 8 bits (d) Quantization with 16 bits

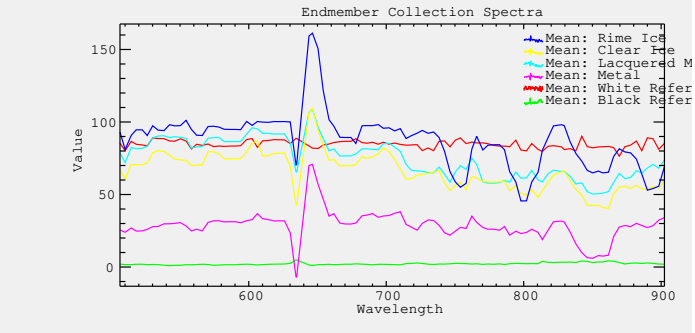
Figure 5.29: Reconstructed hypercubes of ice_Luce_5.dat



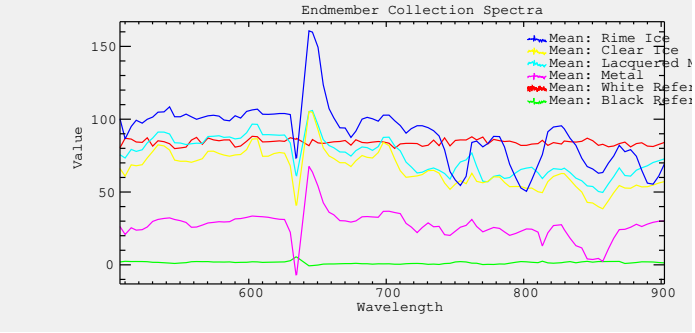
(a) Original



(b) NO Quantization



(c) Quantization with 8 bits



(d) Quantization with 16 bits

Figure 5.30: Reconstructed hypercubes of ice_Luce_5.dat

As it can be observed in Fig.5.29 (c) and (d) the effect of quantization on spatial information is considerable. If no quantization is applied, instead, even if the compression is lossy, there is not appreciable differences between the reconstructed (b) and the original (a) hypercubes.
For what concerns to spectral signatures, peaks are highlighted, even strongly with quantization, but the shape and the behaviour of the endmembers extracted from ROIs is not altered (Fig.5.30).

Conclusions

In short, this thesis explored hyperspectral imaging compression and its basic concepts. Starting from understanding the meaning of spectral information collected by sensors, it illustrated the state of the art of HSI compression algorithms, explaining the elementary principles of compression techniques, such as prediction filters and transforms. Three solutions have been studied and then implemented in software. Simulations have been done on a real dataset acquired in laboratory, on which recognition of ice on metallic surfaces has to be performed.
Experiments have been done to test if spectral signatures were preserved or disrupted and to evaluate the compression ratio. Results show that using JPEG-LS on each per-band image instead of PNG, compression can improve up to 21%. However, if 2D-Wiener filter is used before compression, results produced by the two standards are comparable. Moreover, JPEG in its baseline version, i.e. lossy, is not suited to this type of hyperspectral images, since the maximum supported bits is 12, and so spectral signatures are impaired. Then, lossless compression through recursive least squares filter (RLS), based on what proposed by Song et al. in[1] has been tested. After filter implementation, experimental results demonstrate that with increasing number of prediction bands, size of compressed data is further reduced. However, amount of size reduction achieved is too low with respect to other techniques, and encoding procedure is time demanding. Last explored algorithm was inspired by Karami et al.[2], implemented in lossy version. In fact, after 2D-DWT only the approximate coefficients are first decomposed via TD, and eventually quantized and encoded. In this case compression achieved is high (approximately 96% with respect to original size) and even if there are some differences in the spectral signatures, their behaviour is not affected by the compression algorithm.
Future work that could not have been handled in this thesis, could address an optimized implementation of the RLS filter and an efficient adaptive arithmetic encoder, for the second explored algorithm, which should lead to better performances. Another possible further development could be testing different implementation of the Tucker Decomposition in the last proposed solution. Moreover, taking as reference spectral signatures of the compressed hypercubes, it could be meaningful

exploring and analyzing classification results, that was out of the scope of this work.

Bibliography

[1] Song Jinwei, Zhongwei Zhang, and Xiaomin Chen. «Lossless compression of hyperspectral imagery via RLS filter». In: *Electronics Letters* 49 (Aug. 2013), pp. 992–994. DOI: 10.1049/el.2013.1315 (cit. on pp. 2, 23, 29, 30, 54, 66).

[2] Azam Karami, Mehran Yazdi, and Grégoire Mercier. «Compression of Hyperspectral Images Using Discerete Wavelet Transform and Tucker Decomposition». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 5.2 (2012), pp. 444–450. DOI: 10.1109/JSTARS.2012.2189200 (cit. on pp. 2, 22, 32, 34–37, 56, 58, 66).

[3] Dimitris G. Manolakis, Ronald B. Lockwood, and Thomas W. Cooley. *Hyperspectral Imaging Remote Sensing: Physics, Sensors, and Algorithms*. Cambridge University Press, 2016. DOI: 10.1017/CBO9781316017876 (cit. on pp. 3–7, 10, 44).

[4] Umamahesh Srinivas, Yi Chen, Vishal Monga, N.M. Nasrabadi, and Trac Tran. «Discriminative graphical models for sparsity-based hyperspectral target detection». In: July 2012, pp. 1489–1492. ISBN: 978-1-4673-1160-1. DOI: 10.1109/IGARSS.2012.6350822 (cit. on p. 3).

[5] Guolan Lu and Baowei Fei. «Medical hyperspectral imaging: a review». In: *JOURNAL OF BIOMEDICAL OPTICS* 19.1 (Jan. 2014). ISSN: 1083-3668. DOI: 10.1117/1.JBO.19.1.010901 (cit. on pp. 3, 4).

[6] Manuel Campos-Taberner. «Development of an Earth observation processing chain for crop biophysical parameters at local and global scale». PhD thesis. July 2017. DOI: 10.13140/RG.2.2.30411.28962 (cit. on p. 5).

[7] Geert Verhoeven. «The reflection of two fields – Electromagnetic radiation and its role in (aerial) imaging». In: 55 (Oct. 2017), pp. 13–18. DOI: 10.5281/zenodo.3534245 (cit. on p. 5).

[8] Robert A Schowengerdt. *Remote sensing: models and methods for image processing*. Elsevier, 2006 (cit. on pp. 7, 8).

[9] Mario A. Gomarasca. «Elements of Remote Sensing». In: *Basics of Geomatics*. Dordrecht: Springer Netherlands, 2009, pp. 123–184. ISBN: 978-1-4020-9014-1. DOI: 10.1007/978-1-4020-9014-1_4. URL: https://doi.org/10.1007/978-1-4020-9014-1_4 (cit. on p. 8).

[10] Xiaoyong Zhuge, X. Zou, and Yuan Wang. «A Fast Cloud Detection Algorithm Applicable to Monitoring and Nowcasting of Daytime Cloud Systems». In: *IEEE Transactions on Geoscience and Remote Sensing* 55 (Nov. 2017), pp. 6111–6119. DOI: 10.1109/TGRS.2017.2720664 (cit. on p. 9).

[11] James Theiler, Amanda Ziemann, Stefania Matteoli, and Marco Diani. «Spectral Variability of Remotely Sensed Target Materials: Causes, Models, and Strategies for Mitigation and Robust Exploitation». In: *IEEE Geoscience and Remote Sensing Magazine* 7.2 (2019), pp. 8–30. DOI: 10.1109/MGRS.2019.2890997 (cit. on pp. 9, 10).

[12] Natalia Matsapey, Jenny Faucheu, Manuel Flury, and David Delafosse. «Design of a gonio-spectro-photometer for optical characterization of gonio-apparent materials». In: *Measurement Science and Technology* 24 (June 2013), p. 065901. DOI: 10.1088/0957-0233/24/6/065901 (cit. on p. 10).

[13] Yaman Dua, Vinod Kumar, and Ravi Shankar Singh. «Comprehensive review of hyperspectral image compression algorithms». In: *Optical Engineering* 59.9 (2020), pp. 1–39. DOI: 10.1117/1.OE.59.9.090902. URL: https://doi.org/10.1117/1.OE.59.9.090902 (cit. on pp. 11–13, 18, 22).

[14] Enrico Magli. *Lecture notes in Image and Video Processing (01QWKBG)*. Dip. di Elettronica, Politecnico di Torino, A.Y. 2018/2019 (cit. on pp. 14, 15, 26–28).

[15] Charles Burrus, R. Gopinath, and H. Guo. «Introduction to Wavelets and Wavelet Transform—A Primer». In: *Recherche* 67 (Jan. 1998) (cit. on p. 17).

[16] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017 (cit. on p. 17).

[17] Simon Haykin. *Adaptive filter theory*. 5th. Upper Saddle River, NJ, 2002 (cit. on pp. 19–22).

[18] Subash K and Thyagharajan K K. «Hyperspectral Image Compression Algorithms – A Review». In: *Advances in Intelligent Systems and Computing* 325 (Jan. 2014), pp. 127–138. DOI: 10.1007/978-81-322-2135-7_15 (cit. on p. 22).

[19] Ramhark J Yadav and MS Nagmode. «Compression of hyperspectral image using PCA–DCT technology». In: *Innovations in Electronics and Communication Engineering*. Springer, 2018, pp. 269–277 (cit. on p. 23).

[20] R Nagendran and A Vasuki. «Hyperspectral image compression using hybrid transform with different wavelet-based transform coding». In: *International Journal of Wavelets, Multiresolution and Information Processing* 18.01 (2020), p. 1941008 (cit. on p. 23).

[21] Shaohui Mei, Bakht Muhammad Khan, Yifan Zhang, and Qian Du. «Low-complexity hyperspectral image compression using folded PCA and JPEG2000». In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 4756–4759 (cit. on p. 23).

[22] Marco Conoscenti, Riccardo Coppola, and Enrico Magli. «Constant SNR, rate control, and entropy coding for predictive lossy hyperspectral image compression». In: *IEEE Transactions on Geoscience and Remote Sensing* 54.12 (2016), pp. 7431–7441 (cit. on p. 24).

[23] Khalid Sayood. *Lossless compression handbook*. Elsevier, 2003 (cit. on pp. 26, 28, 29).

[24] Enrico Masala. *Lecture notes in Elaborazione e Trasmissione di Informazioni Multimediali (02GPJOV)*. A.Y. 2019/2020 (cit. on pp. 26–28).

[25] Tamara G Kolda and Brett W Bader. «Tensor decompositions and applications». In: *SIAM review* 51.3 (2009), pp. 455–500 (cit. on pp. 32–34).

[26] L3 Harris Geospatial Docs Center. *Using ENVI- Atmospheric Correction.* URL: `https://www.%20harrisgeospatial.com/docs/AtmosphericCorrection.html#empirical_line_calibration` (cit. on p. 39).

[27] Inc. L3 Harris Technologies. *ENVI version 5.2*. 1988-2020 Harris Geospatial Solutions, Inc. (cit. on p. 40).

[28] MATLAB and Hyperspectral Image Processing Toolbox. *Release 2020b*. Natick, Massachusetts, United States: The MathWorks Inc. (cit. on p. 41).

[29] L3 Harris Geospatial Docs Center. *Using ENVI- Collect Endmember Spectra.* URL: `https://www.l3harrisgeospatial.com/docs/collectingendmemberspectra.html` (cit. on p. 44).

[30] Tamara G. Kolda Brett W. Bader et al. *Tensor Toolbox for MATLAB,Version 3.2.1*. Apr. 2021. URL: `www.tensortoolbox.org` (cit. on pp. 56, 59).