

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA CHIMICA E DEI PROCESSI INDUSTRIALI

**Tesi di Laurea Magistrale in
Ingegneria Chimica e dei Processi Industriali**

**Feature extraction and segmentation of
hyperspectral images**

Relatore: Prof. Massimiliano Barolo

Correlatore: Prof. Marco Reis

Laureanda: ANNAPIA SOCCIO

ANNO ACCADEMICO 2018 – 2019

Abstract

Hyperspectral images can collect a significant amount of data, from which useful information can be extracted with proper techniques. This work proposes an approach for hyperspectral images segmentation without direct application of common clustering methods on the hyperspectral data. Since clustering is computationally very expensive, the proposed approach reduces the spectral dimension of the image, through principal component analysis, and its spatial dimension, through wavelet transform, in order to apply the clustering algorithm on a lower resolution version of the data and then train a classifier to label the high resolution image. The method is tested on an image measuring reflectance in the Near Infrared region of the electromagnetic spectrum.

Riassunto

Mentre le normali strumentazioni di fotografia si limitano in genere a catturare l'intensità della riflettanza di una scena o oggetto per un numero limitato di bande spettrali, corrispondenti a quelle necessarie per produrre una immagine interpretabile dall'occhio umano, le fotocamere iperspettrali possono catturare, per ogni pixel, l'intera risposta spettrale in un ampio intervallo quasi continuo, dipendente dal tipo di fotocamera stessa. Di conseguenza, una elevata quantità di misurazioni è disponibile per l'estrazione di informazioni utili. L'estensione dei dati raccolti in una singola immagine iperspettrale è potenzialmente un vantaggio in termini di quantità di informazione ricavabile riguardo la scena o l'oggetto, ma è anche una "maledizione" (Bellman, 1961) poiché elaborare dataset di grandi dimensioni risulta più costoso computazionalmente e dunque in genere più complesso. Anche i normali spettrometri riescono a rilevare ampi range dello spettro elettromagnetico, ma senza fornire alcuna informazione spaziale, ovvero misurando la riflettanza dell'intera scena e non di un singolo punto di essa, fornendo quindi informazioni medie qualora il target non fosse ben delineato. In applicazioni industriali come il controllo qualità o il monitoraggio di processo è importante che le informazioni spettrali, correlabili alla composizione chimica e alla natura fisica, siano identificate in contemporanea per numerosi punti dello spazio in maniera specifica e non in media. Da qui l'applicazione di fotocamere iperspettrali, con la necessità di metodi in grado di processare elevate quantità di dati in modo computazionalmente efficiente. Il presente lavoro, in particolare, si focalizza sul problema della segmentazione di una immagine iperspettrale, individuando la "naturale" classificazione di oggetti/materiali componenti l'immagine. Se nessuna informazione è disponibile riguardo alla specifica "firma spettrale" (*spectral signature*) dei materiali studiati, è necessario eseguire la classificazione con tecniche di apprendimento non supervisionato (*unsupervised learning*), che spesso richiedono una elevata quantità di calcoli dovendo tener conto delle distanze fra tutti i punti e/o gruppi. In particolare, una delle tecniche più rigorose, il clustering gerarchico, richiedendo il calcolo di una matrice di distanze fra tutte le possibili coppie di punti, è anche una delle più impegnative dal punto di vista del costo computazionale, che diventa proibitivo per dataset di grandi dimensioni. Nel seguente studio si propone di ridurre la dimensione spaziale e spettrale della immagine iperspettrale, applicando poi tecniche di classificazione non supervisionata alla versione a bassa risoluzione del dataset originale, e definendo successivamente un classificatore lineare sulla base dei raggruppamenti ottenuti nella fase di apprendimento non supervisionato, con lo scopo di applicare il classificatore lineare alla immagine ad alta risoluzione per ottenerne la segmentazione. Essendo il costo computazionale della classificazione non supervisionata predominante, l'obiettivo è mostrare

che tale successione di passaggi riduce di fatto lo sforzo e dunque il tempo richiesto per ottenere la segmentazione della immagine iperspettrale. L'applicazione della tecnica viene illustrata su un caso di esempio, che vede la necessità di distinguere i diversi materiali di una serie di oggetti di cui si cattura la riflettanza nel vicino infrarosso (NIR). Per ridurre la dimensione spettrale si propone di applicare l'analisi delle componenti principali, che proietta le "osservazioni" (ovvero i punti dell'immagine) su una nuova base costruita in modo tale che il primo vettore, o "prima componente", catturi la maggior parte della variabilità dei punti, massimizzando l'errore residuo, e ogni vettore successivo, ortogonale ai precedenti, catturi di volta in volta la maggior parte della variabilità residua. In questo modo le prime componenti sono in grado di trattenere la gran parte dell'informazione, nell'esempio specifico più del 99% della variabilità totale. Successivamente, la risoluzione spaziale viene ridotta con applicazione della trasformata wavelet, utilizzando wavelet di Haar. Ad ogni passo, l'applicazione della trasformata dimezza larghezza e lunghezza dell'immagine tramite una operazione di media fra punti vicini e sottocampionamento (*downsampling*). In questo modo, la risoluzione può essere abbassata fino al limite in cui gli oggetti sono ancora ben distinguibili. Successivamente, avendo ridotto la dimensione sia spettrale che spaziale dell'immagine, è possibile applicare un metodo di classificazione non supervisionata, quale ad esempio il metodo gerarchico, qui preferito ad altri perché più rigoroso. Avendo associato ogni punto ad una classe, è dunque possibile "addestrare" un classificatore lineare da poter applicare all'immagine ad alta risoluzione. I classificatori lineari definiscono degli iperpiani che dividono lo spazio delle osservazioni in regioni corrispondenti a classi diverse. Nel caso di esempio, il classificatore sviluppato su un dataset ridotto performa ottimamente anche sulla immagine ad alta risoluzione, mostrando come le tecniche proposte per ridurre la dimensione del problema consentano comunque di mantenere le informazioni significative contenute nel dataset originale. Ovviamente ciò rimane vero solo se le tecniche sono applicate correttamente. Infatti, il costo computazionale continua a diminuire riducendo la risoluzione, ma la qualità del risultato a risoluzioni troppo basse viene compromessa, non preservandosi la quantità di informazione minima richiesta per un corretto addestramento del classificatore. Di fatto, uno studio a diverse risoluzioni mostra l'ottimo dal punto di vista del bilancio fra qualità del risultato e sforzo computazionale. Si noti, infine, come la processabilità di una immagine iperspettrale non sia solo limitata dal numero di operazioni elementari richieste, direttamente collegata al tempo impiegato, che dipende dalla velocità della CPU. Infatti, l'effettiva realizzabilità della classificazione potrebbe essere compromessa, poiché limiti di RAM possono rendere impossibile la memorizzazione delle matrici di distanza necessarie per dataset di dimensioni molto elevate, nel qual caso metodi alternativi di elaborazione dei dati sono necessari, come ad esempio la suddivisione del dataset in parti. In futuro, i limiti della procedura proposta possono essere esplorati processando immagini iperspettrali più complesse, ad esempio immagini con trame intricate come foto di vegetazione.

La struttura del documento vede nel Capitolo 1 una introduzione generale alla fotografia iperspettrale e ai problemi connessi all'uso di questa tecnica, in particolare nell'ambito della segmentazione, proponendo una procedura di segmentazione alternativa alla diretta applicazione degli algoritmi di *clustering*; il Capitolo 2 richiama i concetti fondamentali dei metodi matematici utilizzati per la procedura; infine, i risultati ottenuti applicando la tecnica ad un caso di esempio sono presentati e commentati Capitolo 3.

Table of contents

Introduction	12
1 Hyperspectral images and segmentation	14
1.1 Hyperspectral imaging.....	14
1.1.1 A problem to be solved	14
1.1.2 Integration of space and frequency: Hyperspectral Imaging	16
1.2 Image segmentation.....	17
1.2.1 Introduction	17
1.2.2 Method proposed.....	18
2 Methods.....	20
2.1 Image acquisition.....	20
2.2 Opening a hyperspectral image	21
2.3 Principal Component Analysis	23
2.3.1 General purpose.....	23
2.3.2 PCA Fundamentals.....	25
2.4 Wavelet Transform	28
2.4.1 General overview	28
2.4.2 The wavelet transform	30
2.5 Clustering.....	38
2.5.1 Introduction	38
2.5.2 Clustering fundamentals and methods	39
2.6 Classifier.....	45
2.6.1 Introduction	45
2.6.2 Classification fundamentals and methods	46
2.6.3 Input dataset	50
2.7 Refining the model	52

3	Results and Discussion.....	56
3.1	Data inspection	56
3.2	Principal component analysis	60
3.2.1	Pre-processing	60
3.2.2	Number of components	61
3.2.3	Visualization of results.....	66
3.3	Wavelet transform	74
3.3.1	Pre-processing	74
3.3.2	Visualization of results.....	74
3.4	Clustering.....	80
3.4.1	Pre-processing	80
3.4.2	Visualization of results.....	83
3.5	Classifier.....	88
3.5.1	Pre-processing	88
3.5.2	Results	89
3.6	Problem dimension, time complexity and execution time	92
3.6.1	Introduction to the problem.....	92
3.6.2	Overall computational cost.....	95
3.7	Refining the model	97
3.7.1	Spectral resolution.....	97
3.7.2	Spatial resolution.....	99
	Conclusions	104
	Bibliography.....	106
	Web sites	108
	Symbols.....	110
	List of tables and figures	112
	Appendix	116
	Appendix 1 – Hyperspectral camera datasheet	117
	Appendix 2 – Camera report	119
	Appendix 3 – Rescaling and contrast-stretching.....	120

Introduction

Common colour pictures capture reflectance of objects and scenes for specific electromagnetic bands corresponding to the three colours needed to compose a colour image, either in printing or on screen. Conversely, hyperspectral images can capture a broad and almost continuous spectral response for each pixel representing the different points of a scene, also for regions of the electromagnetic spectrum not visible to the human eye. Clearly, several industrial applications such as process monitoring or product quality assessment could benefit from the use of this technique. The main challenge in deriving useful information is related to the size of the dataset itself, that is also the main advantage of this sensing technique. Various solutions have been proposed in literature to overcome this obstacle. Chapter 1, besides a more detailed presentation of the “curse of dimensionality” (Bellman, 1961), provides a synthetic overview of the techniques commonly utilized for this scope. In this study, the attention is focused on extracting information concerning the different materials appearing in a captured scene, a problem pertaining to the so-called *image segmentation*. The objective of the work is to propose a procedure combining several size reducing and classification techniques in order to *segment* an image that otherwise would require unbearable computational effort to be processed for extracting meaningful information. This procedure is briefly introduced and schematized at the end of Chapter 1. Chapter 2, then, presents the theoretical background necessary to understand and properly apply the methods employed. Chapter 3, finally, collects and comments the results obtained for a sample dataset very similar to possible industrial applications, capturing spectral response in the near infrared region for objects made of different materials.

Chapter 1

Hyperspectral images and segmentation

In this Chapter a synthetic overview on hyperspectral imaging is presented, highlighting advantages and problems of this technique, with particular focus on the dimensionality issues. Afterwards, a procedure to approach the segmentation of hyperspectral images is introduced schematically, leaving to the following Chapters the detailed illustration of the technique, that is the objective of this study.

1.1 Hyperspectral imaging

1.1.1 A problem to be solved

The human eye is only able to capture signals in a limited range of the electromagnetic field. According to the widely accepted Retinex theory (Land *et al.*, 1971), there are three independent cone systems consisting of receptors peaking in three different wavelength regions of the visible spectrum, going from about¹ 380 nm to 740 nm. Each system forms one image of the world, and subsequently the three images contributes to generate the final colour picture of the outside scene.

The common still image camera, that is the most widely used image capturing tool and historically the first one, is also optimised to capture light photons from the visible spectrum, and specifically from the wavebands necessary to build an image interpretable by the human

¹ De facto, the amplitude of the range perceived by the human eye varies from person to person. The most common minimum and maximum value for this range are 380 nm and 740 nm, therefore this portion of the electromagnetic spectrum is conventionally called “colour spectrum” or “visible spectrum”.

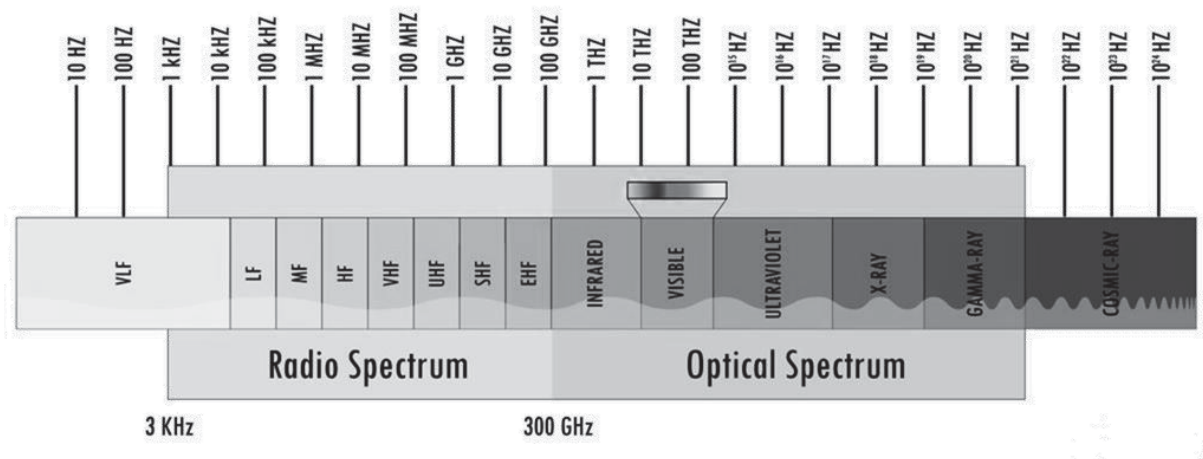


Figure 1.1 – Electromagnetic spectrum. Schematic representation by NASA² (black and white conversion of the original colour image)

eye, providing very limited spectral information (Elmasry *et al.*, 2012). In reality, the visible light is only a very small portion of the electromagnetic spectrum (Figure 1.1).

The solar radiance itself, in fact, covers almost the whole electromagnetic spectrum (Iqbal, 1983), and every material can absorb specific wavelengths more effectively than others, depending on its physical and chemical properties. As a consequence, peaks in the reflectance spectrum, detected by specific equipment, correspond to low absorption of the incident radiance and define the so-called “spectral signature”, unique of each material (Ceamanos *et al.*, 2016). Hence, the possibility to exploit these differences in reflectance to characterize different materials through spectral response detection.

Detection in the “frequency domain” of the spectral response and in the “spatial domain³” of traditional imaging technology are usually separate techniques using different instrumentation. Nevertheless, several industrial applications would benefit from simultaneous exploration of both spatial and frequency domain. A representative example is given by the meat industry, where assessing the quality of the product usually requires destructive, time consuming, expensive evaluation methods, often necessitating sample preparation (Damez *et al.*, 2008). Conventional spectroscopy can give very detailed information about different materials with a non-invasive inspection of the sample, but, if the sample is analysed as a whole, only average spectral information can be obtained, and spectral response of specific areas can be measured only removing samples from those areas of the product, resulting again in a destructive method. On the other hand, colour imaging can only provide limited information about the chemical and physical characteristics of the product,

² Refer to https://www.nasa.gov/directorates/heo/scan/spectrum/txt_electromagnetic_spectrum.html (last access: 07/2019)

³ Technically, digital colour imaging detects frequency from three wavebands, related to three colours used to build all the visible spectrum on modern screens, but the spectral information is so limited, if compared to the amount of spatial information provided, that traditional imaging can be considered a mostly “spatial” detection technique.

and may be insufficient in quality assessment, especially in today's highly competitive market requiring high quality products at an affordable price (Elmasry *et al.*, 2012). Similar problems are encountered in the pharmaceutical industry, where even higher quality requirements demand a tight real-time process monitoring and control at all stages, and traditional destructive techniques preclude the possibility to examine the whole batch, limiting the inspection to small samples (Gowen *et al.*, 2008).

Therefore, a technique able to integrate the detection of both spatial and frequency information would be a very promising solution. From this industrial need of non-invasive powerful techniques of process monitoring and quality assessment derives the active interest of the scientific community in hyperspectral imaging.

1.1.2 Integration of space and frequency: Hyperspectral Imaging

A hyperspectral image collects image information, in the form of reflectance of each point on several adjacent narrow spectral bands. In other words, for each pixel of the full spatial image of the scene/object a hyperspectral camera is able to detect the complete reflectance spectrum. The result is a three-dimensional image containing both spatial and reflectance information of the scene under examination.

The great advantage of the hyperspectral imaging on other techniques, that is the capability to provide a high amount of data, is also the main drawback of this approach, theatrically called the "curse of dimensionality" (Bellman, 1961). In fact, extraction of meaningful information from large dataset might be particularly challenging for hyperspectral images, due to the high correlation among adjacent spectral bands, that translates into significant redundancy (Ceamanos *et al.*, 2016). In practice, for most materials the reflectance varies gradually and slowly along the electromagnetic spectrum, often even near the peaks characteristic of each material, being the sampling of hyperspectral camera usually very dense, or, in other words, being the width of each detected band very small, in order to obtain an almost continuous representation of the spectral response. As a consequence, some processing of the hyperspectral image to be examined might be necessary to obtain actual information, since "data" and "information" do not necessarily have the same meaning. It goes without saying that, in addition to the issues specifically related to the data dimensionality, hyperspectral imaging is affected also by usual problematics of data acquisition, for example noise reduction.

In literature, several pre-processing methods have been proposed, for both generic and specific applications.

Noise might be present even simply because of the temperature reached by the electronic components in the camera, that generates disturbances in the detected signal (Ceamanos *et al.*, 2016), but some applications might have specific disturbances in the data detection related to

their peculiarities, for example in remote sensing of Earth's surface, to study crops and other natural materials, specific methods for correcting the disturbance introduced by the presence of a thick layer of atmosphere have to be applied (Gao *et al.*, 2009).

Dimensionality reduction might involve feature selection, if it is possible to identify the bands that maximize the separability of different classes in the image (Backer *et al.*, 2005). In other cases, especially for highly redundant and correlated data, as is often for hyperspectral images, it might be better to consider feature extraction, that transforms or projects the data into a new space where a reduced dataset can be defined, sometimes with better properties, like the non-correlation of the newly defined features in principal component analysis (PCA). Sometimes, combining specific spectral bands, it is possible to define features with a physical meaning through calculation of the so-called physical indices. This last practice is common in studies of vegetation (Ceamanos *et al.*, 2016).

Dataset size reduction does not involve necessarily and only the spectral domain. Also the spatial extension of the image can be subject to compression, through techniques like the application of a transform, for example the wavelet transform.

Finally, more complex transformation and representation of the hyperspectral data might be useful in specific applications.

1.2 Image segmentation

1.2.1 Introduction

For quality assessment and process monitoring through acquisition of hyperspectral images, it is essential to be able to distinguish different objects or materials in the image itself. According to Yin (2016) image segmentation is the “process of partitioning an image into multiple segments” – in other words it subdivides o “segments” the image into its constituent regions. In truth, also edge detection falls within the definition of segmentation, even if it does not properly find objects but only border lines. In general, being able to segment an image from the point of view of object shape and edges is fundamental in application of process monitoring such as automated inspection of electronic circuits (Gonzalez *et al.*, 2010). Also detection of materials, that is fundamental for example in chemical applications like pharmaceutical process monitoring, is related to the detection of different “regions” in a broader sense, since different materials will have a different spectral signature and will form different clusters (or “regions”) in the hyperspace of the observation points; apart from noise, in fact, pixels capturing regions of the same material have a very similar spectral response. Consequently, in the reflectance space where each point collects the intensities of the responses detected for one specific pixel relatively to the several spectral bands, it is possible to perform clustering operations to detect different materials. Indeed, clustering is one of the

most diffused unsupervised segmentation techniques. In essence, all clustering algorithms try to assign different labels to the data points on the basis of a definition of “distance” or dissimilarity between different groups and points. It is an unsupervised technique because all the groups are created by the algorithm and no information is given except from the distance definition and, sometimes, a termination rule, that could be as simple as fixing the total number of final groupings to be found. Being able to group the different materials constituting objects in an image in a completely autonomous and untrained way, this technique is very appealing for segmentation purposes, but, because of the distance computations for each point, it comes with the huge disadvantage of high computational cost. In fact, the most expensive algorithms commonly used, that are hierarchical algorithms, can be even of the order of $n^2 \log n$, being n the number of pixels of the image (Kurita 1991). If the whole detected spectral response of a hyperspectral image has to be clustered, the computational cost can be unbearable at worst. Hence the necessity to try to reduce the size of the dataset without compromising the classification objective.

1.2.2 Method proposed

This work suggests a possible method involving a sequence of size-reducing techniques able to preserve the effectiveness of the clustering procedure, reducing the overall computational cost. The scheme of the method is shown in Figure 1.2. After eventual pre-processing, at first spectral dimension is reduced through principal component analysis (§2.3) preserving most of the variability in the dataset. In fact, being hyperspectral images highly redundant, it is possible to capture more than 99% of the total data variability defining very few “new” variables, combinations of the spectral bands. Spatial extension is reduced through application of a wavelet transform (§2.4) up to the coarsest level that still allows materials detection. Then, different materials/objects are identified through classic clustering procedure (§2.5). After a labelled dataset is obtained, a classifier can be trained (§2.6) in order to classify not only the original image but also future possible images capturing the reflectance of the same materials under similar experimental conditions. Since the classifier is trained on a reduced “spectral” domain, generated by few linear combinations of the original bands, to apply the classifier on reflectance data it is necessary to project the data onto the directions defined by the PCA. For the specific hyperspectral data used to build the model, this operation obviously produces the same results obtained by PCA, considering only the first few components. For new data, the projection step must be applied before the classifier, but it is clearly less expensive than a complete principal component analysis procedure, since it involves projection onto very few directions. The assumption justifying this long procedure is that, nonetheless, the final computational cost is lower than the cost of performing a clustering operation on the whole dataset (§3.6).

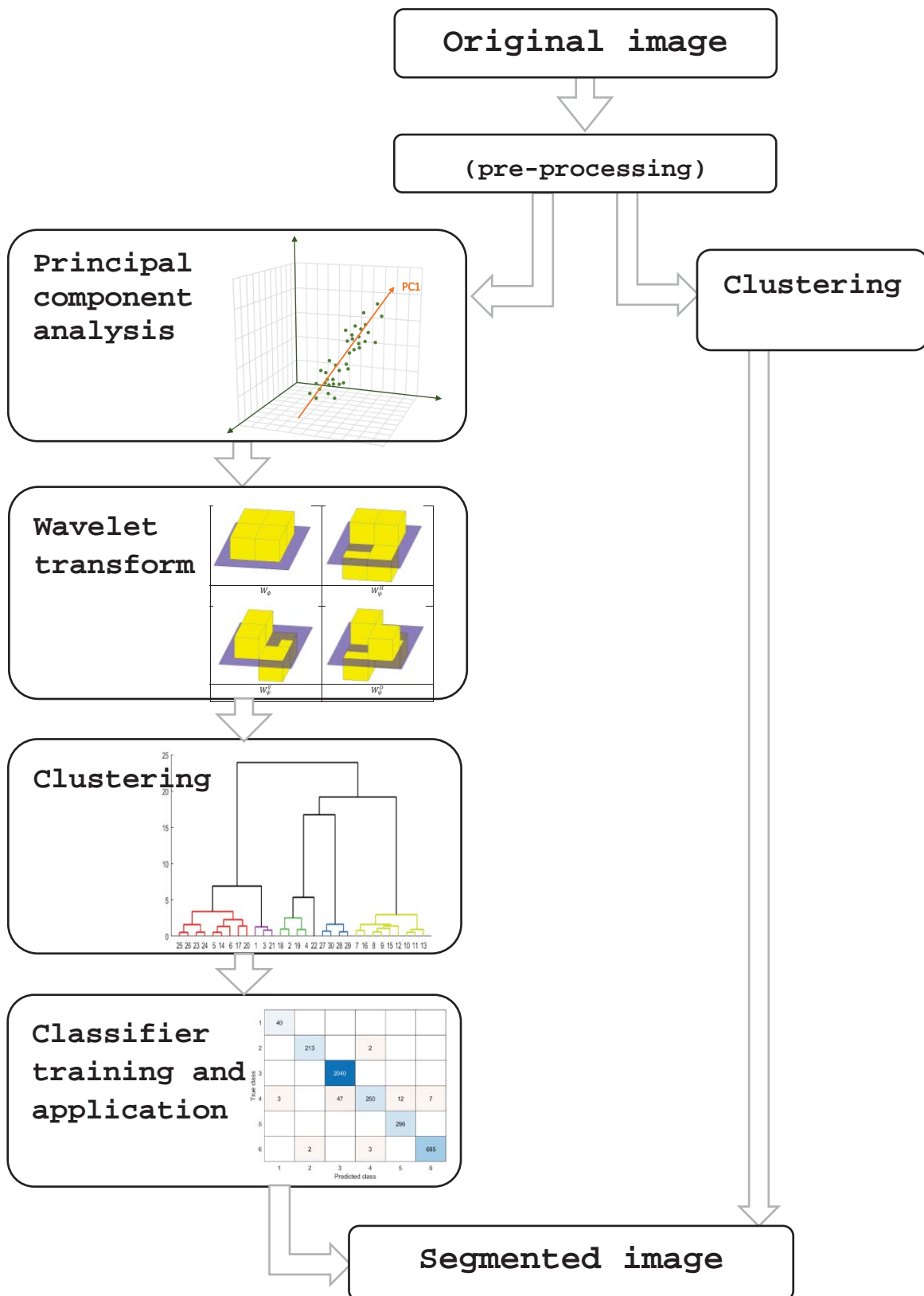


Figure 1.2 – Schematic illustration of the method proposed (left) as an alternative to direct clustering application (right).

Chapter 2

Methods

This Chapter presents a detailed theoretical treatment of all the techniques necessary to implement the procedure proposed for the segmentation of hyperspectral images.

2.1 Image acquisition

The hyperspectral image is acquired through the Specim FX17 camera⁴, as specified under the heading “Sensor Type” in the *SM Report*. The FX17 is a high speed Near Infrared⁵ detector with Indium-Gallium-Arsenide (InGaAs) alloy based photodiodes, designed for industrial applications, e.g. on-line quality control, inspection and process monitoring. The total spectral range detectable with the FX17 is from 900 nm to 1400 nm with maximum number of bands of 224. Full range detection speed is 670 fps, but if only specific regions are selected the detection speed increases. The maximum spatial sampling is 640 pixels. According to the *SM Report*, the full range available (number of Active Bands) and the maximum spatial sampling (number of Samples) were used, in order to store in the final image the maximum amount of information detectable with this specific model of hyperspectral camera. The *Scanner Measurement Report (SM Report)* in Appendix 2 – Camera report includes further information about the image acquisition device and the captured image itself.

⁴For more information:

<http://www.specim.fi/products/specim-fx17/> (last access: 07/2019)

<http://www.specim.fi/fx17-state-of-the-art-in-industrial-hyperspectral-imaging/> (last access: 07/2019)

⁵ The Near Infrared Spectrum conventionally goes from 700 nm to 1400 nm (Byrnes, 2009)

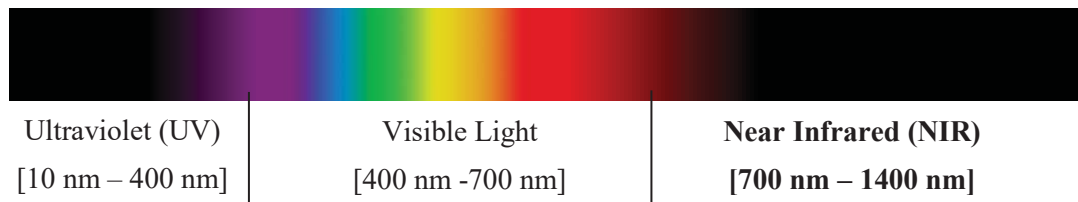


Figure 2.1 – Electromagnetic spectrum from UV rays to NIR radiation

2.2 Opening a hyperspectral image

Reading the information of a hyperspectral image may not be as straightforward as expected, especially if the image analyst and the operator acquiring the image are not the same person. The already mentioned *SM Report* in Appendix 2 – Camera report, contains all the information given to the image analyst. The most important for the analysis appears summarized in Table 2.1. The meaning of these denominations may not be completely clear from the point of view of the structure of the final picture, and the bit depth is unknown, but the problem may be addressed in a very simple way, applicable to the manipulation of pictures with partial information availability, as follows.

As previously stated, *Active Bands (AB)* is a common denomination for the different wavelength ranges (λ) acquired. *Frame Recorded (FR)* and *Samples (S)* are respectively the width and the height of the “picture” acquired for each wavelength range, having a bit depth (*bd*) of 16 bit. This last information may be obtained by trial and error: MATLAB allows to open an unknown file for binary read access with the function `fopen` and to read its content as a single column vector with the function `fread`. Therefore, the total length of this vector should be $S \cdot AB \cdot FR$ with an unknown bit depth, to be specified as a function parameter. Common values for the bit depth are 8 bit, 16 bit, 32 bit, 64 bit. The total amount of bits available in the binary file imported with `fopen` is exactly $S \cdot AB \cdot FR \cdot bd$, and its value can be obtained trying different bit depths in `fread`. Reshaping this 16-bit single column vector in a 3D matrix, more suitable for image analysis, can again be done by trial and error considering the six possible combinations of the three variables ($S \cdot AB \cdot FR$, $S \cdot FR \cdot AB$, $FR \cdot S \cdot AB$, $FR \cdot AB \cdot S$, $AB \cdot FR \cdot S$, $AB \cdot S \cdot FR$) and displaying the greyscale conversion of the picture corresponding to one of the λ -ranges acquired, verifying that the resulting displayed picture has a reasonable resemblance to the picture available in *png* format (Figure 2.3), capturing the objects reflectance in the visible spectrum.

It appears that, for this specific picture, the data was stored as follows: for each sample the intensity of the emission in one wavelength range is acquired for each frame recorded - therefore the reshaped matrix dimensions (rows, columns, depth) are (S , AB , FR). The greyscale conversion for one on the wavelengths acquired by the instrument is obtained by

rescaling the “slice” of the 3D matrix corresponding to the 150th wavelength band⁶ to the range [0,1], in order to obtain the maximum contrast between different objects captured in the picture for that λ range (Figure 2.4). As expected, the objects brightness is different from the one in the *png* picture (visible spectrum) since different materials reflect the selected wavelength range with different intensities.

Information made available online⁷ by the camera manufacturer confirms that the result obtained is reasonable since, if the camera is attached to a standard Specim lab scanner like the assembly shown in Figure 2.2, the scanning sensor collects full spectral information of a thin line of the object at a time, one pixel at a time.

This trial and error technique may be used whenever the information available is not sufficient.

Samples - S	Active bands – AB	Frame recorded - FR
640	224	480

Table 2.1 – Scanner measurement report

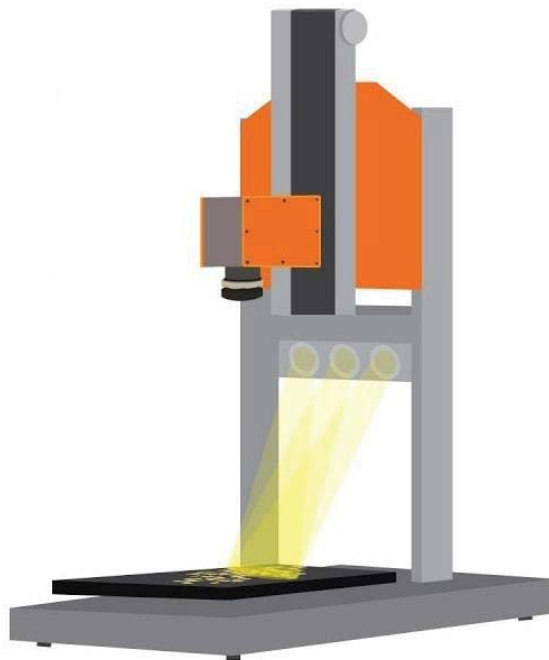


Figure 2.2 – Typical experimental assembly: Specim Lab Scanner 40x20 and Specim Hyperspectral Camera FX 17 (from the SpecimSpectral YouTube Channel)

⁶ The choice to consider a central band and not, for example, the first one or the last one is clarified in chapter 3, §3.1, and is related to lower detection efficiency at the extremes of the detectable range

⁷ YouTube Channel SpecimSpectral, official channel of the manufacturer

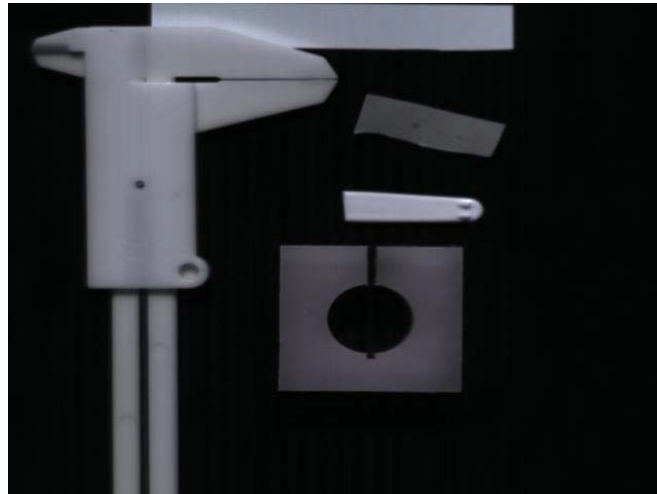


Figure 2.3 – Greyscale picture of the objects to be analysed in png format

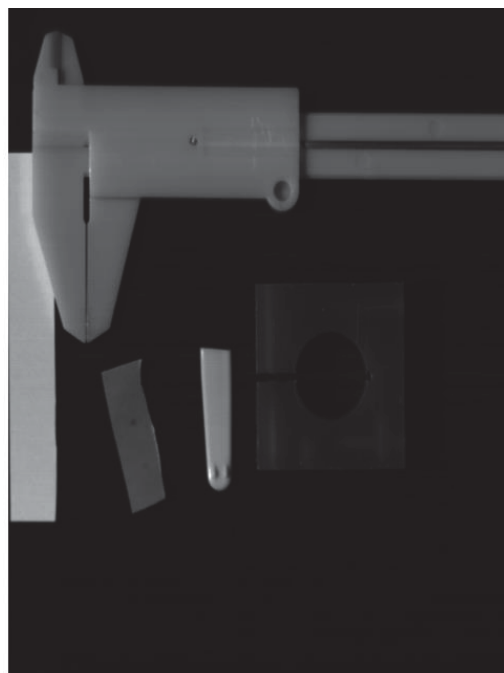


Figure 2.4 – Greyscale conversion of the 150th wavelength range acquired

2.3 Principal Component Analysis

2.3.1 General purpose

Instruments available to the industry are becoming increasingly efficient in collecting large amounts of data in a progressively shorter time (Wise *et al.*, 1996). This improvement in

technology has important consequences, the main one being that the essential information needed for the purpose is hidden in the huge pile of data collected. It might seem obvious that collecting more data means having more information available about the object of study, but the unfortunate truth is that “data” is different from “knowledge”. Therefore, the main problem is that of data dimension reduction and extraction of information, eliminating correlation and redundancies. Principal component analysis (PCA) is a statistical technique widely applied to handle this problem.

As seen in the previous paragraph, hyperspectral cameras can capture reflected electromagnetic radiation of a specified range of wavelength with great detail. Hyperspectral images thereby obtained can contain measurements for a significant number of bands – up to 224 for the FX17 employed for this study. Considering that the picture pic_i corresponding to a specified wavelength band can be unfolded into a single vector, connecting the end of each pixel line to the beginning of the subsequent one, a matrix containing the different pixels as rows and the different $pic(\lambda_i)$ as columns can be built. Figure 2.5 shows for example the unfolding process for N wavelength bands for a square picture of 16 pixels. Through this procedure the 3D hyperspectral image is rearranged into a standard 2-way table, where the set of *variables* (arranged in columns, corresponding to a different λ_i) are observed for each pixel (called *observation, arrange in lines*). This dataset can be statistically analyzed applying PCA.

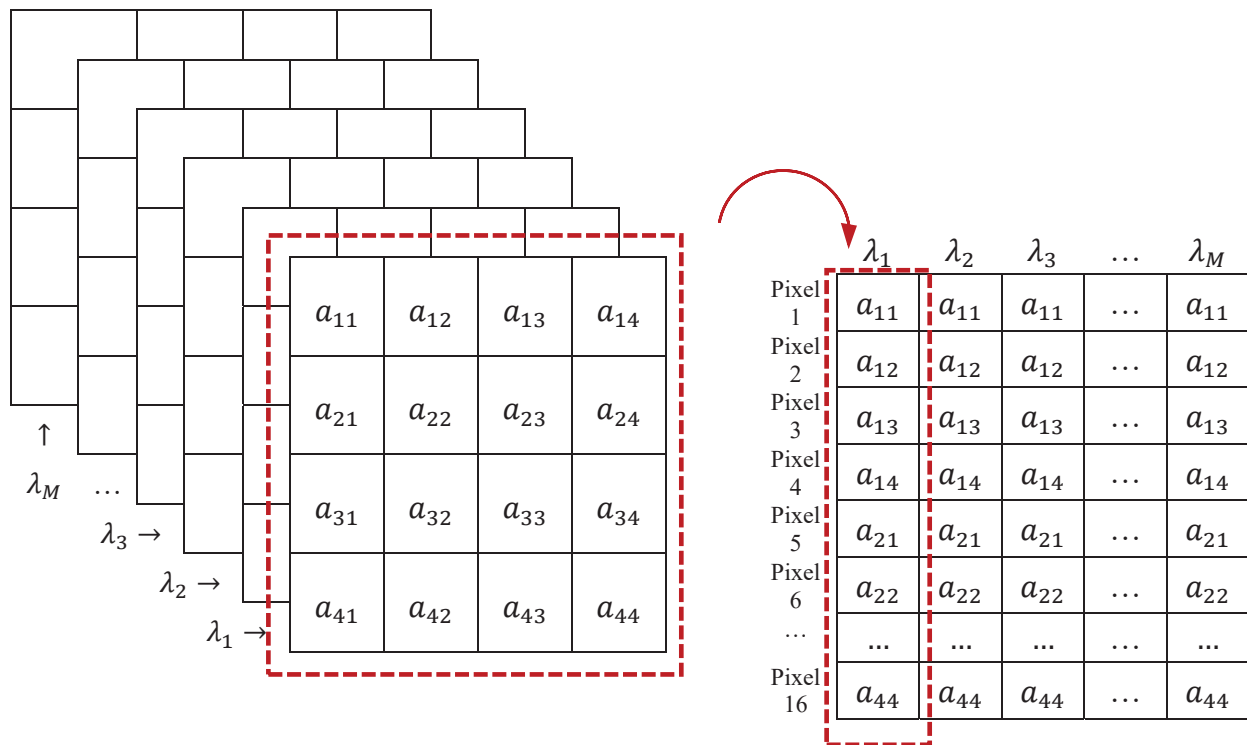


Figure 2.5 – Unfolding hyperspectral image into 2D matrix.

2.3.2 PCA Fundamentals⁸

Principal components are linear combinations of the *variables* ordered according to a criterion of quantity of information delivered by each one of the PCs, i.e. the first principal component retains most of the relevant information, from the second on the components optimize the residual information under the constraint of non-correlation (i.e., orthogonality) with respect to the other components.

For simplicity of visualization, let us consider three *variables* for which several *observations* are registered. The *variables* could be three bands of the electromagnetic spectrum – if the visible light is considered, they could be the RGB components in which usually color pictures are decomposed. The *observations* could be the pixel of the final picture, physically corresponding to a square area of the object seen from the camera angle. Plotting all the observations for the three variables could give a graph like the one in Figure 2.6, in which the observations clearly cluster in a specific area of the RGB space, i.e. the portrayed object color has different shades of a specific combination of red, green and blue. In other words, the color of the picture does not vary in the whole visible spectrum, but rather along a specific direction, i.e. specific linear combination of the variables, called principal component (PC1 in

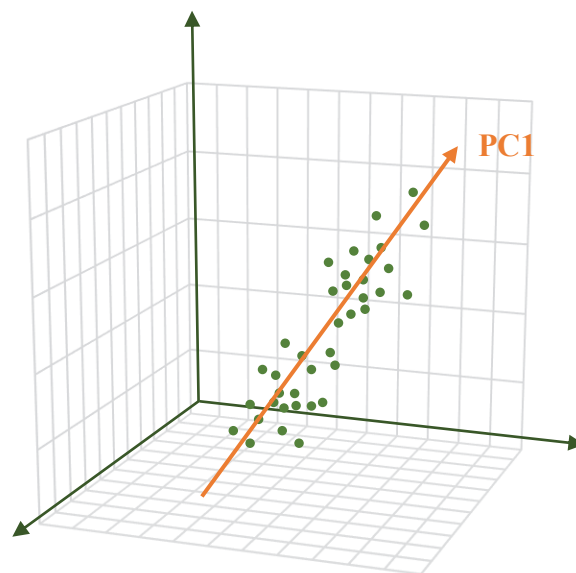


Figure 2.6 – Example of dataset for which variables are not completely independent

⁸ This paragraph presents some basic concepts of principal component analysis, with a focus on its application in image processing. For a deeper and more complete presentation of the topic please refer to Bro *et al.* (2006), Bro *et al.* (2003), Wise *et al.* (1996), Jackson (1991), on which this paragraph is based.

the figure). This means that, with some approximation, the original set of variables (RGB) can be replaced by only one variable, the PC1, retaining most of the information in the original picture, therefore reducing the dimensionality of the problem. If the dataset has to be further processed, its compression could have important effects on the computational cost. For example, on the figure, along the PC1, two clusters may be noticed by eye inspection. Reducing the dimension of the problem, a potential clustering operation for classification is going to be less expensive from the computational point of view.

2.3.2.1 Mathematical model

Let \mathbf{X} be the data matrix with *variables* as columns and *observations* as rows. The objective is to find the linear combination of all the variables (called PC1 or p^9) that maximizes the variance of the orthogonal projection of the observations on the PC1 itself. In symbols, being

$$\mathbf{t} = \mathbf{X}\mathbf{p} \quad \text{with } \|\mathbf{p}\| = 1 \text{ (i.e. } \mathbf{p}^T\mathbf{p} = 1) \quad (2.3.2-1)$$

the objective may be written as: finding \mathbf{p} such that

$$\max(\text{var}(\mathbf{t})) = \max(\mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p}) \quad (2.3.2-2)$$

Subject to the norm of \mathbf{p} be equal to 1 (\mathbf{p} is a vector of unit length) in order to have a bounded maximization problem (i.e. existence and uniqueness of the solution), otherwise the variance of \mathbf{t} could be made indefinitely small just multiplying \mathbf{p} by smaller and smaller constants. Notice also that $\text{var}(\mathbf{t}) = \mathbf{t}^T\mathbf{t} = \mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p}$.

The vector of the linear combination coefficients \mathbf{p} is usually referred to as the “loading vector” (or simply “loadings”), while \mathbf{t} is the “score vector” (or “s¹⁰cores”). Each row $\mathbf{t}(i)$ of the score vector is the result of the orthogonal projection of the corresponding row of \mathbf{X} (i.e. the corresponding *observation*), indicated as $\mathbf{X}(i, :)$, on the vector \mathbf{p} . In other words, $\mathbf{t}(i)$ is the score of the projection of the vector $\mathbf{X}(i, :)^T$ on the new basis vector \mathbf{p} .

Solving the maximization problem of equation (2.3.2-2) is equal to find the maximum eigenvalue λ_p of the square matrix $\mathbf{X}^T\mathbf{X}$ and its corresponding eigenvector \mathbf{p} . The matrix $\mathbf{X}^T\mathbf{X}$ is the variance-covariance matrix of \mathbf{X} if each column in the matrix \mathbf{X} is centered with respect to the corresponding column average (i.e. each variable is centered with respect to its average).

⁹ Some authors call “principal component” the pair (\mathbf{t}, \mathbf{p}) rather than just \mathbf{p} since the two vectors are closely tied together (Bro *et al.*, 2006)

¹⁰ Notice that by definition the covariance matrix is $\mathbf{X}^T\mathbf{X}/(N-1)$, being N the number of rows of \mathbf{X} (i.e. of observations), but multiplication by a constant does not change the eigenvectors and multiplies all the eigenvalues by the same constant $(N-1)$, not varying their descending order and therefore having no consequence on the principal components selection. For more details on eigenvectors calculation refer to any linear algebra textbook, like Hefferon (2017)

Considering the projections thus obtained, an approximation of the original dataset can be reconstructed as $\hat{\mathbf{X}} = \mathbf{t} \mathbf{p}^T$. Figure 2.7 (left) shows how the approximation would look like for the example considered in the introduction to section §2.3.2.

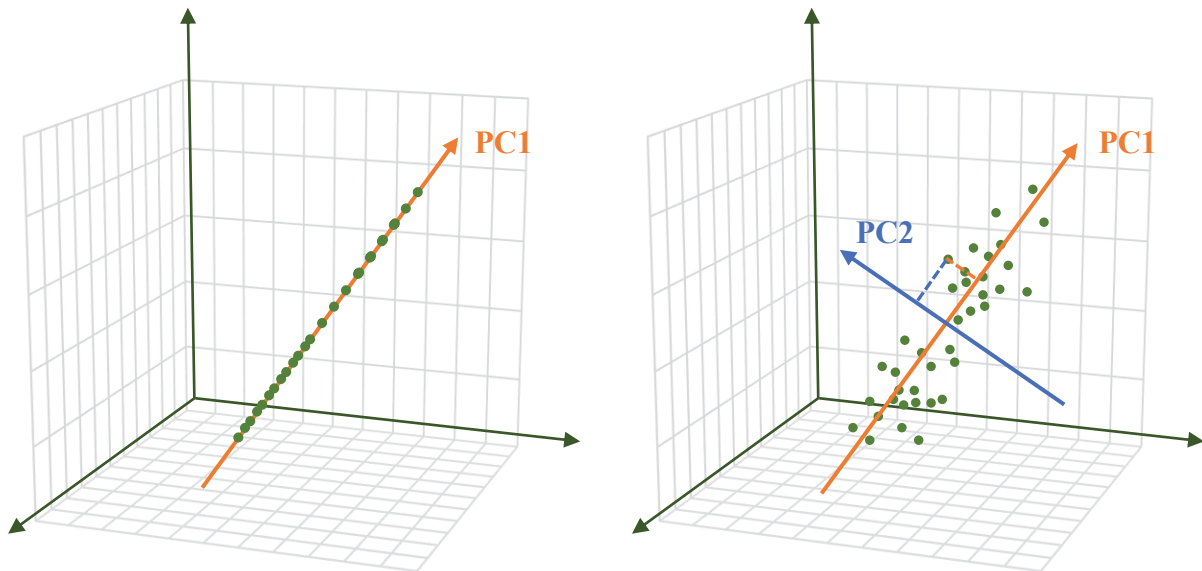


Figure 2.7 – (left) Approximation $\hat{\mathbf{X}}$ of the original dataset \mathbf{X} along the first principal component; (right) Second principal component and example of projection of a point

Nevertheless, it could happen that “compressing” all the information stored in the several variables into just one new variable does not represent the dataset satisfactorily. One or more other new variables (or latent variables), combination of the old ones, could be added, and projecting the data on each one of them a set of corresponding score vectors could be obtained. In order to add latent variables that maximizes the amount of additional information captured by each one of them, an important requirement is for them to be linearly independent from the other latent variables, i.e. they must be orthogonal to each other. In fact, eigenvectors of the covariance matrix $\mathbf{X}^T \mathbf{X}$ are orthogonal by definition, and ordering them according to the magnitude of the correspondent eigenvalue gives, for each new latent variable added, the vector able to capture the maximum residual variation of the data, that is the variation not captured by the latent variables already introduced. Considering more than one latent variable and the corresponding projections of the data on them (Figure 2.7, right), a better approximation of the matrix \mathbf{X} can be built as $\hat{\mathbf{X}} = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{t}_2 \mathbf{p}_2^T + \dots + \mathbf{t}_R \mathbf{p}_R^T$, being R the maximum number of latent variables that can be added (i.e. the rank of \mathbf{X}), and the original dataset can be written as $\mathbf{X} = \hat{\mathbf{X}} + \mathbf{E}$, where \mathbf{E} is the residual and goes to zero with the increase of the number of principal components added. Obviously, the approximation improves if more PC are added, but on the other hand considering a number of PC equal to

the number of the original variables is meaningless from the data compression point of view. The optimal number of PC strongly depends on the problem and there is no general rule always true, but just “rules of thumb” useful to get an idea of the status quo. Some of these techniques will be illustrated through their application in the next chapter, along with other numerical results, for simplicity of discussion.

Reduced the number of variables to only few relevant latent variables, classification could now be applied with significant lower computational burden. Nevertheless, segmentation remains one of the most expensive computational tasks to perform on a dataset, since it requires computing distances between all pairs of “points” (or clusters of points). In this case, each of these points corresponds to the score of a point in the picture on all the PCs retained. Therefore, the total number of points for which two by two distances must be calculated, grows exponentially with the number of points in the picture, and this is only the first step because at each iteration, when new clusters are formed by connecting two near points, the distances must be recalculated. Consequently, it is clear that compressing somehow the picture can produce significant benefits from the computational point of view. At the same time, it is important to remember that some information is inevitably lost with compression, therefore, when these techniques are applied, caution should be exercised to limit the compression up to a level that does not compromise the success of the classification operation.

Since the object of study is an image, data compression techniques specific for images (and signals in general) can be applied, one of them being the wavelet transform, described in the next section.

2.4 Wavelet Transform

2.4.1 General overview

As noted by Reis *et al.* (2009), a transform is nothing more than an alternative way of representing data. In fact, a signal – in this case the profile of the response intensity along the picture – can be decomposed into a combination of expansion functions each one multiplied by the corresponding expansion coefficient, being the set of functions a base if the coefficients are unique, as it is usually desirable. The suitability of a transform and/or of a set of basis functions depends on the scope of the analysis. For example, the Fourier transform has a set of basis functions that are periodic functions, well localized in the frequency domain, but unable to provide any form of insight in the time domain (or spatial domain in the case of a digital image). Therefore, the Fourier transform is appropriate for periodic phenomena, that

could be found in signal denoising (Reis *et al.*,2009) but also in image denoising, e.g. reducing periodic interference caused by malfunctioning imaging system, (Gonzalez *et al.*, 2010), since images could be seen as a signal that is a collection of intensity information for different wavelengths. Fourier transform could, in fact, be perfect for filtering in the frequency domain, but problems arise when the oscillation is not over all the domain, but it is localized on some portions of it, for example only on the background. An example of this situation is shown in Figure 2.8, obtained by increasing the brightness of the grayscale conversion of the image corresponding to the 150th band. Moreover, if the noise is not sufficiently regular and periodic, the Fourier transform could not be appropriate as well, as could happen for aerial photographs of lands where different natural materials must be identified and the irregular noise (in form of shadowing created by the irregular surfaces of natural objects themselves, like rocks or dirt) could be more difficult to be modelled and/or filtered properly in the frequency domain considering also the simultaneous presence of different objects with supposedly sharp edges. If a good localization in time (or space, in the case of a picture) is needed, other transforms are preferable. The simplest solution could be to “localize” the Fourier transform through a window function, applying the transform only to a portion of the signal at a time, covering all the domain (time or space) simply by shifting the window function. This approach is known as windowed Fourier transform or Gabor transform, from the name of the first scientist who proposed it. Basically, it divides the

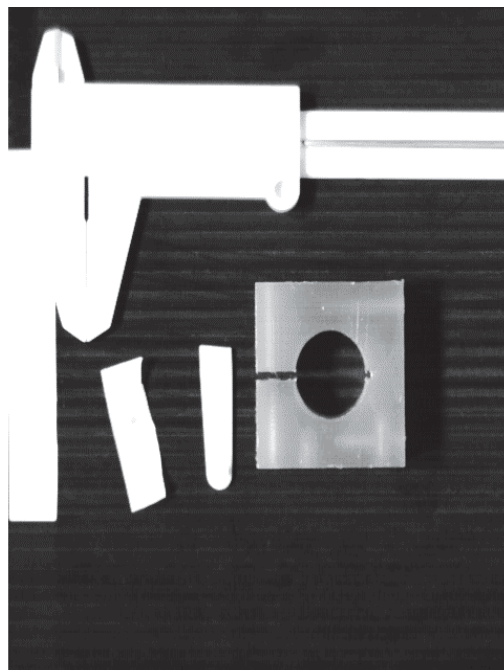


Figure 2.8 – Grayscale conversion of the picture corresponding to the 150th wavelength band. Brightness increased from 50% (reference value for the default brightness) to 80%.

time(space)-frequency plane into equal squares, and therefore the dimension of these square portions of the domain, being equal overall the plane, could be too small to capture properly low frequency features and too coarse to for the high frequency ones. For this reason, the windowed Fourier transform might still not perform well enough in some applications, even with the improved localization with respect to the normal Fourier transform.

The wavelet transform, on the other hand, can provide a good insight on both frequency and time (or space) characteristic of a signal (or image), not only using basis functions well localized in the time(space)-frequency plane, but also dividing the domain into smaller or coarser subdivision depending on the frequency. Being the objective of this study the analysis of an image, from now on only the space-frequency plane will be considered, on the understanding that all the mathematical description of the wavelet transform will not vary if “space” is substituted with “time” in the case of other kinds of signals.

2.4.2 The wavelet transform ¹¹

The wavelet transform results in an alternative representation of the data in the form of a lower resolution version of the original dataset plus details “removed” in lowering the resolution. The original data can then be reconstructed through a specific combination of the “approximation” and the “details”. This representation is obtained considering basis functions that are well localized in space and have the shape of “small waves”, hence the French *ondelette* and its English translation/transliteration *wavelet*. The wavelet transform may operate with real or complex wavelet functions. Complex wavelet transform may be useful to detect small details in pictures and oscillatory behavior (Torrence *et al.*, 1998), but at a higher computational cost. Since this effort is not needed for the picture under examination, where objects of significant size are pictured and there is no oscillation that needs to be described with precision, the following paragraphs will present the wavelet transform using real-valued expansion functions and real-valued expansion coefficients.

2.4.2.1 Wavelet transform in one dimension

The position in the space-frequency domain of a wavelet function depends on a scale parameter s and a translation parameter b , in such a way that each wavelet function can be described as a translation in space and scaling of a mother wavelet, in symbols:

$$\psi_{s,b} = 1/\sqrt{s} \psi((x - ks)/s) \quad (2.4.2-1)$$

¹¹ This paragraph presents some basic concepts about wavelet transform, with a focus on its application in image processing. For a deeper and more complete presentation of the topic please refer to Reis *et al.*, (2008), Reis *et al.*, (2009), Reis *et al.*, (2010), Gonzalez *et al.*, (2010), on which this paragraph is based.

where x is the special coordinate in the space-frequency domain.

Once the mother wavelet has been defined, the family of function obtained as mentioned is able to span the domain, but it is not a “basis” in a strict sense if the coefficients vary continuously and therefore cause a redundant representation. Consequently, the parameters must be discretized in a way that allows to cover the whole domain without redundancies, and this objective could be achieved imposing $s = 2^j$, being j an integer number. Applying this dyadic discretization to equation (2.4.2-1) gives:

$$\psi_{j,k} = 2^{-j/2} \psi(2^{-j} x - k) \quad (2.4.2-2)$$

One of the first wavelet ever introduced, and the simplest one, is the Haar wavelet:

$$\psi_{j,k}^{Haar}(x) = \begin{cases} 1 & k/2^{-j} \leq x < (k + 0.5)/2^{-j} \\ -1 & (k + 0.5)/2^{-j} \leq x < (k + 1)/2^{-j} \\ 0 & \text{otherwise} \end{cases}$$

the shape of which is represented in Figure 2.9 (left).

Increasing j , the mother wavelet is stretched and its height is shortened, therefore the level of detail that can be captured diminishes (Figure 2.9, right). Varying j , therefore, the entire frequency domain can be spanned. Notice that going to negative values of j the mother wavelet itself can be compressed and its height increased, improving the ability to capture details from the signal. Each one of the wavelet obtained by varying j can then be translated in space by changing the (integer) value of k to span all the spatial domain. In this way, a set of basis function is finally obtained. It is possible to expand any squared-integrable signal as a combination of wavelets of the same family, starting from a mother wavelet and equation (2.4.2-2).

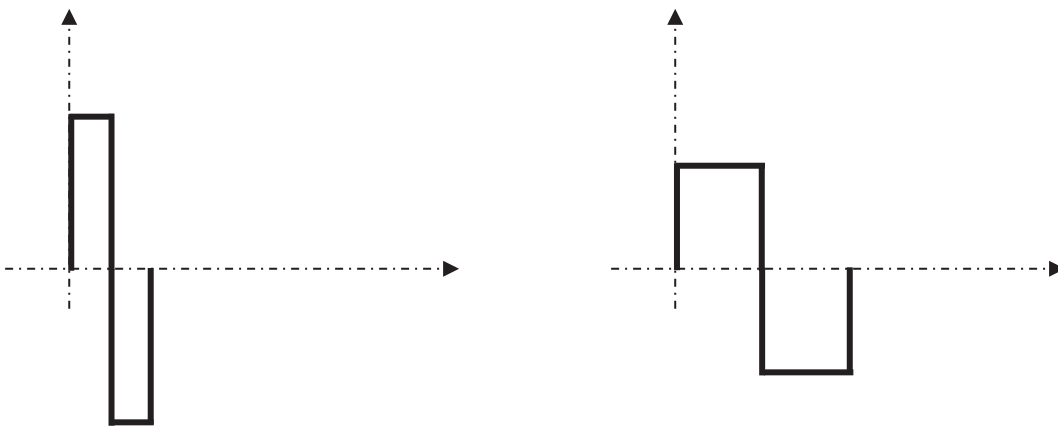


Figure 2.9 – Example of Haar wavelet: (left) mother wavelet; (right) wavelet obtained from the mother wavelet imposing $j = 1$ in equation (2.4.2-2)

Nevertheless, for a continuous irregular function (for example with random noise) in principle an infinite number of wavelet would be needed, to capture all the detail and be able to reconstruct the original signal afterwards. Therefore, the choice of introducing the “father wavelet” ϕ , or “scaling function”, corresponding to the mother wavelet (or to the wavelet with minimum value of j , if also negative values are considered):

$$\phi_{\mathcal{F},k}^{Haar}(x) = \begin{cases} 1 & k/2^{-j} \leq x < (k+1)/2^{-j} \\ 0 & \text{otherwise} \end{cases}$$

As an example, Figure 2.10 shows the qualitative shape of the Haar scaling function corresponding to the Haar wavelet in Figure 2.9 (left).

Considering only positive values for j , the maximum level of detail that can be captured is defined by the father wavelet and the mother wavelet. For a continuous signal, it is necessary to choose the precision in the detail. For a digital signal, like an image, since it is inherently discretized, the maximum level of detail could be chosen to be the signal itself. In fact, usually this is the starting assumption for the wavelet transformation of an image. The father wavelet and the family derived from the mother wavelet together constitute a set of basis function that can be linearly combined to represent the original dataset, in the form of the so-called wavelet expansion:

$$f_0(x) = \sum_k a_{\mathcal{F}}(k) \phi_{\mathcal{F},k}(x) + \sum_{j=1}^{\mathcal{F}} \sum_k d_j(k) \psi_{j,k}(x) \quad (2.4.2-3)$$

where \mathcal{F} is the decomposition depth, $a_{\mathcal{F}}(k)$ is the coefficient for the linear combination of the scaling functions ϕ and $d_j(k)$ is the coefficient for the wavelet functions. Notice that the father wavelet is translated to span all the spatial domain varying k , and a different coefficient corresponds to different translations. Notice also that the father wavelet can be stretched in the same way as the mother wavelet, in order to create new scaling functions.

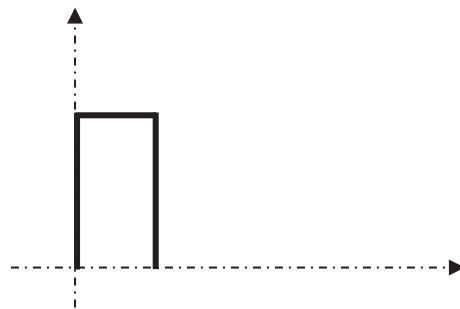


Figure 2.10 – Haar scaling function (or “father wavelet”)

After choosing a mother wavelet and a scaling function (e.g. the Haar ones), the coefficients in equation (2.4.2-3) can be evaluated. Knowing that the dyadic discretization allows to obtain an orthogonal set of wavelets, if the mother wavelet is also defined in such a way that, fixed j , the wavelets obtained with translation are also orthonormal, which means that

$$\langle \phi_{\mathcal{F},k}(x), \phi_{\mathcal{F},l}(x) \rangle = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$$

Then the coefficients can be evaluated as a simple inner product between the function and the wavelets $d_{j=fixed}(k) = \langle \psi_{j=fixed,k}(x), f_0(x) \rangle$. Similarly, for the scaling coefficients, if the $\phi_{\mathcal{F},k}(x)$ varying k are an orthonormal basis, $a_{\mathcal{F}}(k) = \langle \phi_{\mathcal{F},k}(x), f_0(x) \rangle$. The discrete wavelet transform presented in equation (2.4.2-3) can be therefore rewritten as,

$$f_0(x) = \frac{1}{\sqrt{N}} \mathbf{w}_{\phi}(\mathcal{F}, k) \phi_{\mathcal{F},k}(x) + \frac{1}{\sqrt{N}} \sum_{j=1}^{\mathcal{F}} \mathbf{w}_{\psi}(j, k) \psi_{j,k}(x) \quad (2.4.2-4)$$

with

$$\mathbf{w}_{\phi}(\mathcal{F}, k) = 1/\sqrt{N} \sum_x f(x) \phi_{\mathcal{F},k}(x)$$

$$\mathbf{w}_{\psi}(j, k) = 1/\sqrt{N} \sum_x f(x) \psi_{j,k}(x).$$

As mentioned, to different values of j correspond different levels of detail. Fixing the value of \mathcal{F} (that is the maximum detail) to 1 in equation (2.4.2-3) the dataset is expanded into its approximation $\sum_k a_1(k) \phi_{1,k}(x)$ and the detail lost during the approximation $\sum_k d_1(k) \psi_{1,k}(x)$. If another level of detail is subtracted from the approximation, a new approximation is obtained $\sum_k a_2(k) \phi_{2,k}(x)$ and an additional detail is loss $\sum_k d_2(k) \psi_{2,k}(x)$ and so on. A scheme of this procedure can be found in Figure 2.11. Being the $\phi_{\mathcal{F},k}$ and the

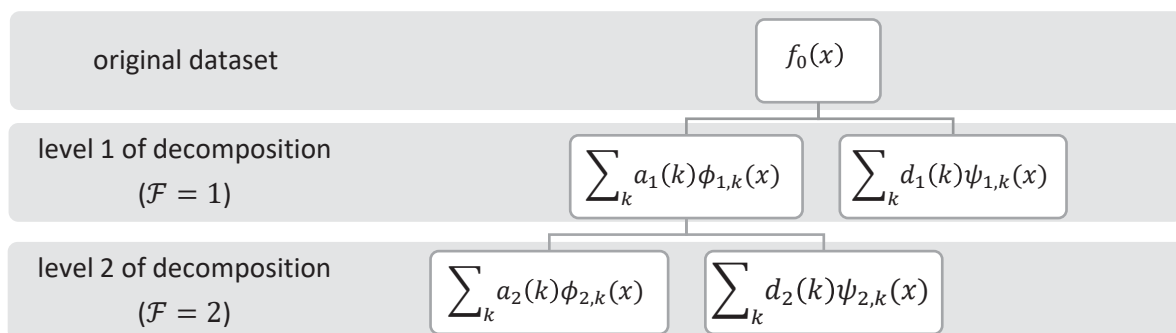


Figure 2.11 – Schematic representation of the wavelet expansion

$\psi_{j,k}$ derived from the mother wavelet and the father wavelet through the procedure in equation (2.4.2-2), for each step of the decomposition it is only required to retain the coefficients $a_{\mathcal{F}}(k)$ and $d_j(k)$. Since the approximated dataset is halved in dimension at each step and the detail coefficients to retain are equal in number to the approximation coefficient of each step, the dimension of the collection of data does not vary after the application of the wavelet transform, if both detail and approximation coefficients are retained.

2.4.2.2 Wavelet transform in two dimensions

Being a hyperspectral image the result of a concatenation of images captured for different wavelengths, each of these “slices” is a dataset that must be transformed independently to reduce its size to a fixed scale, equal for all the bands. Each image is a two-dimensional dataset, therefore to adequately approximate it the transform must be performed in a bidimensional way. One possibility is to use the tensor product of two unidimensional basis functions, to derive the new bidimensional basis set:

$$\psi(x, y) = \phi(x)\phi(y) \quad (2.4.2-5)$$

Now, the details lost along columns (V =vertical), rows (H =horizontal) and diagonally (D) are captured by three bidimensional wavelets that are “directionally sensitive” and separable, giving three “detail images”

$$\psi^H(x, y) = \psi(x)\phi(y) \quad (2.4.2-6) \text{ (a)}$$

$$\psi^V(x, y) = \phi(x)\psi(y) \quad (b)$$

$$\psi^D(x, y) = \psi(x)\psi(y) \quad (c)$$

The discrete wavelet transform for the image is therefore

$$f(x, y) = \frac{1}{N} \sum_m \sum_n \mathbf{W}_\phi(\mathcal{F}, m, n) \phi_{\mathcal{F},m,n}(x, y) \quad (2.4.2-7)$$

$$+ \sum_{i=H,V,D} \sum_{j=1}^{\mathcal{F}} \sum_m \sum_n \mathbf{W}_\psi^i(j, m, n) \psi_{j,m,n}^i(x, y)$$

Where m, n are equivalent to k and are related to spatial translation, and W_ϕ and W_ψ^i can be expressed as

$$\mathbf{W}_\phi(\mathcal{F}, m, n) = 1/N \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \phi_{\mathcal{F}, m, n}(x, y)$$

$$\mathbf{W}_\psi^i(j, m, n) = 1/N \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \psi_{j, m, n}^i(x, y) \quad \text{with } i = \{H, V, D\}$$

In literature, the approximation matrix $\mathbf{W}_\phi(\mathcal{F}, m, n)$ may be indicated also as $a(\mathcal{F})$, while the detail matrices $\mathbf{W}_\psi^i(j, m, n)$ may be indicated as $d_1(j)$, $d_2(j)$, $d_3(j)$, for simplicity of notation.

Figure 2.12 shows the bidimensional Haar scaling function and wavelet functions as an example. Notice how they can be obtained from the one dimensional Haar scaling function and wavelet from Figure 2.10 and Figure 2.9 (left) through equations (2.4.2-5) and (2.4.2-6) (a).

Historically, several different scaling/wavelet functions have been proposed, each one with specific properties. The choice of the appropriate scaling function does not depend strictly on the application in a general sense (sound processing, image processing, etc.), but rather it is related to the shape of the dataset to fit and approximate and to the scope of the transformation (Ngui *et al.*, 2013). The wavelet function should reflect the feature of the signal, like sharpness or smoothness, for which is respectively more appropriate a box-like

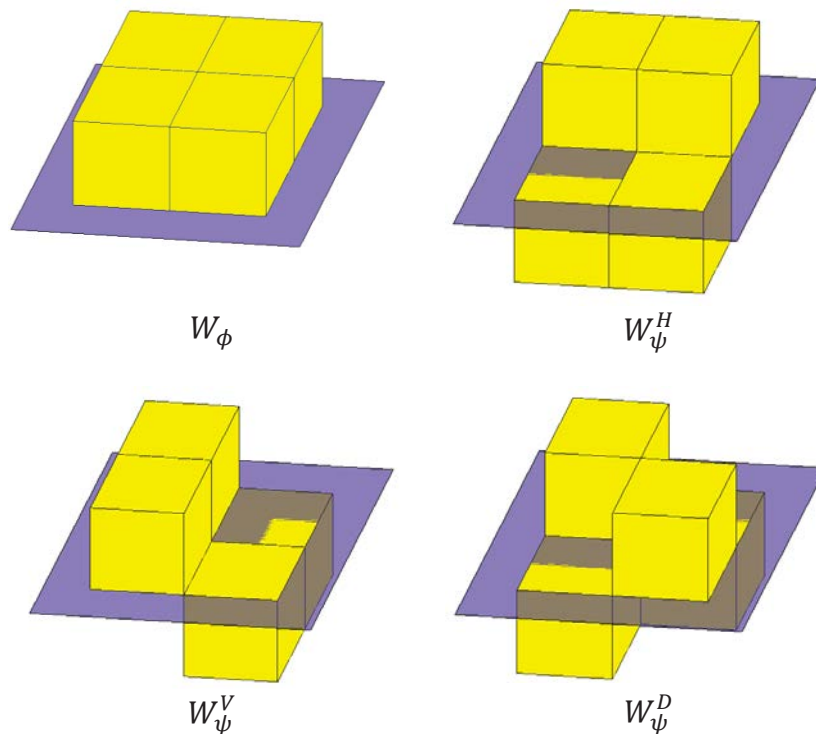


Figure 2.12 – Bidimensional Haar scaling function and wavelets.

function (e.g. Haar) or a smooth function (Torrence *et al.*, 1998). In this way the number of significant coefficients should be reduced, i.e. a perfect reconstruction of the signal should be obtainable with few decomposition levels (Megahed, *et al.*, 2008). Another important aspect is the final objective. In fact, in application where the gradual variation is not of interest, and effective edge detection is more important, Haar transform could be a good choice even for smooth varying datasets¹². This simple transforms also mitigates boundary effects, given its compact support. The image under examination in this study has some objects well distinct from the background; the edges are partially smoothed by the shadowing resulting from an angled light source (see following paragraph §3.1 for more information about the raw data), and it is important to not increase this blur to preserve the intensity values (corresponding to each material) for most of the points of each object, if possible, to improve classification; there is no oscillation needing to be detected, on the contrary oscillations of intensities on objects surfaces are just a noise from the point of view of the objective, i.e. distinguishing the materials. Therefore, the Haar wavelet, which happens also to be the simplest one, should be more than enough for the scope, on the understanding that the quality of results is the only criterion truly important in the selection of the wavelet (Ngui *et al.*, 2013).

A closer look to equation (2.4.2-7) reveals that the dimension parameter is only one, even if the dataset is two-dimensional. This is because equation (2.4.2-7) is valid only for a square image, since usually, to match the dyadic nature of the decomposition, the transform is applied square images of dimension N equal to a power of 2. Observing Figure 2.4, it appears clear that in the lower part of the image no useful information is stored, being mostly just background, therefore before wavelet analysis the image can be cut into a square. Remembering that the dimension of the data does not change after wavelet analysis, If all the coefficients are stored, the results of the analysis can be summarized in a square image like the one in Figure 2.13 ($\mathcal{F} = 1$) where the upper left corner contains the final approximation,

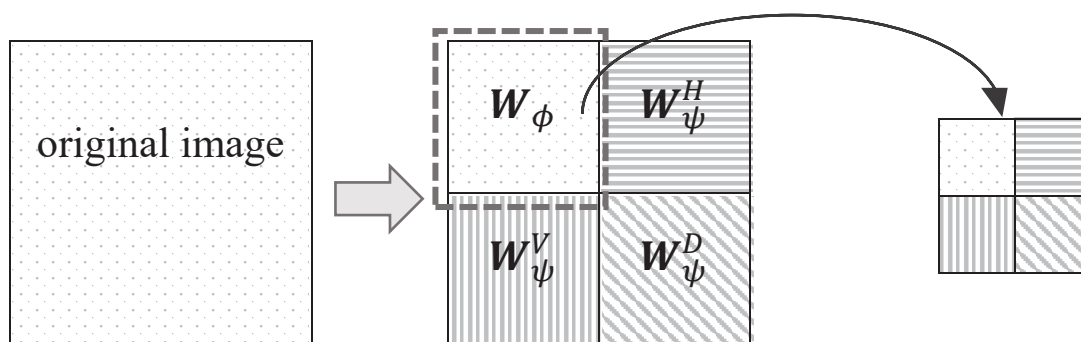


Figure 2.13 – Decomposition resulting after wavelet transform

¹² Refer to official MATLAB® online documentation for an overview about the main features of each mother wavelet (<https://it.mathworks.com/help/wavelet/gs/choose-a-wavelet.html>, last access: 07/2019)

while the other three squares contain details about the three directions of the decomposition (horizontal, vertical, diagonal). Applying again the decomposition to the approximation in the left corner (i.e. choosing \mathcal{F} greater than 1) the image can be decomposed up to the desired level of detail, and again information has the same dimension of the transformed image, that in this case is the first level of approximation.

As mentioned before, the original size of the data is preserved if all the detail coefficients are retained. Therefore, the advantage in using this decomposition, if the details are not discarded, lies in the (usually) high regularity of the linear combination of the details, that could be modelled and compressed, allowing to reconstruct the original picture in a sufficiently accurate way through the inversion of the transform using modelled details instead of the original ones, obtaining a standard for data compression. Other uses could be multiresolution analysis (for more information see Reis *et al.*, 2009).

For the case under examination, the objective is reducing the image size for the purpose of performing clustering efficiently, possibly in a way that diminishes also noise. Simple downsampling of the image points is not a preferable solution since the retained points could not be able to represent adequately the area from where they were taken, for example because of noise or because of smooth object borders. Therefore, some filtering is needed, and the wavelet transform can be seen as a series of filtering and downsampling operations: high-pass filters to obtain the approximations, low-pass filters to obtain the details, followed by dyadic downsampling. For this operation, an efficient procedure consisting of the application of quadrature mirror filters and dyadic downsampling was proposed by Mallat (1989), with a final computational cost comparable to the size of the dataset (order $\mathcal{O}(n)$ where n is the total number of points). The picture under examination contains objects of large size, therefore a very high resolution could be unnecessary for the classification purpose. For classification through hierarchical clustering, in fact, pairwise distance between points must be computed, and the cost of this operation grows with the number of points itself (being generally at least of order $\mathcal{O}(n^2)$, depending on the algorithm used). Therefore a smaller dataset could improve significantly the performance of this operation in terms of required time, without necessarily affect the quality of the results. Since wavelet transform produces an approximated, lower-size image filtering also noise contributions, it appears to be an appropriate choice for the purpose.

As a final note, it should be pointed out how the proposed approach is not the only solution to reduce the size of the dataset. In fact, the pre-processing could have involved a step of selection of a subset of the wavelength bands, but, due to the significant number of bands and the continuous nature of the electromagnetic spectrum, selection of one waveband instead of another would be questionable. Moreover, to have a size reduction comparable with PCA, the number of wavebands selected should not be more than the number of principal components retained, that is in general very low. But the capability to discern different materials would be

significantly worsened in the case of wavebands selection if, conversely, the principal components selected would sum up the information delivered by several bands. Therefore, for this application, feature extraction through principal component analysis was considered to be more appropriate. In addition, also reduction of the length and width of the image could have been performed differently. In fact, instead of performing wavelet analysis, an alternative approach could have been subsampling the image by simply picking a subset of the total points (e.g. drawing a mesh on the image and picking the points at each mesh intersection), but this simple approach brings up the problem of the representativeness of the selected points, that could be, by chance, just random noise. In fact, the mesh could be built in such a way that, accidentally, the periodic sampling corresponds exactly to a periodic noise, therefore all the points sampled contain noise-corrupted information. The smoothing action of wavelet transform prevents these kinds of problems.

2.5 Clustering

2.5.1 Introduction

Clustering is one of the techniques used for information extraction in the field of “data mining”, or “data science”. The objective is trying to obtain knowledge about the structure of the data itself. Clustering is a descriptive method, i.e. it is a tool meant to extract information specifically about the studied dataset, and not to make predictions. In particular, it aims to find the natural groups (or “clusters”) in data, depending on a measure of similarity/dissimilarity between data points. Since, in principle, nothing is known a priori about the group structure, this method is classified as an “unsupervised” machine learning technique, i.e. a class of methods able to extract information from data with little to no information available. This can be an advantage in the sense that these methods require almost nothing more than the dataset itself, but the other side of the coin is that there is no easy way (and sometimes no way at all) to judge the quality of the results, especially if also the quality check has to be performed by artificial intelligence. Being “unsupervised”, clustering should reveal the natural grouping of the points in the multidimensional space that is the dataset in the case of a hyperspectral image, where the proximity between points should be related to a similarity between the material constituting the portion of object to which that point belongs. At the end of the procedure, it is desirable that a label corresponding to each cluster is created and all the points of an object are correctly labelled with the corresponding cluster index.

2.5.2 Clustering fundamentals and methods¹³

In image analysis, clustering can be employed as a segmentation method in which the input is an image and the outputs are feature/attributes extracted from the image, with the aim of subdividing the image into its “constituent regions or objects” (Gonzalez *et al.*, 2010), each region being considered homogeneous with respect to some “image property of interest” (Jain *et al.*, 1995). Despite their very specific application in image analysis, clustering techniques are non-specific in their core characteristics, and adapting to the specific application the definitions of the variable quantities involved in the algorithm, the same algorithm can be used for a wide variety of problems, going from flower species unsupervised classification based on different lengths and widths of sepals and petals (the famous Fisher’s Iris dataset¹⁴) to automatic storage and retrieval of documents based on the subject, with each documents being labelled with a series of topic information and with the necessity to group together books of similar subjects¹⁵ (Rasmussen, 1992).

2.5.2.1 Introductory definitions and stages

Before describing the steps necessary to perform a clustering task, some definitions are necessary. An *observation* corresponds to a specific point/subject for which several *attributes* (or features, or variables) are measured. Notice that this nomenclature is similar to the one used for principal component analysis (section §2.3.2). A *distance measure* is a metric (or quasi-metric) used to assess the similarity/dissimilarity between observations, on the base of different values of the attributes. Notice how, if the attributes are quantitative and continuous, this metric can be the simple Euclidean distance between observations in the variables space (e.g. the distance between the points in Figure 2.6). The definition of a proximity (or distance) measure must be appropriate to the data domain, since obviously, for example, for qualitative attributes Euclidean distance cannot be applied.

Typically, a clustering algorithm involves the following steps (Jain *et al.*, 1999): pre-processing, definition of a distance measure, clustering, data abstraction (if needed) and assessment of the output (if needed). Pre-processing may involve the selection of a subset of the measured attributes, picking the ones able to capture most of the difference between the

¹³ This paragraph presents some basic concepts about clustering, with a focus on its application in image processing. For a deeper and more complete presentation of the topic please refer to Jain *et al.*, (1999), Gonzalez *et al.*, (2010), on which this paragraph is based.

¹⁴ Refer to https://en.wikipedia.org/wiki/Iris_flower_data_set (last access: 07/2019) for more information. Notice that this dataset already contains the information about species classification, therefore the groups obtained with clustering algorithms can be compared to the true species classification available in the dataset. In general, when applying clustering, no information is known about the true labels, hence the adjective “unsupervised” to describe this machine learning technique.

¹⁵ Notice how for this application defining the concept of similarity is quite difficult, being the features qualitative and unordered.

groups “intrinsic” in the dataset, and/or a feature extraction, converting the initial set of variables into a new (possibly smaller) set of variables more adequate for clustering operations. After defining the distance measure, grouping can be performed with several techniques. These methods can differ for the structure of the output, that can be a tree showing a hierarchy of clusters at different levels (hierarchical clustering), or it can be just a single partition of the dataset in a predefined number of groups (partitional clustering – e.g. k-means). Moreover, for each observation the output can be a single class (hard clustering) or a “degree of membership” distribution (fuzzy clustering). Clustering results may need to undergo an abstraction operation to obtain a representation that is easier to interpret for the human eye. And finally, an (optional) assessment of the output concludes the clustering operations, checking if the results are reasonable through statistical techniques.

2.5.2.2 Clustering techniques and similarity measures

As mentioned above, several clustering techniques exist, with different specific characteristics. Hereafter some of the most popular algorithms will be described and compared, and general similarity definition for points and clusters will be introduced.

Hierarchical algorithms. Among the most common, hierarchical clustering follows the most conceptually straightforward algorithm from the point of view of human reasoning, since it operates merging together the two nearest observations at each step, as a human would group items one step at a time on the basis of similarity between it and other items. More rigorously, the algorithm for agglomerative hierarchical clustering can be schematized as follows (Jain *et al.*, 1999):

Hierarchical clustering algorithm

Step 1 – Evaluate the distance matrix containing the distances between each pair of points. Each observation is considered a single “cluster” at the beginning of the procedure;

Step 2 – Find the two most similar observations/clusters (the two “nearest” ones, according to the metric defined) and group them into one bigger cluster and update the distance matrix;

Step 3 – Repeat *Step 2* until there is only one big cluster containing all the data points, then stop.

Since at each step of the algorithm two points/clusters are merged into one cluster, a binary cluster tree is generated, and keeping track of all the merging operations and the distances between the two objects merged at each step, it is possible to perform a “data abstraction” of

the clustering operations, i.e. to extract a compact representation of the results producing a dendrogram. Conventionally, the horizontal axis in a dendrogram represents the indices of the observations, linked vertically with bridges the height of which is a measure of the distance between clusters/observations (i.e. higher bridges indicate further clusters). Notice that the cluster tree can be cut at any level to obtain the desired number of clusters, that can be decided through specific diagnosis statistics of clusters coherence, or even just by eye inspection of the dendrogram itself – for example if the last and the second-to-last clusters group objects very far from one another, three groups may be the best choice, as shown in Figure 2.14.

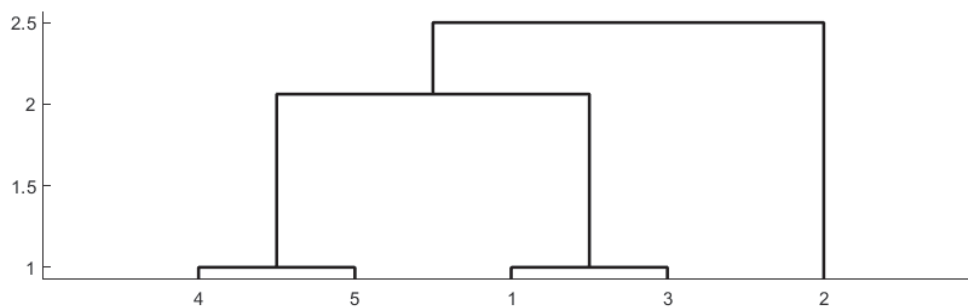


Figure 2.14 – Example of dendrogram

Among the statistics available, the *cophenetic correlation coefficient* is one of the most popular. As mentioned, the height of the bridge linking two clusters represents the distance between these two clusters, also known as *cophenetic distance* between the objects contained in those two clusters. The cophenetic coefficient compares this “cophenetic” distance to the “true” original distance between the two objects in the dataset, i.e. the distance evaluated at Step 1 of the hierarchical clustering algorithm, and quantifies their correlation, since if the clustering operation gives valid results, not occurred by chance, the cophenetic distance and the actual distance should be strongly correlated. The cophenetic coefficient can be evaluated as:

$$c_h = \frac{\sum_{x_2 > x_1} (d(x_1, x_2) - \bar{d})(c(x_1, x_2) - \bar{c})}{\sqrt{\sum_{x_2 > x_1} (d(x_1, x_2) - \bar{d})^2 \sum_{x_2 > x_1} (c(x_1, x_2) - \bar{c})^2}}$$

Where $d(x_1, x_2)$ is the distance between the two observations x_1 and x_2 , \bar{d} the overall average distance, $c(x_1, x_2)$ the cophenetic distance between x_1 and x_2 , \bar{c} its overall average. For a meaningful output the cophenetic coefficient should be close to 1, that is the cophenetic distance and the true distance should be highly correlated. In practice, the cophenetic coefficient can be used either to assess the quality of the results or to decide where to “cut” the dendrogram, obtaining a certain number of meaningful clusters.

Partitional algorithms. Another popular class of algorithms is the partitional algorithms, among which the k-means is the most common. From the point of view of the results, the most pronounced difference between a hierarchical algorithm and a partitional one is that the latter do not produce a cluster tree that can be cut, afterwards, at the desired level to obtain the more appropriate grouping. For partitional algorithms, the number of clusters has to be decided before starting the clustering task, with obvious consequences in the absence of even an estimate of the possible number of natural groupings in the data. A huge advantage of these methods is the quickness and the lower memory requirement, since it does not operate on the distances but on the data itself. Most commonly, the algorithm follows a scheme in the form of (Jain *et al.*, 1999):

Partitional clustering algorithm

Step 1 – Decide the initial partition of the data, fixing the centre of each group;

Step 2 – Assign each observation to the closest cluster centre and compute new centroids;

Step 3 – Repeat step 2 until stable grouping is reached, i.e. no points (or only very few of them) change group during new iterations

Notice that usually, the first step is randomized, and then results obtained with different initial random partitions are compared, to be sure that the algorithm did not converge to a local minimum of the function quantifying the overall distances from cluster centres (that could be for example a squared error function). Since for each point only the distance from the k cluster centres has to be calculated at each step, for a finite number of steps (usually low), the overall number of elementary operations required by a partitional clustering is less than that necessary for hierarchical clustering, where all pairwise distances have to be calculated, and some of them need to be updated at each step. This can be an advantage especially when clustering large datasets. The computational complexity (or time complexity) is commonly estimated by counting the number of elementary steps. In the case of a k-means partitional algorithm, the most common one, for each of the n observations, k distances from each cluster centre have to be calculated for each of the l total steps of the method, therefore the computational complexity is $\mathcal{O}(nkl)$. Usually, the number of iterations is fixed, as well as the number of clusters, therefore the dependency on the number of points is linear (Jain *et al.*, 1999). Hierarchical clustering, conversely, requires the computation of the pairwise distance matrix, counting $\mathcal{O}(n^2)$ comparisons (actually $n(n-1)/2$, because the distance matrix is symmetric), for n times, since there are n total merging operation after which the pairwise distances have to be updated. Therefore, if the algorithm is implemented straightforwardly, the time complexity is $\mathcal{O}(n^3)$, but efficient algorithm implementations using a heap to store

the distances (Kurita 1991) can reduce the time required to $\mathcal{O}(n^2 \log n)$, that is still $n \log n$ times longer than the time required by the k-means algorithm.

Nevertheless, hierarchical clustering is more rigorous and has the advantage of high versatility, due to the possibility to choose the number of clusters after the grouping tree is built. Moreover, k-means algorithm is highly sensitive to the positioning of the initial cluster centres but has the advantage that new data points can be added to the groups even after the main clustering operation was completed¹⁶, simply computing the distance from the final centroids, and also corrections to the final grouping can be evaluated afterwards, while for hierarchical clustering an erroneous classification is impossible to correct a posteriori, being deeply embedded in the tree. Nonetheless, the existence of a tree structure allows the definition of a cluster belonging criterion (*linkage*) different and independent from the distance between points, that is defined by the distance metric. In conclusion, the best algorithm depends on the specific problem, but if the size of the dataset is small enough, hierarchical method might be preferable for its rigorousness and flexibility, producing results of better quality.

Proximity measure. Independently on the method chosen, before performing the clustering it is necessary to define the meaning of “similar/dissimilar”, i.e. the distance metric. Different definitions of similarity between objects are available. For example, MATLAB offers several options¹⁷ among which even the possibility to set a custom defined metric. The most common metric is the *Euclidean distance*, defined as

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_m (x_{i,m} - x_{j,m})^2 \right)^{1/2} = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (2.5.2-1)$$

Where \mathbf{x}_i and \mathbf{x}_j represent two observations and $\|\bullet\|_2$ represents the Euclidean norm, that is the length of the vector “inside” the norm operator (in this case the distance vector). The Euclidean distance is a special case of the *Minkowski distance*:

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_m (x_{i,m} - x_{j,m})^p \right)^{1/p} = \|\mathbf{x}_i - \mathbf{x}_j\|_p$$

Other options of metrics give different weights to the different variables measured for each observation (i.e. each $x_{i,k}$ varying k), and these different weights can be the observations variances/covariances, the standard deviation, or other user defined weights. More elaborate metrics exist for more complex cases.

¹⁶ See section “Assign New Data to Existing Clusters and Generate C/C++ Code” on official MATLAB® documentation for more information (<https://it.mathworks.com/help/stats/kmeans.html>, last access: 07/2019)

¹⁷ See official MATLAB® documentation on <https://it.mathworks.com/help/stats/pdist.html> (last access: 07/2019) for more information

The Euclidean distance has the advantage of being very intuitive, and it works well if the clusters have a more compact structure (Mao *et al.*, 1996), i.e. they are hyperellipsoids in the hyperspace representing the observation points. In addition, if the data is not normalized, variables of larger scale tend to dominate (since they have higher influence on determining the final value of the norm, as can be deduced from equation (2.5.2-1)), and as a consequence some normalization in the pre-processing might be helpful. If this normalization was not performed, it can be addressed through the definition of a metric involving scaling functions, like standard deviation or even custom-built weight vector. On the other hand, using covariance to weight different variables measurements for an observation point can solve problems like linear correlation between variables, if present and not already targeted in the pre-processing (e.g. through PCA). In conclusion, each metric has strengths and weaknesses, and the choice of the best proximity measure definition depends largely on the properties of the dataset to be clustered, as happens for the choice clustering method.

Hierarchical clustering implementation in MATLAB allows also to choose also the definition of “distance” between two clusters, or *linkage*. There is no possibility of defining a custom function, but several possibilities are available. The conventional distance between two clusters can be taken to be the smallest distance between the points in the two clusters (*single linkage*)

$$d(c1, c2) = \min \left(\text{dist}(x_{c1,m}, x_{c2,l}) \right)$$

with $m \in (1, \dots, n_r), l \in (1, \dots, n_s)$

the largest (*complete linkage*)

$$d(c1, c2) = \max \left(\text{dist}(x_{c1,m}, x_{c2,l}) \right)$$

with $m \in (1, \dots, n_r), l \in (1, \dots, n_s)$

or the average (*average linkage*)

$$d(c1, c2) = \frac{1}{n_{c1}n_{c2}} \sum_{n_{c1}} \sum_{n_{c2}} \text{dist}(x_{c1,m}, x_{c2,l}).$$

It can also be defined by special functions like the *Ward's linkage* that considers the increase in some error metric resulting from the merging of two clusters; in MATLAB the formula used is:

$$d(c1, c2) = \sqrt{\frac{2n_{c1}n_{c2}}{n_{c1} + n_{c2}}} \|\bar{x}_{c1} - \bar{x}_{c2}\|_2$$

Where $c1$ and $c2$ are two clusters, n_{c1} and n_{c2} is the number of points respectively in $c1$ and $c2$, and \bar{x}_{c1} and \bar{x}_{c2} are the centroids of $c1$ and $c2$, computed as $\bar{x}_{ci} = (1/n_{ci}) \sum_m x_{ci,m}$. As already mentioned, $\|\cdot\|_2$ represents the Euclidean norm.

Again, the best choice depends on the problem under examination, and even if some logical reasoning can exclude a priori some metrics or linkages restricting the number of options, trying the remaining options and checking the quality of the results for the specific dataset might be the best strategy.

In conclusion, machine learning and data mining are very active research fields, and in literature several methods, with variations and improvements, are available, therefore for difficult problems it may be useful to search for more complex algorithms in the most recent scientific publications.

For the case under examination, the original dataset, that is the hyperspectral image, is huge, and consequently application of hierarchical clustering techniques is prohibitively expensive from the computational point of view. For this reason, reduction of the dimension of the dataset is vital, and the previous mentioned techniques of principal component analysis and wavelet transform can be invaluable instruments for this purpose. The “Results” chapter shows how these pre-processing steps can significantly improve classification time without compromising the quality of the results.

After clustering the processed dataset, a classifier will be used to label the high-resolution image, which can be built starting from the grouping obtained for the low-resolution dataset through unsupervised learning. This procedure will be described in the next section.

2.6 Classifier

2.6.1 Introduction

Classification, like clustering, is about labelling points in a dataset, or in other words dividing the observations into groups characterized by a sufficient grade of similarity between points inside a group and a sufficient dissimilarity between groups. The difference is in the fact that what is commonly known as clustering is a form of *unsupervised* classification of the data points, as mentioned, while usually the term “classification” by itself is referred to *supervised* classification, since it requires a pre-labelled dataset to *train* the classifier for the subsequent classification of new points. Therefore, in this case, the “input” to the method is a set of points and corresponding labels, and the “output”¹⁸ is a model that somehow relates the observations to the corresponding labels, possibly with a small probability of misclassification. In clustering, no information is available to “label” the data, therefore clustering is defined as an

¹⁸ it is very important to not confuse the “input”/“output” of the method with what is commonly referred to as input and output variables. In fact, the “input” to the method is a set of input and output variables (the observations and the labels, respectively), on which a model is trained in order to be able, given new input variables, to predict the output variables. The trained model is the “output” of the method, in the sense that it is what the method produces.

unsupervised technique, since there is no possibility to know the error committed in the labelling. On the other hand, for classification it is possible to know the quantity and location of the misclassified points with certainty at least for the training dataset, being able to infer in which areas there is higher likelihood of misclassification. In conclusion, “labels” associated to groups during clustering are “data driven”, i.e. obtained solely from the data (Jain *et al.*, 1999), while in classification already labelled training data are used to learn how to discriminate the groups building a (hopefully) simple model that can be used in turn to label new data.

2.6.2 Classification fundamentals and methods¹⁹

In classification, input variables, also known as *predictors*, can be qualitative or quantitative variables; the response is always a categorical variable. In literature, the input is usually indicated as X independently on the nature of the observations, while the output there is a different notation if the true classification or the predicted one is considered. For the training dataset, that is the labelled data used to “teach” the classifier the “difference” between groups, the output is denoted by G (that stands for “groups”), whereas the predicted output, that is produced by a trained classifier, is marked with a circumflex accent (\hat{G}). Notice that even a qualitative output can be represented by a number for convenience.

Output classes are defined by what is called a *decision boundary*, that is the hypersurface in \mathbb{R}^λ separating points assigned to different groups by the classifier, where λ is the number of features measured for each point. These boundaries can have specific properties and shapes, corresponding to different classification methods. Problems like a suboptimal shape of the decision boundary or an overlapping between the point “clouds” in the \mathbb{R}^λ space can lead to an incorrect prediction of the true label of the training dataset itself, but these problems cannot be avoided completely, so some degree of misclassification is usually present. In general, a classification theory trying to find the function $f(X)$ able to predict the output G given the value of the input X needs to consider also a *loss function* that penalizes the possible error. For a categorical output, that can assume values from a finite set of classes $\mathcal{G} = \{1, \dots, K\}$, the loss function can be a matrix L of dimension $K \times K$ where each element $L(l, m)$ is the price paid for labelling an object of class l with the erroneous label m , with $l, m \in \{1, \dots, K\}$. Notice that of course the diagonal of this matrix is zero. The non-diagonal elements can be all equal, and in particular commonly they are set to be equal to one (*0-1 loss function*), but also a higher or lower penalization cost can be chosen dependently on the application. Defined a loss function, the expected prediction error can be evaluated as

¹⁹ This section is largely based on (Hastie *et al.*, 2001) and on the online documentation about “Classification” available on the MATLAB® website. Refer to these sources for more information

$$EPE(f = \hat{G}) \triangleq \mathbb{E} \left[L(G, \hat{G}(X)) \right] = \mathbb{E}_{X,Y} \left[L(G, \hat{G}(X)) \right]. \quad (2.6.2-1)$$

The expectation is taken with respect to the joint distribution $P(G, X)$, and since G is conditionally dependent on X , i.e. $P(G, X) = P(X)P(G|X)$, equation (2.6.2-1) can be written as

$$EPE(\hat{G}) = \mathbb{E}_X \mathbb{E}_{G|X} \left[L(G, \hat{G}(X)) \mid X = x \right]. \quad (2.6.2-2)$$

The best value of $\hat{G}(X)$ is the one that minimizes the expected prediction error. Equation (2.6.2-2) shows that the minimum can be calculated pointwise since the overall minimum is going to be obtained if the fixed-point minimum is ensured for each point, obtaining:

$$\begin{aligned} \hat{G}(x) &= \operatorname{argmin}_{g \in \mathcal{G}} \mathbb{E}_{G|X} [L(G, g) \mid X = x] \\ &= \operatorname{argmin}_{g \in \mathcal{G}} \sum_{k=1}^K L(G, g) P(G_k \mid X = x). \end{aligned} \quad (2.6.2-3)$$

2.6.2.1 Linear classifiers

The simplest shape that a decision boundary can have is linear, and in fact linear classifiers form a very important and widely applied class of procedures where the hypersurfaces separating different groups are hyperplanes. The construction of these boundaries can be based on the value of a *discriminant function* $\delta(x)$, classifying each object x to the class corresponding to the largest value of this function. This approach is known as *discriminant analysis*. Also models using posterior probabilities $P(G|X)$ are in this class, since also a probability function (or some monotonic transformation of it) can be viewed as a discriminant function. If the discriminant, or a monotonic transformation of it, is linear in x , the decision boundary is also linear. The same holds true for the posterior probability. In fact, the decision boundary between two classes l and m can be also thought as the surface for which for each point the probability of being class l or class m is the same.

With the addition of some assumption, a linear discriminant model can be represented through the already mentioned equation (2.6.2-3):

$$\hat{G}(x) = \operatorname{argmin}_{g \in \mathcal{G}} \sum_{k=1}^K L(G, g) P(G_k \mid X = x). \quad (2.6.2-3)$$

The conditional probability $P(G_k \mid X = x)$, equivalently written also as $P(G = k \mid X = x)$, is also called *posterior probability* and it represent the probability that, if the point is x , the real class of the point is k , so it is called *posterior* because it refers to the probability that an event

occurs *after* picking a specific point x from the dataset X , the event being that the class is actually k . Applying the Bayes theorem:

$$P(G = k | X = x) = P(k, x) = \frac{P(x|k)P(k)}{P(x)} = \frac{P(x|k)P(k)}{\sum_k P(x|k)P(k)} \quad (2.6.2-4)$$

Where $P(k)$ is the *prior probability* of the class k , that means the probability of the class k in general in the domain, and not for a specific point x . Being unknown, it can be taken to be equal for all classes, or *uniform*, assuming the expression of $P(k) = 1/K$ where K is the total number of classes. If it is believed that not all classes have the same likelihood in the domain and that the dataset is a fair representation of the overall likelihood, *empirical* representation of the prior probability can be more appropriate, defining it as the ratio between the number of points of class k and the total number of points, or in symbols $P(k) = N_k/N$.

The conditional probability $P(x|k)$ is actually the probability density function of points x in class k , or class-conditional density, and in the linear discriminant model it is assumed to be a multivariate Gaussian where all the classes have the same covariance matrix $\Sigma_k = \Sigma$, $\forall k$:

$$P(x|k) = \frac{1}{2\pi^{\lambda/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \cdot \Sigma_k^{-1} \cdot (x - \mu_k)\right) \quad (2.6.2-5)$$

Where μ_k is the average for the class k , evaluated as $\mu_k = \sum_{g_i=k} x_i / N_k$ and $|\Sigma_k|$ and Σ_k^{-1} represent respectively determinant and inverse of the class covariance matrix Σ_k , equal for all the classes, calculated as

$$\Sigma = \sum_{k=1}^K \sum_{g_i=k} (x_i - \mu_k)(x_i - \mu_k)^T / (N - K).$$

Therefore, for the linear discriminant method the predicted value can be calculated as

$$\hat{G}(x) = \operatorname{argmin}_{g \in \mathcal{G}} \sum_{k=1}^K L(k, g) P(k | x). \quad (2.6.2-6)$$

This method is defined to be linear because, thanks to the assumption of a multivariate Gaussian density with constant covariance matrix, it is possible to write that, for the decision boundary between classes l and m

$$P(l|x) = P(m|x)$$

And substituting equation (2.6.2-5)

$$\begin{aligned} & \frac{\frac{1}{2\pi|\boldsymbol{\Sigma}_l|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_l)^T \cdot \boldsymbol{\Sigma}_l^{-1} \cdot (x - \mu_l)\right) \cdot P(l)}{\sum_k P(x|k)P(k)} \\ &= \frac{\frac{1}{2\pi|\boldsymbol{\Sigma}_m|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_m)^T \cdot \boldsymbol{\Sigma}_m^{-1} \cdot (x - \mu_m)\right) \cdot P(m)}{\sum_k P(x|k)P(k)} \end{aligned}$$

Remembering that the covariance matrix is equal for all groups ($\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma} \forall k$) and simplifying

$$\begin{aligned} & -\frac{1}{2}(x - \mu_l)^T \cdot \boldsymbol{\Sigma}^{-1} \cdot (x - \mu_l) + \log P(l) \\ &= -\frac{1}{2}(x - \mu_m)^T \cdot \boldsymbol{\Sigma}^{-1} \cdot (x - \mu_m) + \log P(m) \end{aligned}$$

That can be written also as

$$\begin{aligned} & x^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_l - \frac{1}{2}\mu_l^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_l + \log P(l) \\ &= x^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_m - \frac{1}{2}\mu_m^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_m + \log P(m) \end{aligned}$$

Therefore, the discriminant function can be defined as

$$\delta_k(x) = x^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_k - \frac{1}{2}\mu_k^T \cdot \boldsymbol{\Sigma}^{-1} \cdot \mu_k + \log P(k)$$

That is a function linear in x , as requested to define the method as linear.

In practice, implementation in MATLAB of linear discriminant analysis uses the form of equation (2.6.2-6) to build and apply the model.

Since the model is based on some assumptions about the dataset and the shape of the boundaries between groups, problems may arise if these assumptions are far from the real properties of the dataset. For example, the distribution of the training dataset could be very far from a multivariate gaussian. Figure 2.15 shows a distribution that could resemble a gaussian with approximations and in fact is obtained plotting together a multivariate gaussian with some random clouds of points added. If all the points in the figure have the same label, they are going to be used all together to evaluate the average and the covariance matrix needed for the model, and since the point distribution is not really gaussian this in principle should worsen the results. Problems should also arise if the natural boundaries between groups are not linear. Being the application of linear discriminant analysis so widely spread, it could be thought that real data have very often a (sufficiently) gaussian distribution with equal covariance and boundaries tend to be linear, but in reality, as proposed by Hastie *et al.* (2001),

the reason behind the wide success of this simple method is that data can better support simple decision boundaries (linear or quadratic), and the estimates under the gaussian model are in fact quite stable. Furthermore, too complex models can end up adapting too closely to the training data reducing their capability of generalization to a new dataset.

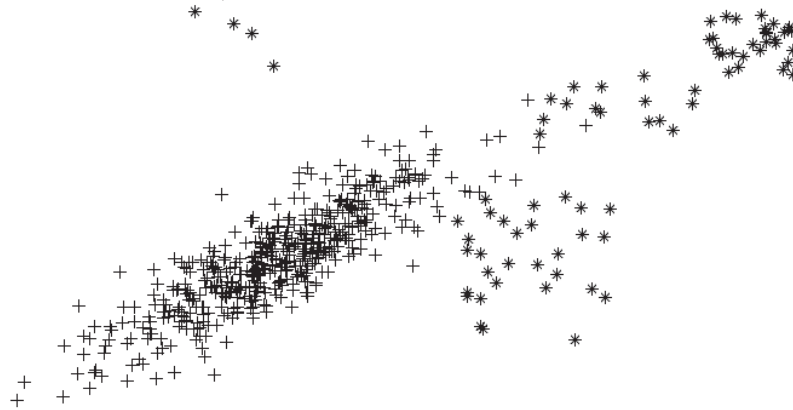


Figure 2.15 – Example of a multivariate gaussian distribution (+) with some disturbance points (*) that alter the sample average and the covariance matrix.

2.6.3 Input dataset

Before concluding the section about classification, it is important to notice some peculiarities of the problem under examination, in particular as concerning the input dataset.

2.6.3.1 Input dataset: labels

In its original conception, a classifier is meant to be trained on a dataset with measured classes. In this specific case, the classes are not available in the original dataset, and are “extracted” from the data structure through clustering operation, defining the concept of similarity between points. This means that the results from the unsupervised grouping operation can be inaccurate especially on the transition borders from a group to another. Nevertheless, to obtain a good classifier it is important to provide a training dataset as “polished” as possible, because erroneous labelling in the cluster results will undermine the success of the classifier training, degrading the quality of the results attainable in its future application.

As misclassification affects mostly borders between groups or small areas, image enhancement methods are commonly applied to mitigate these artefacts, including morphological operations tools.

In mathematics, morphology is related to concepts as shape, convexity, connectivity, etc. In image processing, morphological operators deal with description and processing of regions

shapes, boundaries, convex hull, and similar properties (Gonzalez *et al.*, 2010). Dilation and erosion are two of the fundamental morphological operations available for image processing, from which more complex image modification operators can be derived.

Dilation and erosion.²⁰ Both dilation and erosion are based on a *structuring element*, that is simply a shape of given size that is made to interact with the image and modify it as a consequence of this interaction. In MATLAB, the structuring element is represented by a usually small binary matrix with zeros for the background and ones corresponding to the actual object shape. Several options are available, such as diamond, disk, line, rectangle, but also introduction of custom structuring elements is possible. The shape and the size of the structuring elements, as well as the position of the so called “origin” that is the reference point of the element, will influence the dependence on the direction and the extent of the interaction between the structuring element and the image regions.

For a binary image, dilation is aimed to “thicken” objects in the image, assuming that the objects are the “white” part of the image, i.e. they correspond to ones in the image matrix. In greyscale, dilation actually “dilates” clear objects in the picture at the expense of neighboring “dark” objects. During binary image dilation the structuring element is translated to all the points of the image, and the dilated image is the set of all the origin locations for which the translated and reflected-by-the-origin element overlaps at least one point of the object (that is a “1” in the image matrix). For a greyscale dilation, if a binary (or “flat”) element is considered, the same operation of translation and rotation are performed, and after a local maximum is computed among the points overlapped by the ones of the structuring element. Erosion is the opposite operation, evaluating a local minimum for a greyscale image, whereas for a binary image erosion can be seen as the set of origin points for which the structuring element is completely contained by the object.

Sequential applications of dilation and erosion operations in a specific order can result in “new” morphological operations, like *opening* and *closing*. The opening of an image is obtained by applying erosion and then dilation, with the general result of suppressing the bright details that are smaller than the structuring element, vice versa closing applies erosion after dilation, suppressing dark details, or “closing” bright objects.

An appropriate and smart combination of these morphological operations can improve the quality of the labels matrix obtained after the clustering operation removing misclassified points at the borders.

2.6.3.2 Input dataset: data points

Another problem specific of this application is related to the data points to be used as input to the training procedure. In fact, during the clustering, contrast enhancing operations are

²⁰ For more details, refer to Gonzalez *et al.* (2010), on which this paragraph is based

performed on the wavelet transform results (“*wt-data*”) to enhance inter-group distance and improve the clustering results. Nonetheless, since the objective is to classify the 3-“bands” image obtained after principal component analysis, that is just a high resolution version of the *wt-image*, the training dataset must consider the wavelet transform results *before* the contrast enhancement, associating to it the labels obtained from clustering the enhanced image. As a matter of fact, the image intensity transformation considers the properties of the image itself, since the rescaling considers the extremes values *of the specific image* transformed, and the contrast is enhanced saturating the top and bottom 1% of all pixel values *of the image itself*. In other words, one point with intensity $p = [k \ l \ m]$ in the hyperspace \mathbb{R}^3 of intensity measurements for 3 wavebands could be matched to different points $p' = [k' \ l' \ m']$ and $p'' = [k'' \ l'' \ m'']$ depending on the distribution of intensity values of the initial image. If the classifier was trained on an enhanced image where the point p of intensities $[k \ l \ m]$ was transformed into the point p' after enhancement, this could cause a problem for future applications because, if for the new image to classify the histogram has a different distribution of intensities, the enhancement is going to map the point of intensities $[k \ l \ m]$ to a point p'' very distant from p' and the classifier is not going to be able to perform the labelling correctly. Also if the only image to be classified is the high resolution version of the training dataset, it is important to remember that the wavelet transform has the effect of smoothing the peaks and valleys in the data, therefore the same problem exposed for future applications affects also the use for this specific image, even if less clearly.

2.7 Refining the model

In science, it usually happens that the procedure or model proposed to address and solve a problem has some “degrees of freedom” or, in a broader sense, some choices among various options available must be made in order to be able to apply the technique. Therefore, after a working algorithm is implemented, it is good practice to explore the different solutions that can be obtained if different choices for the model parameters are considered. In other words, some form of sensitivity study and model refinement can be performed. For the procedure proposed, the two main parameters are related to the size reduction achieved in both the spectral and the spatial domains. In fact, the number of retained components in principal component analysis and the resolution of the wavelet transformed image both influence the quality of the final result. A full sensitivity analysis usually would require simultaneous variation of both parameters to be studied, i.e. if the spatial parameter and the spectral parameter can assume respectively i and j different values, the total number of combinations to be considered is in principle $i \times j$. In truth, the two parameters could be sufficiently

unrelated that general behaviour could be extracted starting from the base case and varying just one parameter or the other.

In the hyperspace where points correspond to observations and the variables are the spectral bands, if data are naturally grouped in certain regions, removing some points from all the groups, like basically happens when spatial resolution of the image is lowered²¹, is not going to change the grouping themselves and so it is not going to affect the ability of the spectral variables to separate the clusters. Figure 2.16 (a) and (b) illustrate this situation. The worsening action on the final clustering caused by the lowering of the resolution is independent from the spectral bands, and depends on the fact that after a certain level of coarseness there are not sufficient points for each group anymore and the approximation added at each step starts to accumulate errors. In other terms, adding more and more principal components cannot improve clustering results if the spatial resolution has been lowered up to a point where the approximation smoothed every difference between groups and very few points compose each group.

Vice versa, as long as the spatial detail is sufficient for a clustering operation, varying the number of components has consequences that are independent on the number of points. The groups, in fact, remain virtually the same but the separation between groups projections onto the principal components (the new virtual “wavebands”) can be worsened significantly if the vectors of the new basis are not enough in number. In other words, a higher spatial resolution would not improve clustering results if the number of principal components selected is inadequate. This problem is illustrated in Figure 2.16 (c) and (d), where it appears clearly that considering only one principal component would not allow the separation between groups (1) and (2).

The suggested approach would be to start with a parsimonious description of the spatial and spectral domains, and progressively improve the resolution of analysis if and only if, the results get significantly better. Otherwise, the process stops, and the current model is good enough for segmenting all pixels of the image. The additional computational burden will not lead to improves results and may even render the whole segmentation operation unfeasible.

²¹ In truth, also some approximation in the form of averaging among adjacent points is also done at each step

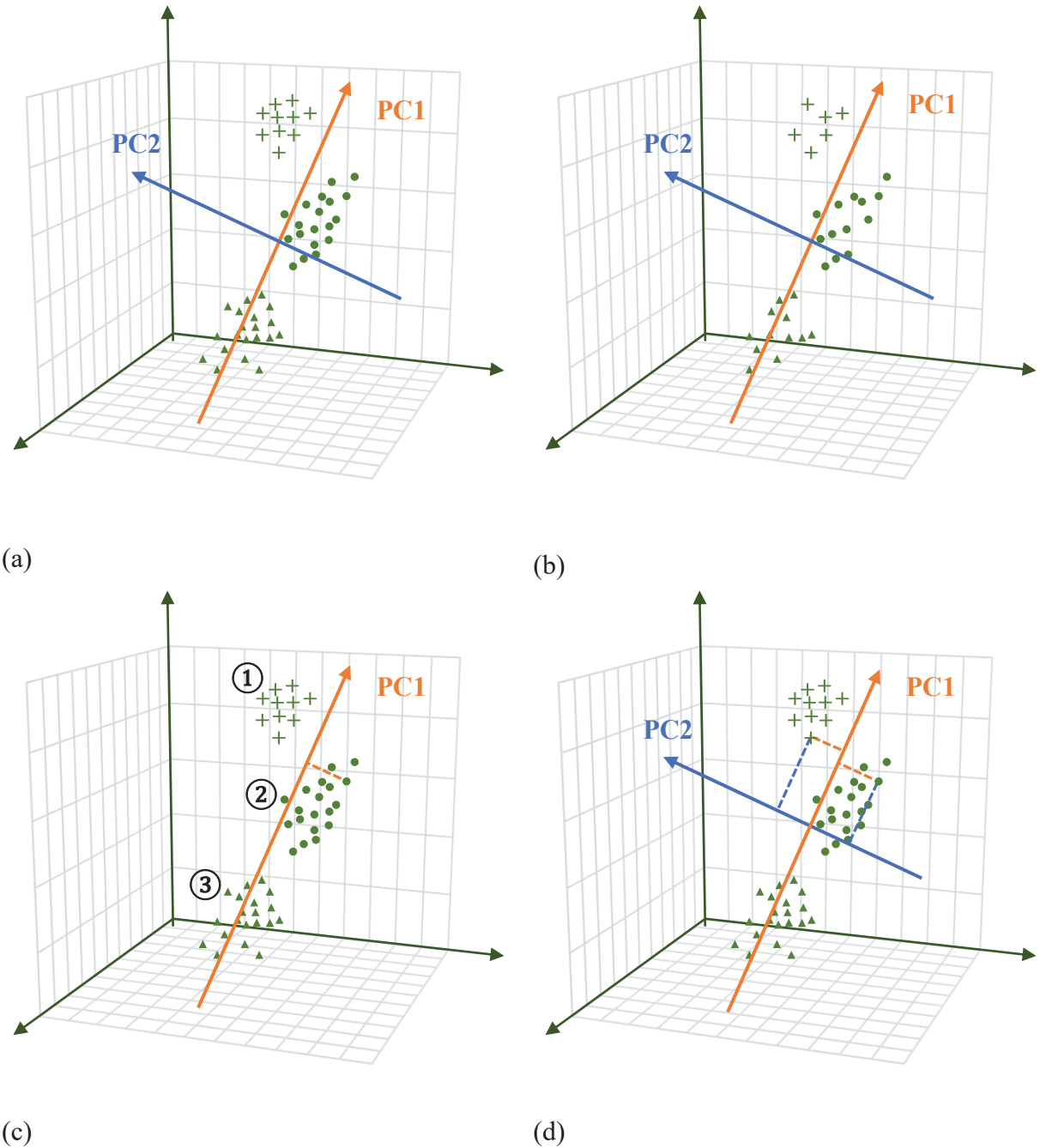


Figure 2.16 – Example of reduction of the spatial dimension – (a) and (b) – and of the spectral dimension – (c) and (d). Objects naturally belonging to different groups are represented with markers of different shapes.

Chapter 3

Results and Discussion

This chapter covers the main results obtained in the analysis. Preliminary results have been presented in the previous chapter for simplicity of illustration, being some minor conclusions the natural consequence of the theoretical and practical discussions addressed in the previous chapter itself. On the other hand, some brief description of strictly practical methods for selecting and understanding the results are included in this “Results” chapter for continuity in the exposition of the topic, being the understanding of the results unavoidably related to the description of the method itself.

Results are divided in sections related to the different steps of the analysis – data inspection, principal component analysis, wavelet transform, clustering, linear model, ...

3.1 Data inspection

Before transforming the data, it is good practice to inspect them with some detail, even if the significant size of the dataset already suggested to use the principal component analysis to reduce its dimension.

A typical way of representing spectral data is through line plots, showing the spectral response of a specific point in the picture. As an example, in Figure 3.1 spectral response of two *observations* (pixels) are plotted. These two points are chosen such that they correspond to two different objects from the picture. After numbering the objects in a greyscale photo of the configuration, these objects can be identified as object (1) and object (5) in Figure 3.2. For both objects, the intensity of the reflected electromagnetic waves captured by the camera is very low for the first and the last wavelength bands of the range considered. Extracting, from the hyperspectral image, “slices” corresponding to the first and the last wavelength bands it is clear that for these variables the quality of the data collected is very low and the picture is very blurred and low in contrast (Figure 3.3). Notice that the pictures appear to be light

because, even if the intensity measured for each pixel is very low compared to other bands, when representing the picture with `imshow` the data is automatically rescaled considering the total variability of the single picture corresponding to a single lambda-range. On the contrary, a band taken from the centre of the range, for example the 150th band, has high sharpness and contrast in comparison with the other two examples. This could be related to the spectral

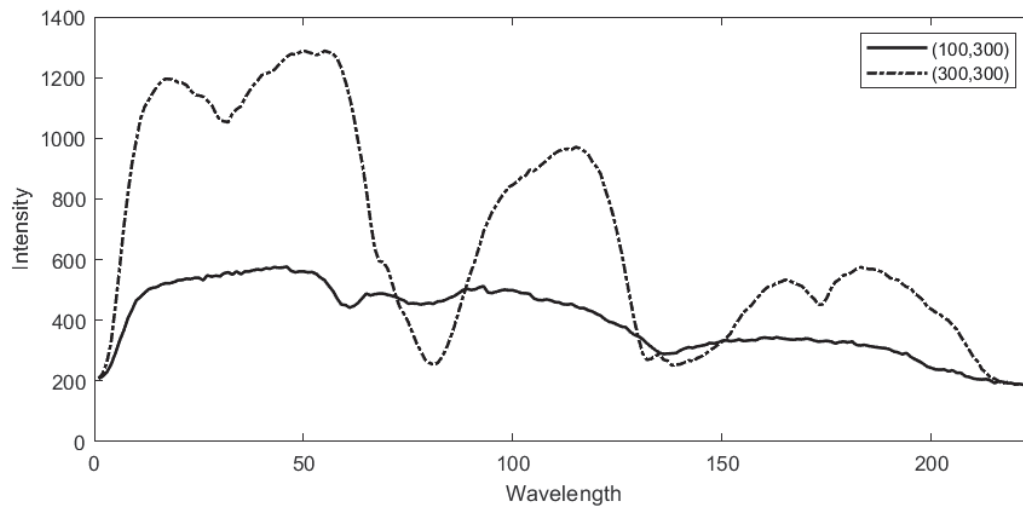


Figure 3.1 – Spectral response of different points in the picture. Continuous line for the point of coordinates $(row,col)=(100,300)$, dashed line for the point $(300,300)$

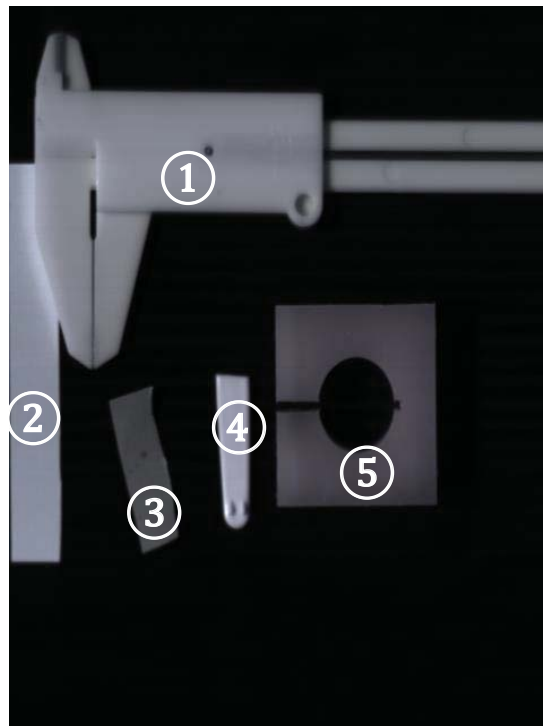


Figure 3.2 – Identification numbers for the objects in the picture

sensitivity of the instrument. In fact, it could be that the wavelength range considered comprises, at the extremes, bands for which the efficiency of detection of the camera is lower. Since the full range corresponds exactly to 224 bands, that is the number of bands considered for the experiment, assuming that the full range is from 900 nm to 1700 nm and that the absolute quantum efficiency has a shape as the one represented in the datasheet of the camera (Appendix 1 – Hyperspectral camera datasheet), low efficiency at extremes is exactly what causes lower quality/lower contrast images for the first and the last bands in the range. Plotting the intensity response of a row and a column intercepting an object in the picture may lead to more interesting discoveries. Figure 3.4 for example shows the profiles of the 90th row and the 155th column, both intersecting object (1)²². It appears that along the row the response is less noisy than along the column, having oscillatory profile in the last case, especially for the intervals corresponding to the background. Moreover, the transition from background to the object is sharper along the row, especially on the right side where there is even a peak in the signal. Repeating the analysis for rows and columns intercepting different objects, a qualitative assessment of the profile leads to similar results, that can therefore be considered general for the picture. For the rows, the peak on the right side is likely determined by the position of the illumination source, since observing Figure 3.2 from the direction of the shadowing in object one it can be inferred that the “light” source is positioned on the right side. Remembering also that the acquisition of the data is carried out measuring the response in all the spectrum for one row at a time²³, it could be reasonable to think that the profile along the row is automatically smoothed by the recording system itself, while the new row is seen as a new “sample” and therefore most of the noise could happen along the column if there is not any form of automatic correction/interpolation of the data. The joint contribution of these two experimental conditions could be the cause of what observed in Figure 10. In reality, likewise the explanation for the waves along columns, especially on background, could be related to the use of a wavy base for the objects, for example a piece of cardboard. Experimental conditions are not known in their completeness, therefore no definitive answer exists. Considering also how, for the noisy pictures at the beginning and the end of the spectrum, there is a significant band noise along columns, both the hypothetical wavy base and the horizontality of the acquisition method could cause a noisier and more wavy profile for the column plot.

The typical laboratory configuration represented in Figure 3.2, in the previous chapter, could confirm the directionality of the light source. Also the wavy base could be related to the use of this assembly, since the Specim Scanner base has a wavy appearance in photos. It could be that no additional monochromatic base was positioned on the Scanner base, below the objects to be scanned, or that the one used was too thin and slightly transparent, maybe just for the

²² Refer to Figure 3.2 for object number

²³ Refer to the previous chapter, paragraph §2.2on Opening a Hyperspectral Image for more details

infrared camera and not for the human eye (e.g. a thin sheet of dark paper). This is again only speculation, but it appears to be a very reasonable explanation.

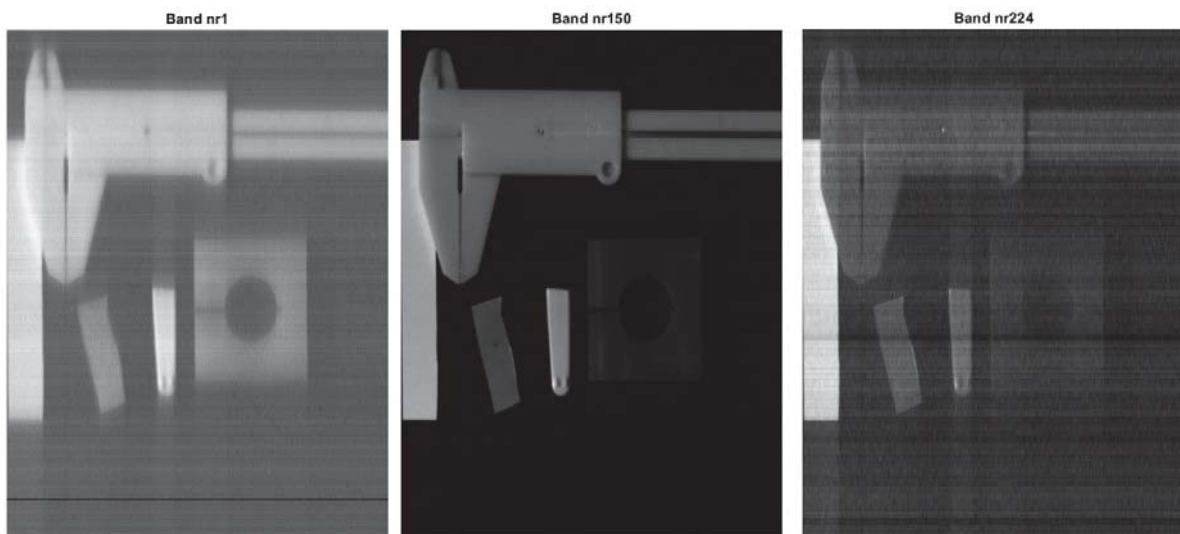


Figure 3.3 – Greyscale conversion of the response for bands number 1 (left), 150 (middle), 224 (right).

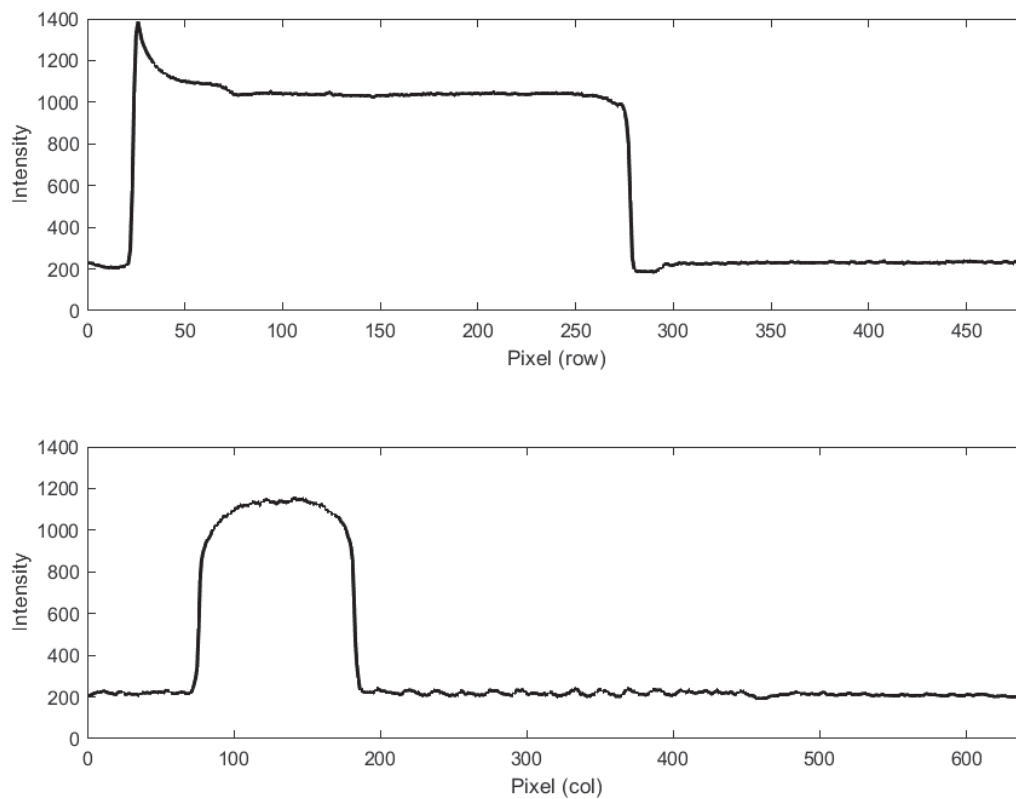


Figure 3.4 – Intensity of the response of row 90 (up) and column 155 (down) for the 150th band

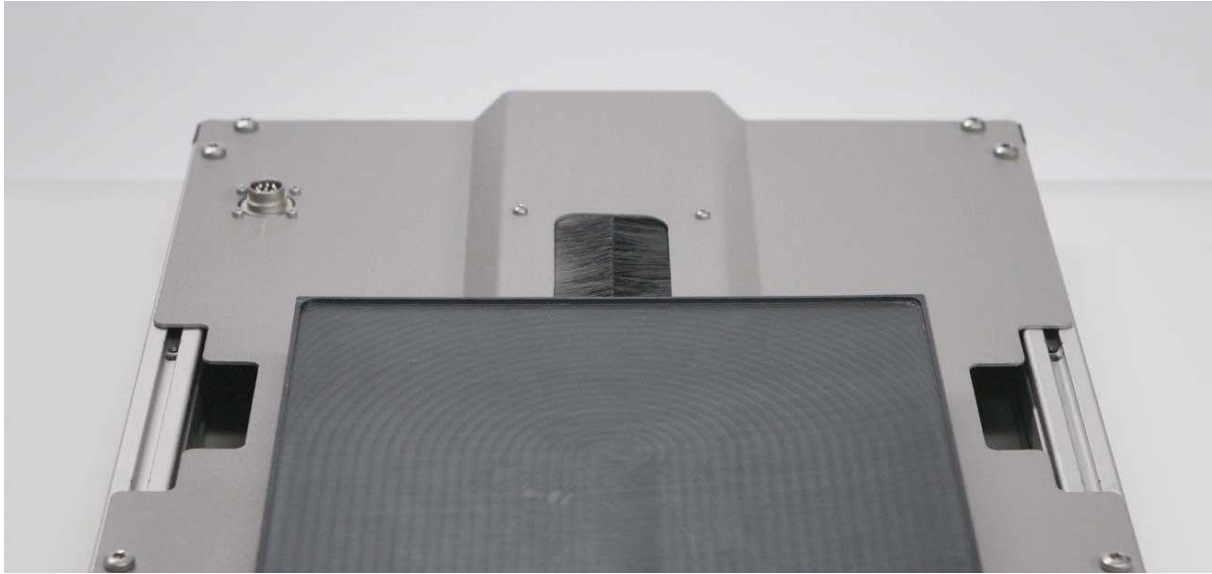


Figure 3.5 – Base of the Specim Lab Scanner 40x20 (from the SpecimSpectral YouTube Channel)

3.2 Principal component analysis

3.2.1 Pre-processing

In industrial applications, applying PCA on raw data may not give good results or may even be meaningless. This can happen for several reasons, the most common one being that the data matrix contains measurements for different properties of the system expressed in different units, such as temperatures, flowrates, pressures, so that the typical order of magnitude may vary significantly between one property and the other. For this dataset, all the variables are equivalent, since they are all measurement of reflectance, even if for different wavelength bands²⁴, and therefore they have the same unit of measure (UOM). If the UOMs are different and the scale of the data in these UOMs are such that there is strong difference in the numerical values of the measurement only related to the UOM, usually autoscaling is performed to avoid numerical errors and modelling problems. In this case, therefore, autoscaling is not necessary. On the contrary, dividing observations for each lambda by their standard deviation could amplify noise signals with respect to actual useful information if the spectral response of one point is observed. Considering the point P in the image, and the intensity of the response measured for that point P at each band, it is clear that with

²⁴ In reality, the sensor may be more reactive to electromagnetic waves of some bands than of others, but the absolute quantum efficiency curve for the camera FX17 is enough flat in the operating region, otherwise some correction on the data would be needed (if not automatically performed by the acquisition computer program supplied with the hyperspectral camera)

autoscaling the dividing factor is going to be different for each lambda-range, and in particular it is going to be greater for bands with higher variability of the data, that contain the actual information, and lower for bands that contain less information (i.e. less variability) and more noise (or better a higher ratio between noise and actual information). Therefore, the spectral plot for the point P (like the one in Figure 7) may be actually flattened by division by different standard deviations (one for each lambda band), losing significant information and amplifying the noise. For this reason, for the specific data under examination, not rescaling seems to be the best choice.

An important reason for centring, even without rescaling, may be numerical (Bro *et al.*, 2003). It can be proved that when principal component analysis is performed with non-explicit methods, the ratio between the two largest eigenvalues determines the convergence rate. With some methods also strongly correlated components may cause problems. Some of these computational problems may be avoided if the data are centred across the first mode, i.e. along variables, subtracting for each variable the mean of the values observed for that variable. Centring can also avoid problems related to the presence of offsets. Evaluating the absolute minimum of the hyperspectral matrix, an offset of 128 can be found for the intensities measured, and this is not a small value considering that for some points intensities can go from 200 to 600 (e.g. the continuous line in Figure 3.1). Finally, sometimes centred dataset allows better fitting, i.e. lower residuals. Nevertheless, these hypothetical advantages cannot be checked before the analysis.

3.2.2 Number of components

Quoting a famous article by Raymond Cattell (Cattell, 1966) "*Unfortunately [...] a test [to choose the number of components] does not exist - even a long or complex one - which is both mathematically precise and logically satisfying*".

In practice, there are some methods that could help assessing approximately the actual dimension of the problem.

3.2.2.1 Practical methods for selection of the number of principal components

Qualitative approach. At first, a wise choice could be to not fix a priori the number of components but just exploring the results obtained considering the first few components and then adding the following ones increasing the cumulative variability explained (Bro *et al.*, 2014). Nevertheless, this approach is too qualitative to be used for more than giving a first glance to the data, therefore it will not be further considered.

Scree test. A useful technique widely accepted is the Scree test (Cattell, 1966). It is based on eigenvalue inspection. In point of fact, it can be proved mathematically (Bro *et al.*, 2014) that

the eigenvalue corresponding to a specific component is equal to the variation for that component. For the Scree test, eigenvalues in descending order are plotted against the number of components, and usually the eigenvalues decrease fast for the first components, which are assumed to retain most of the actual information and minor noise contribution. In contrast with the fast decreasing for the first components, it is assumed that the lower magnitude eigenvalues, corresponding to the eigenvectors capturing most of the noise, decrease slowly and almost linearly. These assumptions are the result of "a hundred or more factor analyses carried out over thirty years", as claimed by Cattell himself in his publication introducing the Scree test (Cattell, 1966).

Eigenvalue-greater-than-one rule. Another common test considers only the eigenvalues above one. It can be applied only if the data are autoscaled. Being the dataset under investigation mean centred but not divided by the standard deviation in the pre-processing step, this technique cannot be applied.

Variation explained. The most intuitive quantitative rule for the selection of PCs involves the evaluation of the amount of variation explained. As already mentioned, each eigenvalue quantifies the variation of the scores on the corresponding principal component, therefore it is clearly related to the fraction of the original total variability of the data explained by that component. This fraction may be computed as $\lambda_i / \sum \lambda_j$, where λ_i is the eigenvalue related to PC_i and the sum is for all the eigenvalues of the covariance matrix.

Residuals inspection. As derived in the previous chapter (paragraph §3.3.2.1), after defining a principal component an approximation of the original data, i.e. the result of its projection on the principal component, can be calculated, and by definition subtracting the approximated matrix from the original data matrix produces the error matrix. A prohibitive amount of data, as it is the case for a hyperspectral image, makes it impossible to inspect data by means of a plot or a table. Statistics summarizing these results are necessary. The error in general might be positive or negative, so a good statistic variable could be for example the overall mean of the absolute value of the error. The specific typology of the dataset under examination has a significant advantage on other kinds of huge datasets: error can be represented in form of picture.

3.2.2.2 Application of the methods

Scree plot. Empirical inspection of the Scree plot, representing eigenvalues in descending order versus principal component indexes, can give an idea of the number of components to retain. With datasets from real problems, it could happen that the first eigenvalue is extremely large if compared to the others, and that is the case for the hyperspectral image under analysis. In these cases, it is necessary to zoom to even be able to discern the different eigenvalues

(Figure 3.7). From this analysis, it appears that considering the first three or four principal components might be enough for capturing most of the information.

Variation explained. As abovementioned, the fraction of variation explained through each principal component is equal to the corresponding eigenvector divided by a constant, being the sum of all the eigenvectors obviously a constant. Consequently, the qualitative shape of the explained variation plot (Figure 3.8) is identical to the scree plot, the only difference being the scale of the ordinate, that is normalized for the variation plot. For the Scree test the objective was a qualitative inspection to find when the eigenvalues start to level off, and per se the numerical value of the eigenvectors gives little information. The fraction of the total variation explained (Figure 3.9), on the other hand, is a quantity that might be worth analysing with more detail. Table 3.1 shows that even considering only one component 98% of the total

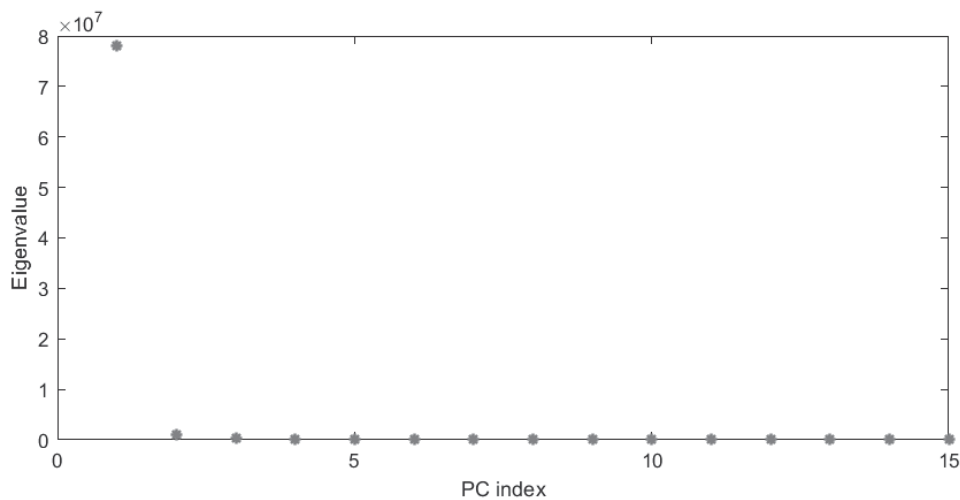


Figure 3.6 – Eigenvalues corresponding to each principal component (first 15 components)

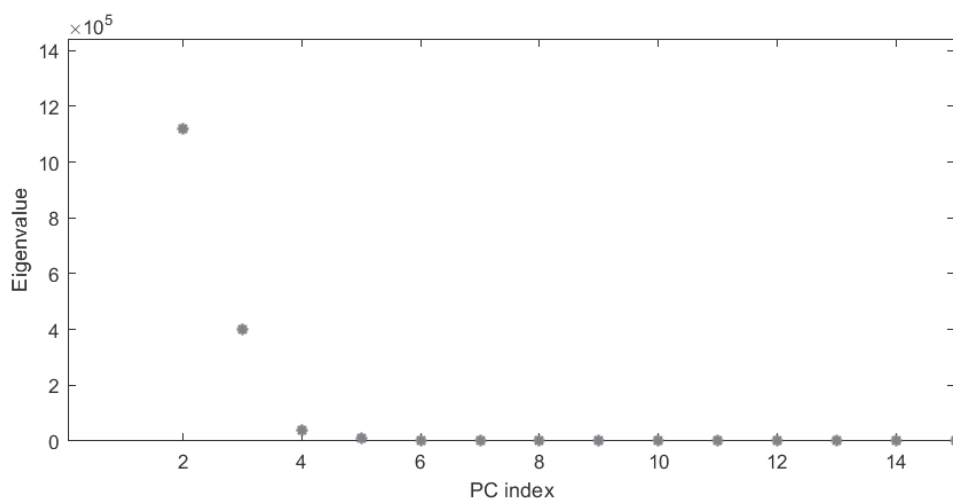


Figure 3.7 – Eigenvalues corresponding to each principal component (zoom on components 2-15)

Component	Fraction of variation explained	Cumulative fraction
1	0.98016	0.98016
2	0.01405	0.99422
3	0.00503	0.99925
4	0.00048	0.99974
5	0.00011	0.99985
6	0.00003	0.99989
7	0.00002	0.99991

Table 3.1 – Fraction of total variation explained and cumulative fraction for the first 7 components

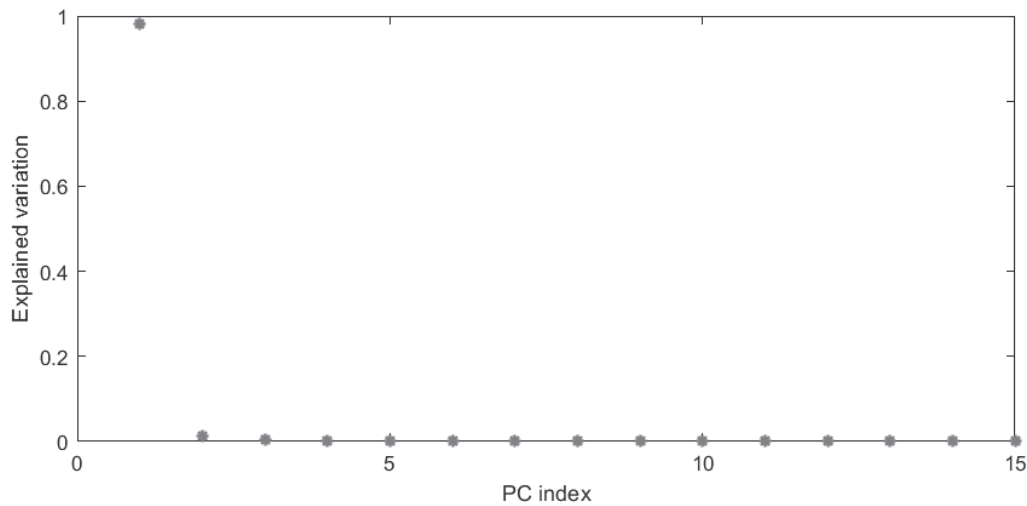


Figure 3.8 – Fraction of variation explained by each principal component (first 15 components)

variation is explained, and the percentage goes even above 99.9% for three components, the second and the third components adding very little information. This said, it is very important to not fall into the common mistake of considering 98% a high percentage of variation explained in absolute terms, since it strongly depends on the typology of data and more specifically on the quality of the measurements, i.e. the amount of noise. In truth it is not possible to know this a priori without very detailed information on the experimental condition, but the FX17 datasheet reports a nominal (maximum) signal to noise ratio of 1000:1, so it is reasonable to assume that the detection accuracy is high. For highly precise data, even components apparently capturing a small fraction of information could in fact be useful for classification purposes, since that slight difference in the projection on the second or third components could be crucial for the segregation problem and contains actual

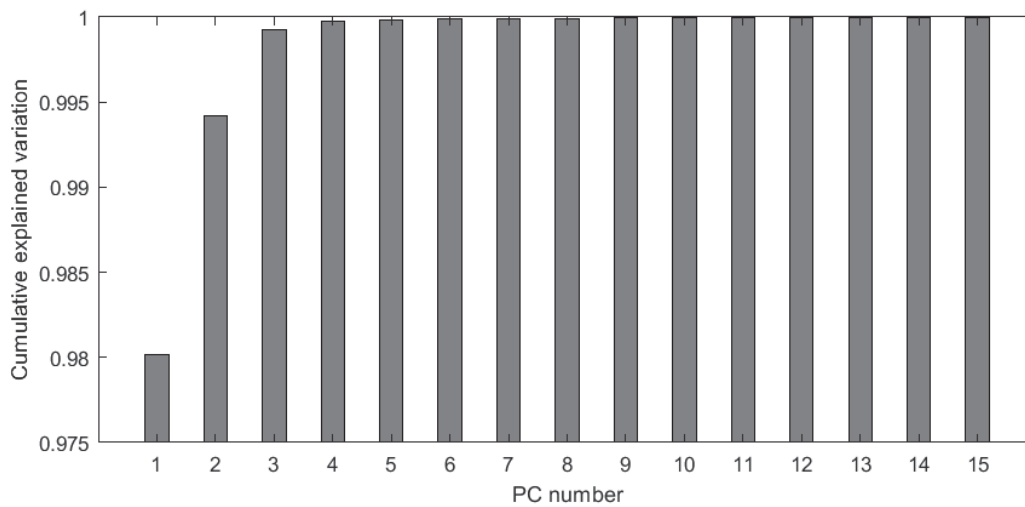


Figure 3.9 – Cumulative explained variation by the number of principal components retained (from 1 to 15 retained)

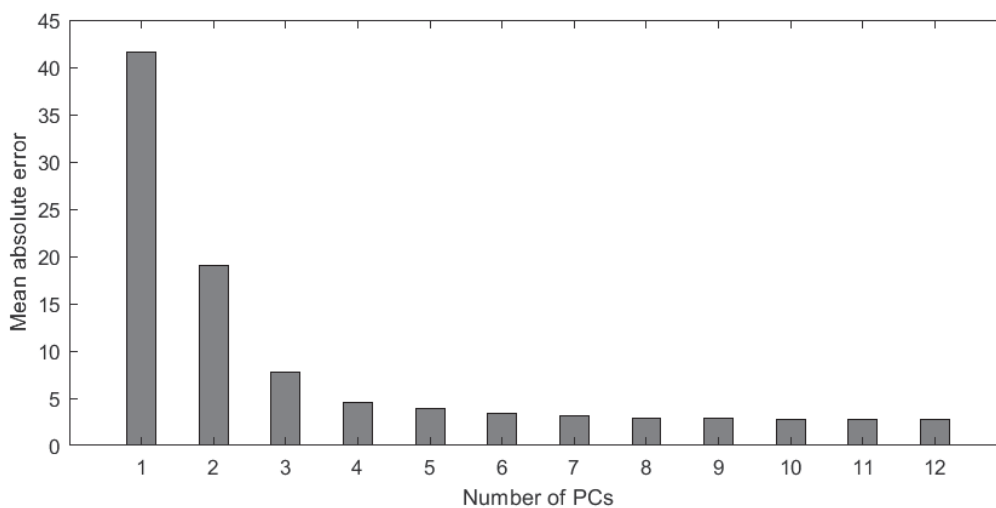


Figure 3.10 – Mean of the absolute error as function of the number of principal components

information more than the simple noise that low eigenvalues components would capture for less sophisticated instrumentation.

Residuals inspection. Figure 3.10 shows that the mean value of the absolute error decreases with the addition of each principal component for the first components, but beyond the fourth component it levels off, making the further addition of variables meaningless. This confirms previous conclusions considering other typologies of test. Being the dataset under analysis a picture, errors can also be represented graphically in form of picture, with brighter areas meaning higher value of the residual. For easier understanding and interpretation of the

results, the analysis of graphical representation of residuals is coupled with the visualization of the results themselves, topic of the following section §3.2.3.

Eventually, after all the previous tests, the best choice for the number of components appears to be three.

3.2.3 Visualization of results

3.2.3.1 Scores

As previously mentioned, scores correspond to the projection of each observation on the new basis vectors, i.e. the loading vectors (or "principal components"). Along the first principal components, a significant part of the total variability is captured (98%), as shown in the previous paragraph (§3.2.2.2). Therefore, a first separation of the data, corresponding to different objects in the picture, should be recognised in scores plots, that may plot the score vector related to one component against the "observations" (i.e. different pixels) or may represent variation of scores relatively to one another in a scatter plot. It will be clear that some plots can be more useful than others, and that for the specific case of a hyperspectral image there could be some other ways to "plot" the dataset, i.e. in the form of an image.

The representation of the scores for one principal component for each observation produces a plot of really difficult interpretation, being the abscissa's length, equal to the total number of observations ($640 \cdot 480 = 307200$), the major obstacle to the ability in the discernment of the different points (Figure 3.11)²⁵. Zooming into small sections (Figure 3.12) and remembering that each 640 points a new column of the picture starts²⁶, interpretation of the data is made easier. It can be deduced that the descending and ascending series of points that make Figure 3.11 appear very chaotic are the transition from an object to another or to the background. Classic identification of data clusters by eye inspection is in this case very difficult with this kind of plot. In fact, being the dataset the result of the unfolding of a picture, the best way to visualize scores results against all the observation is rearranging the scores into the shape of the original picture ($640 \cdot 480$) and rescaling the scores between 0 and 1, plotting a the resulting matrix as a greyscale picture. Figure 3.13 shows the greyscale conversion of a random wavelength picture and of the scores images for the first three components. It can be noticed the different contrast of the scores pictures compared to the random picture extracted

²⁵ The plot is obviously unreadable and is here reported only for the sake of completeness.

²⁶ The data is unfolded along columns, i.e. in a way such that at the end of each column is "attached" the beginning of the next one.

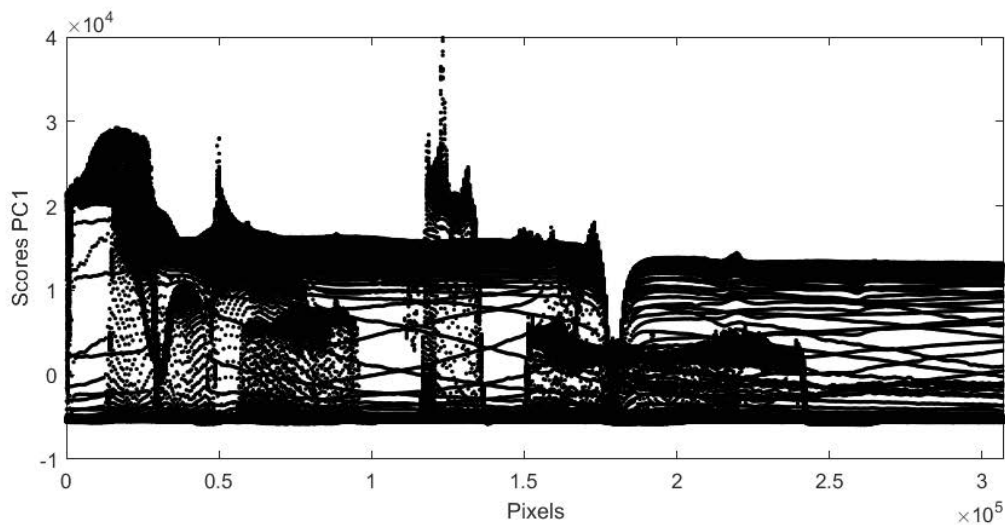


Figure 3.11 – Score vector for the first principal component against pixel indexes

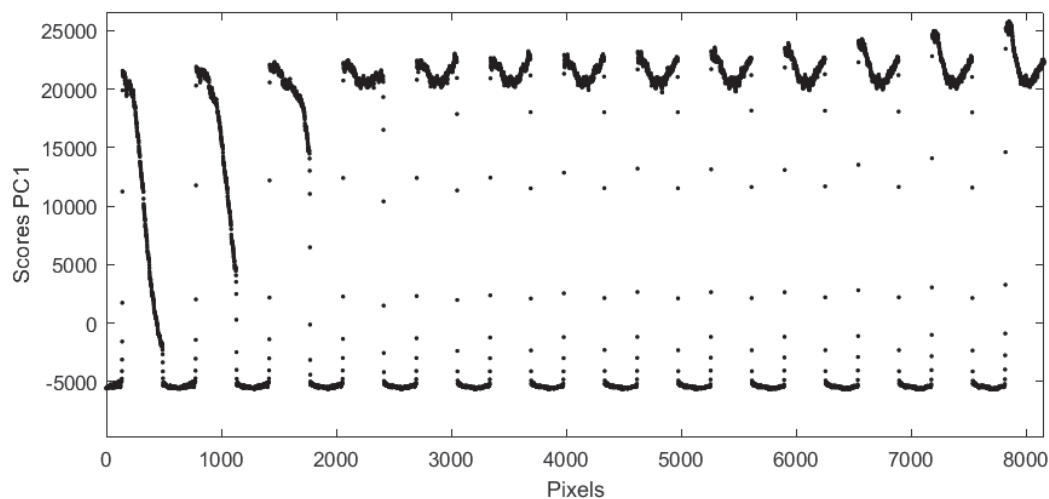


Figure 3.12 – Score vector for the first principal component against pixel indexes (first 8000 points)

from the hyperspectral image (corresponding to one spectral band). Different objects are better highlighted in different pictures, giving a glimpse on the classification results obtainable.

The number of principal components chosen gives additional advantage for graphical representation and human eye inspection. In fact, the three pictures corresponding to the first three components can be concatenated and plotted as if the three matrices were the red, green and blue channels of a colour picture (RGB colour space). For this purpose, the concatenated matrix was normalized in the $[0 \ 1]$ interval to avoid automatic saturation of lowest and highest pixel values performed if `imshow` is applied without this pre-processing operation. The resulting picture is shown in Figure 3.14 (left). The aim of PCA was building

few latent variables able to capture all the information needed for classification, and the different colours of different objects in this picture, capturing differences in materials, already confirm the effectiveness of the method. Contrast enhancement through a contrast stretching transformation (Figure 3.14 – right) improves the capability of the human eye to appreciate these results. Knowing basic notions about colour maps, information about the

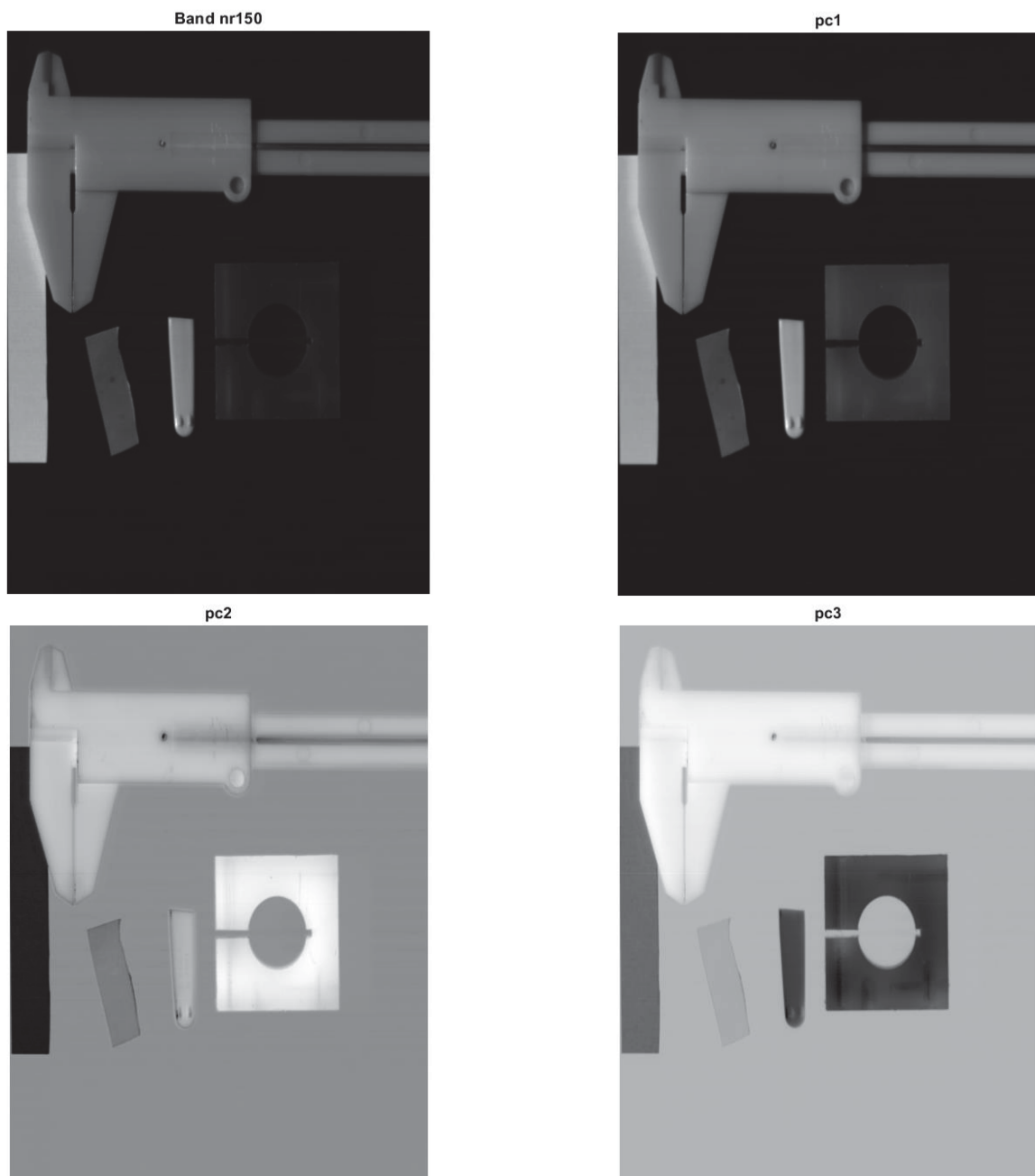


Figure 3.13 – Greyscale conversion of data corresponding to (1) a random wavelength, (2) score vector for the first principal component, (3) score vector for the second principal component, (4) score vector for the third principal component (images are ordered from left to right and from top to bottom)

principal component(s) on which object(s)'s points have higher scores can be inferred by the dominant colour of the object itself (Table 3.2). Referring to the numbering in the greyscale photo of the configuration (Figure 3.2), object (1), for example, can be said to have high value for the scores on PC1 and low value for scores on PC2 and PC3. In other words, object 1 is “along” the direction of PC1. Similar considerations can be made for the other objects. Scatter plots representing scores for one principal component against scores for the other one are also possible. An example is the one in Figure 3.15, considering the first two components. Observations close to each other on the scatter plot are points that are “close” to each other also in the actual data, i.e. their spectral response is similar, being most of the variation already represented with the first two components. Dashed ellipses enclose regions where clusters of data appear clearly. Comparing data represented in Figure 3.15 with Figure 3.2 some data clusters can be matched with the corresponding object. For example, object (2) has high scores on PC1 (bright on PC1 image) but low scores on PC2 (dark on PC2 image), therefore the cluster in the lower left corner of Figure 3.15 can be marked as “object (2)” with no doubt.

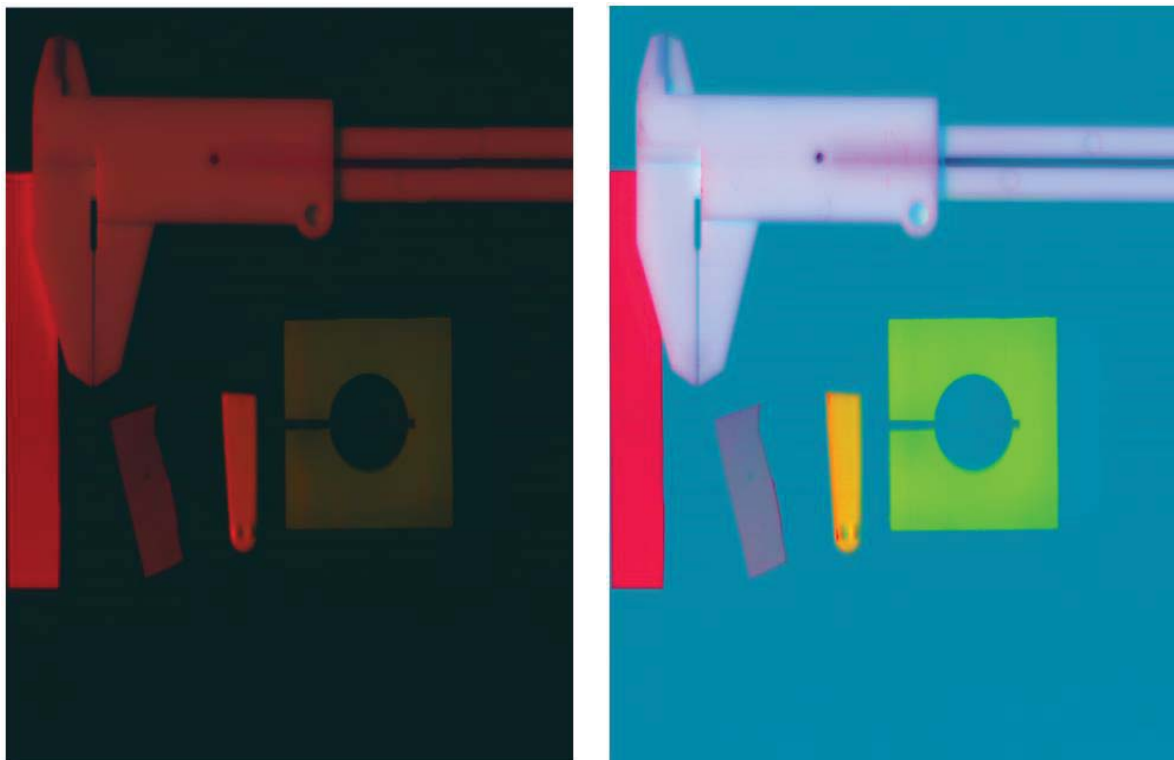


Figure 3.14 – (left) Combination of the first three components into a RGB picture; (right) Contrast enhancement for the RGB conversion

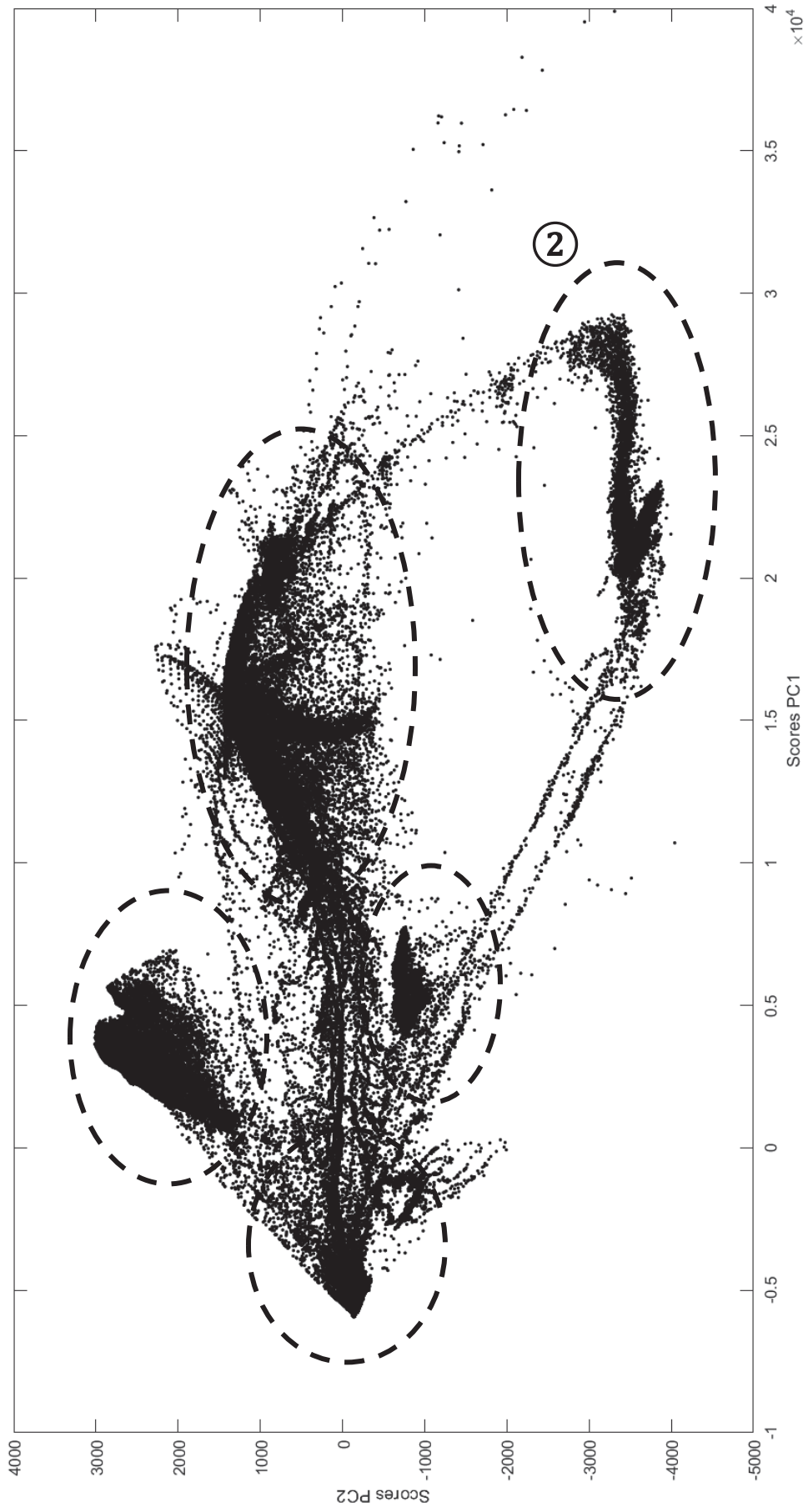


Figure 3.15 – Score vector for the first principal component versus the one for the second principal component







Colour of the object	PC
Red 	1
Green 	2
Blue 	3
Yellow 	1+2
Purple 	1+3
Cyan 	2+3

Table 3.2 – Correspondence between colours and principal components on which the scores are higher

Notice that the objects contour, i.e. the transition between objects or between object and background, is not sharp, but the points connecting the clusters indicate that the transitions are somehow smoothed. The other clusters are quite close to each other, therefore it is harder to mark them just by eye inspection of Figure 3.2 and Figure 3.15.

3.2.3.2 Loadings

Each element of loading vector is the weight that the corresponding variable has in defining the new latent variable (i.e. the principal component corresponding to that loading vector). Visual representation of the loading vector against the spectral bands enables to qualitatively check which variables influence more each principal component (Figure 3.16). The most noticeable feature of the loadings of the first three components is that none of them gives importance to the first wavelengths measured and the last ones. The problem might be that, being at the end of the spectrum measurable with the camera, the quality of the data is lower and the images for these bands are very noisy and do not contain useful information. This can be confirmed by examining the greyscale conversion of the first and the last picture of the “stock” in the hyperspectral image, corresponding to the first and the last wavelength bands (Figure 3.3).

Further inspection reveals that the first principal component appears to get significant amount of information from each one of the non-extreme variables, though for the higher wavelengths the contribution is less than for the first half of the spectrum. More complex is the profile for the second and the third component. For example, wavelengths in the middle, from the 70th band to the 100th band, have high (positive) influence on PC1 and PC3 but low (negative) weight in PC2, and this is the only region where the profiles have this behaviour relatively to one another. Therefore, “objects” – or better “materials” – having high positive scores on PC1 and PC3 and low negative scores on PC2 are best described by the wavelengths having more influence on PC1 and PC3 and are also having peaks on bands that have negative weight on

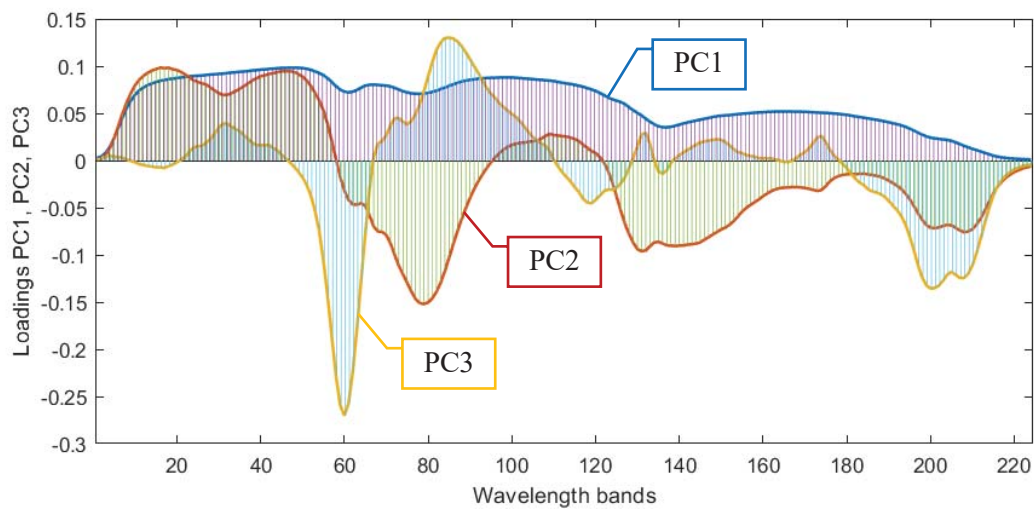


Figure 3.16 - Loading vector for the first three principal components for each spectral band

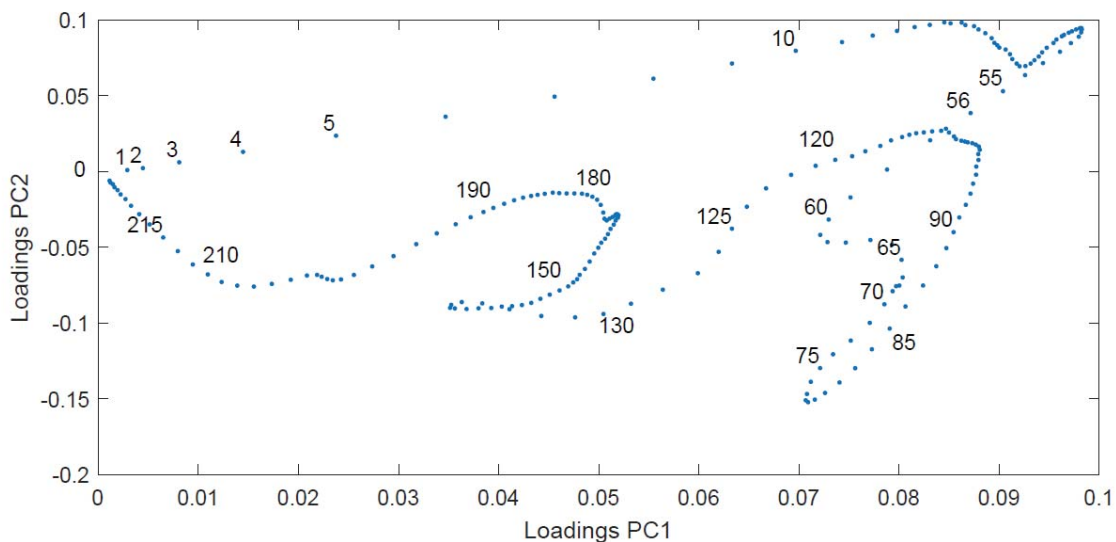


Figure 3.17 – Scatter plot for the loading vectors of the first two components

PC2. Analogous reasoning can be done for relative behaviours in different band regions. Associating high scores on some components (or on the opposite of those components) with the corresponding band contribution, different materials could be linked to peaks of reflectance in specific bands or spectral regions.²⁷

Loadings can also be analysed through a scatter plot (Figure 3.17). For spectral data, since two variables correspond to two contiguous wavelength bands, the loading plot has the shape

²⁷ Notice that if spectral response of different materials were known from previous studies, classification would actually proceed from comparison of the response of each material with the response of specific objects in the picture, i.e. in a somehow “reverse” way

of a continuous curve, as can be noticed checking the labels included for some points in the plot. Therefore, as predictable, bands near each other in the spectrum are going to co-vary, in general, and so are also close to each other in the scatter plot for loadings. This is particularly true for the case under examination, since even just the first two components explain most of the variation of the “sample”. In general, two variables close in the scatter plot of the loadings are considered to co-vary only with respect to the percentage of total variability explained by the principal components (i.e. loadings) themselves. It is also confirmed what already observed in Figure 3.16, that is the scarce contribution of first and last wavelength bands to the first principal components.

3.2.3.3 Residuals

As mentioned in the previous section §3.2.2, a smart way to represent residuals for an image is rearranging them in a matrix of the same dimension of the image itself. Qualitatively, the error matrices for all the PC projections appear similar to the one in Figure 3.18 (left). Predictably, the most significant error is committed along the boundaries of the objects. As for the scores, also residuals can be concatenated and showed as a colour picture where each PC corresponds to one of the three channels in the RGB colour space. Again, colour representation can significantly improve human eye interpretation of the results. For example,

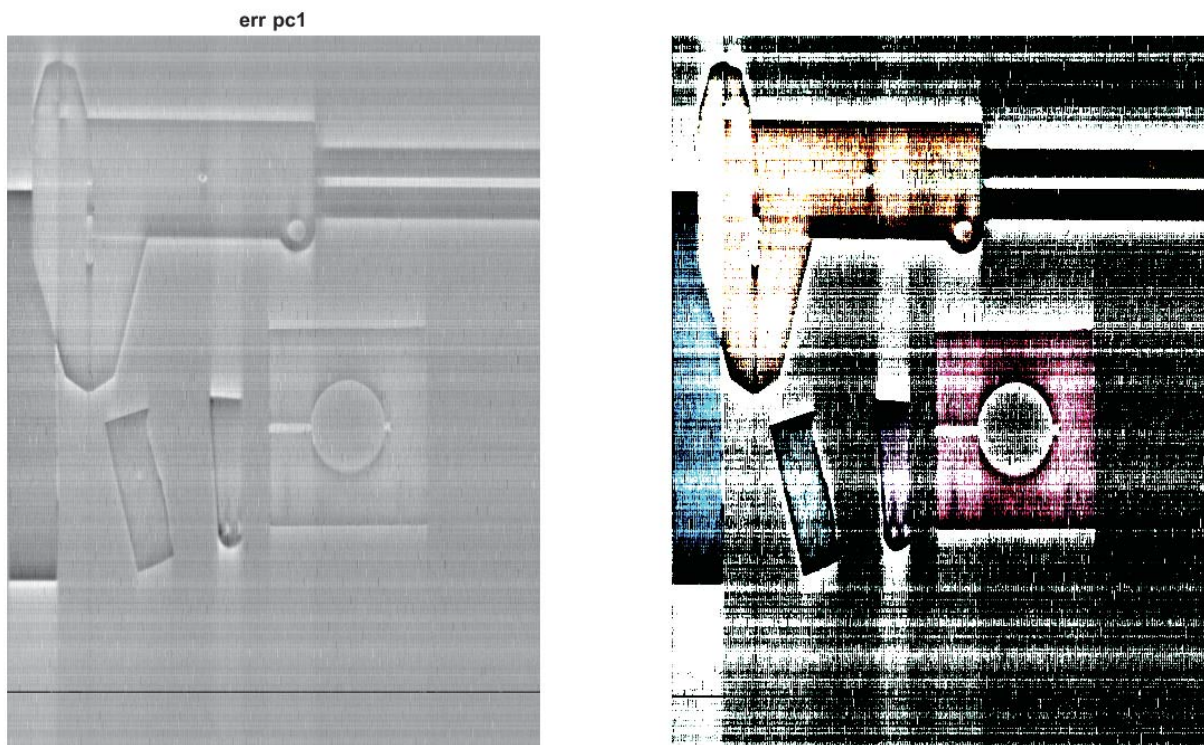


Figure 3.18 – (left) Residual matrix for the first principal component; (right) representation of the three residual error matrices as a colour image

it is clear that for object (2)²⁸ the points have low(er) residuals for projection on the first two components and higher residual for the third, therefore it appears blue in Figure 3.18 (right). Similar conclusion can be drawn for other objects.

3.3 Wavelet transform

For the principal component analysis, algorithm implementation does not present particular difficulties, even when opting for step by step calculations instead of the build in MATLAB[®] function `pca`. Wavelet analysis is more complex to realize, therefore an entire MATLAB[®] toolbox is available for purchase²⁹. Nonetheless, if someone wishes to implement custom functions able to perform the wavelet transform without the need of the toolbox, Gonzalez *et al.* (2010) provide useful function drafts in the dedicated chapter. In addition to these options, another valid possibility is the free toolkit implemented by the Department of Statistics at Stanford University, known as WAVELAB 850³⁰. The results in this section are obtained using this last toolkit.

3.3.1 Pre-processing

In order to be able to use the WAVELAB 850 toolkit, the score matrix correspondent to each principal component needs to be cut into a square matrix. As mentioned in section §2.4.2.2, this requirement is quite standard in bidimensional wavelet analysis, making possible an overall representation of the results as the one in Figure 3.19, and it does not pose any problem for the specific image under examination since the lower part of the picture is mostly occupied by the background, and there is already enough visible background in the upper part of the image to be able to further process the square cut of the image without affecting the applicability of the results to the whole picture in a second stage. In case of a rectangular image for which no portions can be ignored because important details are distributed overall the image itself, it is common practice to just cut the rectangle to obtain several squares covering the whole picture or attach to the long side of the rectangle a copy (or a mirror-copy) of the image itself and cut the final result in a square.

3.3.2 Visualization of results

As anticipated in section §2.4.2.2, applying a bidimensional wavelet transform results in a matrix of approximation plus three matrices of details (horizontal, vertical, diagonal) at each step. Figure 3.19 shows the coarser resolution approximation image after three decomposition

²⁸ See Figure 3.2 for object indexes

²⁹ See www.mathworks.com/products/wavelet.html (last access: 07/2019)

³⁰ See <https://statweb.stanford.edu/~wavelab/> (last access: 07/2019)

stages and the different detail matrices corresponding to each step. To be able to represent the data in a greyscale image, each detail matrix and the approximation matrix have been independently rescaled to the range $[0,1]$; for easier human eye inspection, contrast-stretching and saturation of the bottom and top 1% of the pixels was also performed for each matrix independently.

In the traditional configuration of the decomposition results, the images in the upper-right corner (i.e. the ones above the diagonal) correspond to the horizontal details. The horizontal

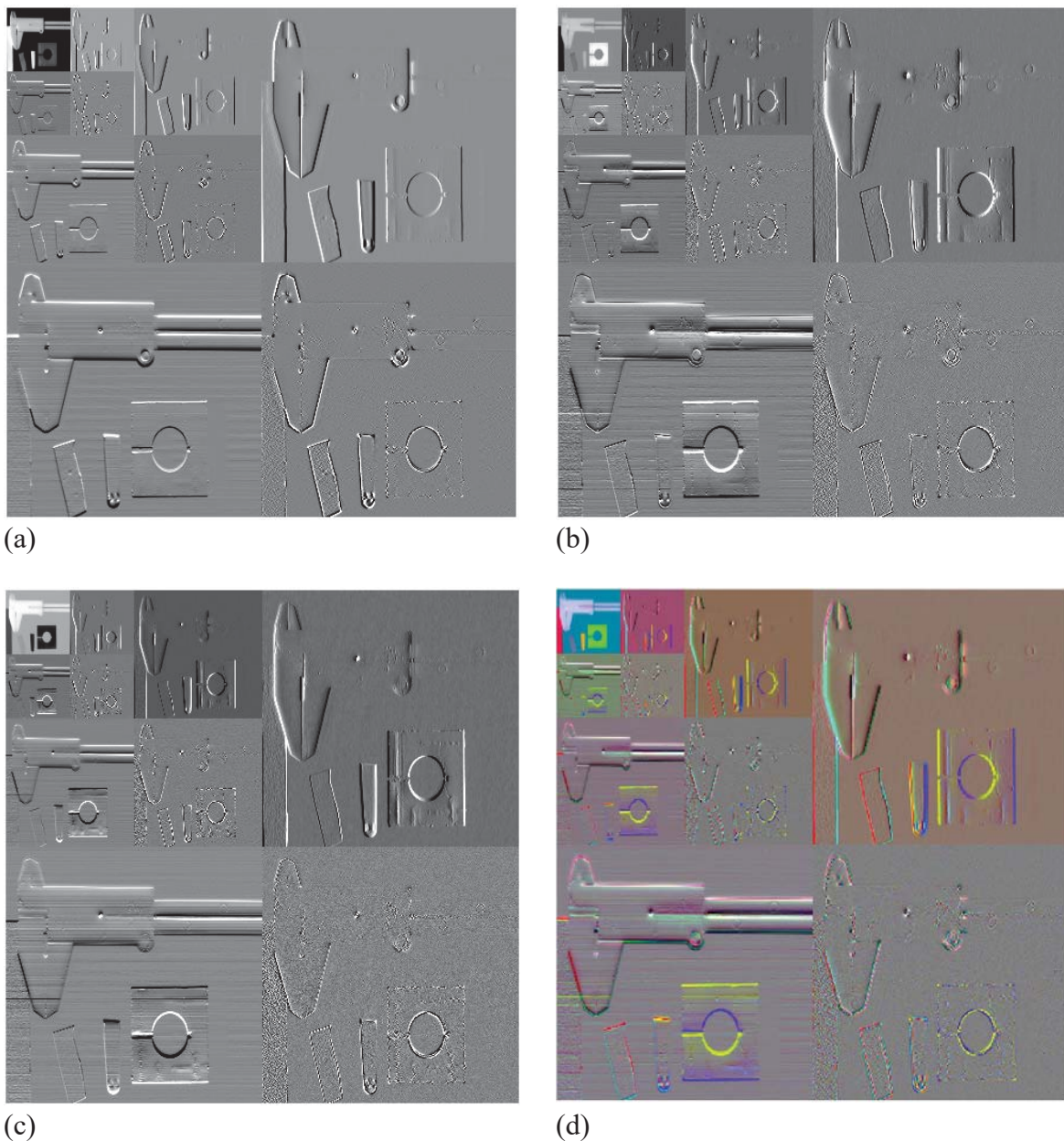


Figure 3.19 – Decomposition resulting from the application of a two-dimensional wavelet transform to the score matrix of the first component (a), the second (b), the third (c) and a joint representation of the three score matrices in rgb color space (data rescaled and contrast stretched).

detail matrix d_H stores the information that is “loss” after the application of a horizontal transform. Lower and upper peaks in the horizontal detail images corresponds to the vertical edges “crossed” (and, in a way, “detected”) during the translation of the basis functions along rows, to obtain the approximation (and as a consequence also the detail loss). Similar reasoning applies to the vertical detail matrices, situated below the diagonal.

Observing vertical detail images, the horizontal waves discussed in section §3.1 appear clearly. Since, during the vertical transformation of the matrix, the Haar scaling function approximates two near points with one averaged value (see Figure 2.12), if the intensity profile along columns is wavy the detail loss for each pair of points will be more significant than in the case of a more smooth intensity profile. Similarly, in correspondence of peaks the approximation will be significant and will lead to high detail coefficients (i.e. peaks also in the detail matrix), that corresponds to higher amount of detail loss in the approximation. Figure 3.20 exemplifies the problem for a one-dimensional wavelet transform. Figure 3.20 (a) and (b) represent a scaling function (father wavelet) and a wavelet function (mother wavelet) that can be used to expand the function in (c). Figure 3.20 (d) represents the approximation (obtained multiplying the scaling function for the approximation coefficients) and compares it to the original function; the first two points, having very different intensities, are approximated with significant “error”, i.e. higher detail loss (e) if compared to the subsequent couple of points. The detail in (e) is obtained multiplying the mother wavelet for the detail coefficients, that are stored in the “detail vector” (or, in two dimensions, “detail matrices”). Similar situations, in two dimensions, lead to peaks and valleys in the detail matrix. Besides this effect on the detail matrix, this approximation by averaging values of two contiguous points, obtained using Haar basis functions, results also in an overall smoothing/denoising of the signal, clearly visible in Figure 3.21, especially by comparison with Figure 3.4. In fact, the approximation (Figure 3.22 – right) obtained after the application of the wavelet transform is $2^{\mathcal{F}}$ times smaller than the original square image “entering” the processing, in this case, the maximum level of decomposition \mathcal{F} is equal to 3, and so the approximation is 8 times smaller than the input matrix. Therefore, the 90th row from Figure 3.4 corresponds to the 11th row in the approximated image ($90/8 \cong 11$), and similarly the 155th column becomes the 20th column in Figure 3.21. Notice the intensity axis going from 0 to 1 since the transformed image in Figure 3.22 (right) has been rescaled to be able to represent it using `imshow`; moreover, the image contrast was enhanced to allow the human eye to better appreciate differences in intensities on the three retained components, similarly to the visual enhancement operations performed on the PCA scores for the first three components, shown in Figure 3.14. Figure 3.20 (f), finally, illustrates the reconstruction obtainable with inverse transform if the detail is stored, adding the detail vector to the approximation vector. Notice how the approximation matrix stores the function approximation, while the detail matrices store only the coefficients

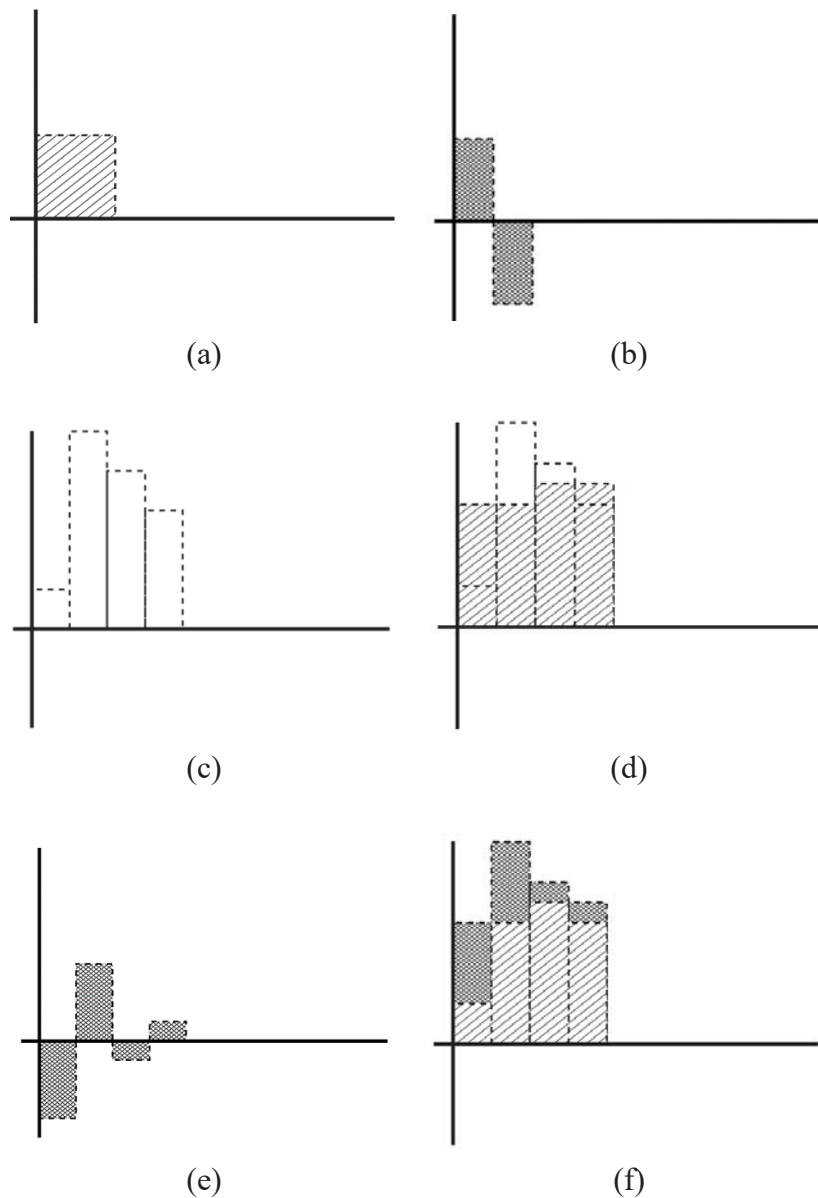


Figure 3.20 – Example of wavelet expansion using the scaling function in (a) and the wavelet function in (b). In order: (c) original function, (d) approximation, (e) detail, (f) reconstruction from approximation and detail

by which the mother wavelet is to be multiplied to obtain the reconstructed image in (f). Finally, notice that the abovementioned waviness, registered for intensity profile along columns, appears to affect not only the background but also the objects, as can be deduced observing the detail matrix, where horizontal bands (corresponding to peaks in the detail loss) cover the whole picture, including the objects. This can be more easily detected after saturating the decomposition results corresponding to one principal component (e.g. the first one), obtaining Figure 3.23. Therefore, even if for the background the “waviness” may appear more pronounced due to the physical structure of the background itself (Figure 3.5), also the

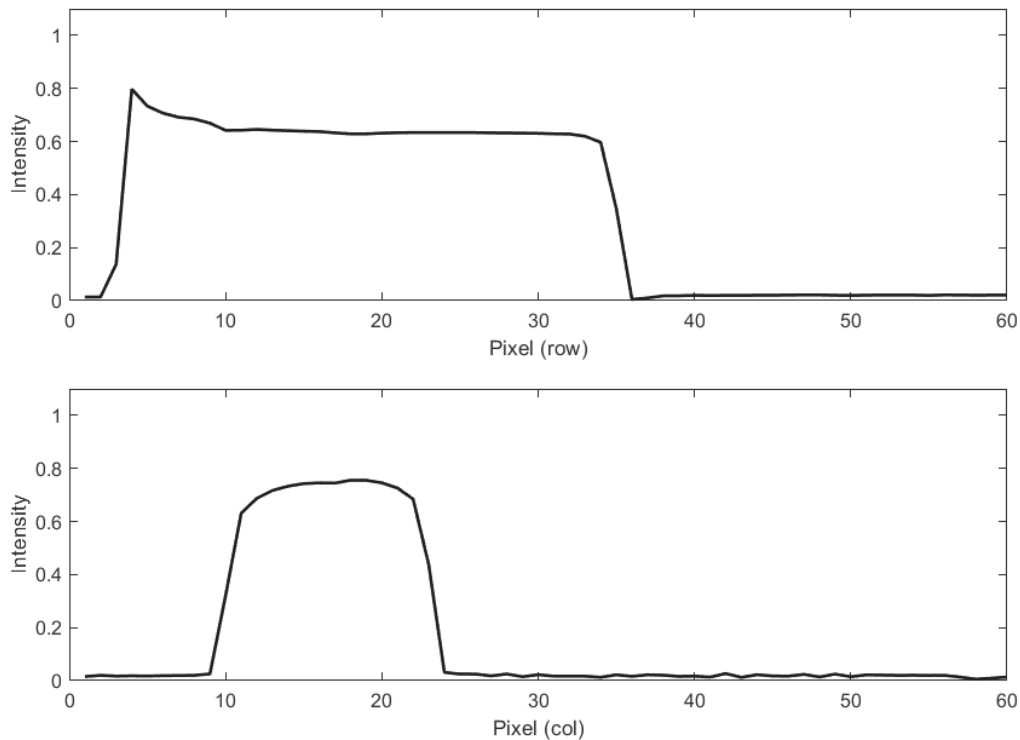


Figure 3.21 – Intensity profiles of row 11 (up) and column 20 (down) for the 1st pc scores after wavelet transform application, followed by rescaling and contrast-stretching

detection technique, capturing a row at a time, may have influenced the quality of the intensity profile along the columns (Figure 3.4), as pointed out in section §3.1.

As concerning the diagonal details, for this specific picture they are not of particular interest, resulting mostly in white noise because of the absence of diagonal edges.

The low-resolution version of the original image (Figure 3.22 – left) appears to be a good approximation of the original image, even if the edges are not smooth as in higher resolution (Figure 3.14). Notice that wavelet analysis was performed on the original data matrix, and not the rescaled, higher contrast version in Figure 3.14 (right), so that the low resolution hyperspectral image is actually a smaller dimension approximation of the original score vectors (rearranged in the shape of the original picture), and therefore certain types of analysis performed on the approximation can be extended to the high resolution scores image. In fact, after clustering an implementation of a linear classifier, if the classifier is built using the non-rescaled picture as input data, then it can be used to classify the high-resolution score matrix (or equivalently the corresponding image obtained after reshaping) without further processing.

Figure 3.22 (left) can be obtained after three decomposition stages. It is possible to reduce even more the dimension of the dataset, the inferior limit of the number of decomposition stages (nds) being $\log_2 N$ for a square image whose size N is equal to a power of 2, or otherwise the lowest number for which $N/2^{nds}$ is an integer. In this case, $N = 480$, therefore

the maximum nds is 5. Figure 3.22 (right) shows the approximation obtainable after five decompositions. Clearly, the resolution is so low that it is almost impossible to give a meaning to the pixels in the image and to recognize the presence of objects and their nature, especially for the smaller objects that merged almost completely with the background.

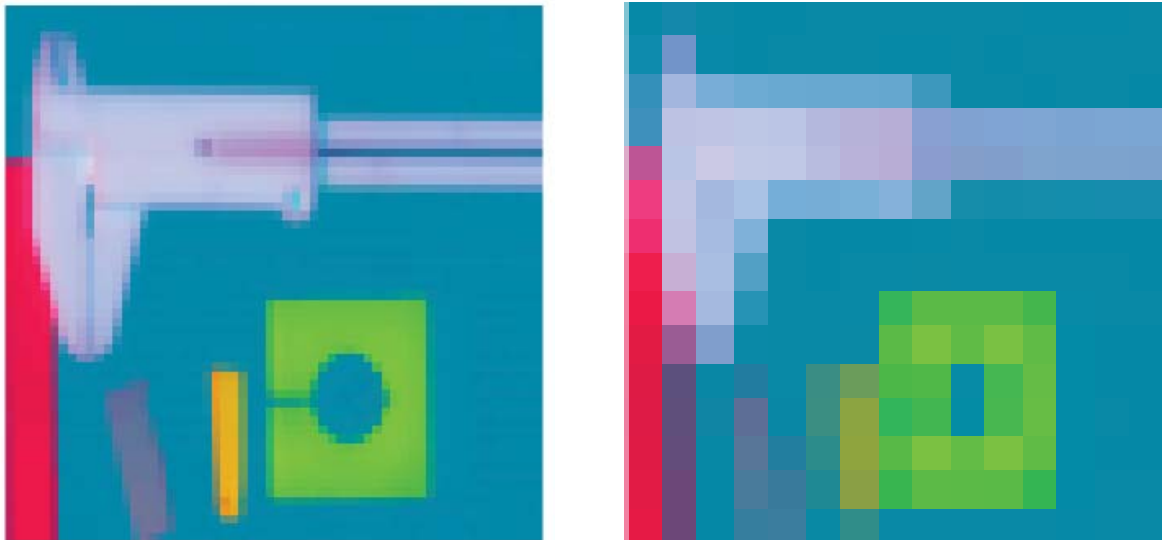


Figure 3.22 – Approximated image obtained after wavelet transform application, followed by rescaling and contrast-stretching. Respectively: three decomposition stages (left) and five decomposition stages (right). The images have been zoomed.

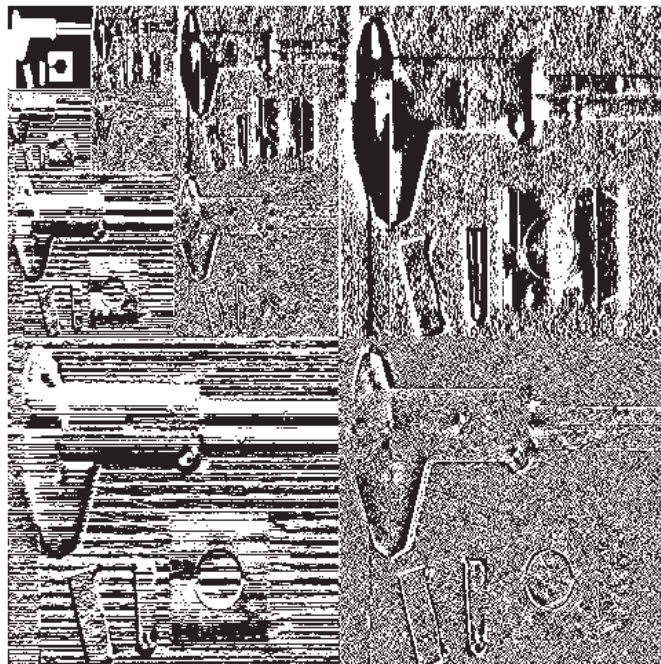


Figure 3.23 – Decomposition resulting from the application of a two-dimensional wavelet transform to the score matrix of the first component. Results are saturated after 1 and below 0.

An image where at least two objects faded away and bigger objects can have a number of points not blended with the background as low as four, e.g. the green square on the bottom, is of little to no use for classification purpose, since, for example, the category to which the two smallest objects belong is not represented, and even for bigger objects, like the green square, just very few points (four) represent what actually is the most popular intensity value for the same object in the high resolution image, and this could degrade the performance of a classifier trained on a dataset with such a small number of representative points. Since the objective is the subsequent clustering and classification, it is of paramount importance maintaining a sufficient number of points which are representative of the bulk of each object. Consequently, the acceptability of a level of detail is actually determined by the effectiveness of the classifier built that can be verified only a posteriori, even if eye inspection can guide the initial choice, e.g. leading to the exclusion of Figure 3.22 (right).

3.4 Clustering

As mentioned before, the final objective of this work is to build a tool able to identify and label different materials, as the ones of which are made the objects in the hyperspectral image under examination. If a clustering task was to be performed on the original 224-bands hyperspectral image, the time required would have been prohibitive (see section §3.5), for the high computational cost – in the case of hierarchical clustering, the dependence on the number of points is more than quadratic. Conversely, grouping points of a 60x60x3 image is relatively fast, even though not necessarily easy. In fact, the pre-processing makes feasible the use of the hierarchical method, due to the low dimension of the “new” dataset entering the clustering process.

3.4.1 Pre-processing

Before performing the calculations, it might be enlightening to take a look at the scatter plot of the wavelet-transformed score vectors (*w_t score vectors*), shown in Figure 3.24. It is clear that along some directions the data are more “well-spaced” than along others, i.e. clusters are further along some direction. Nonetheless, the more the observations belonging to different groups are “far” from one another (i.e. dissimilar, according to the metric used), the easier clustering will be. To enhance the distance between data points of different clusters, expanding the input range in a wider output range, saturating very low and very high values might improve clustering results, also considering that some of these extreme values are actually just noisy bridges between the clusters (i.e. smooth transition between objects, as seen in section §3.2.3.1).

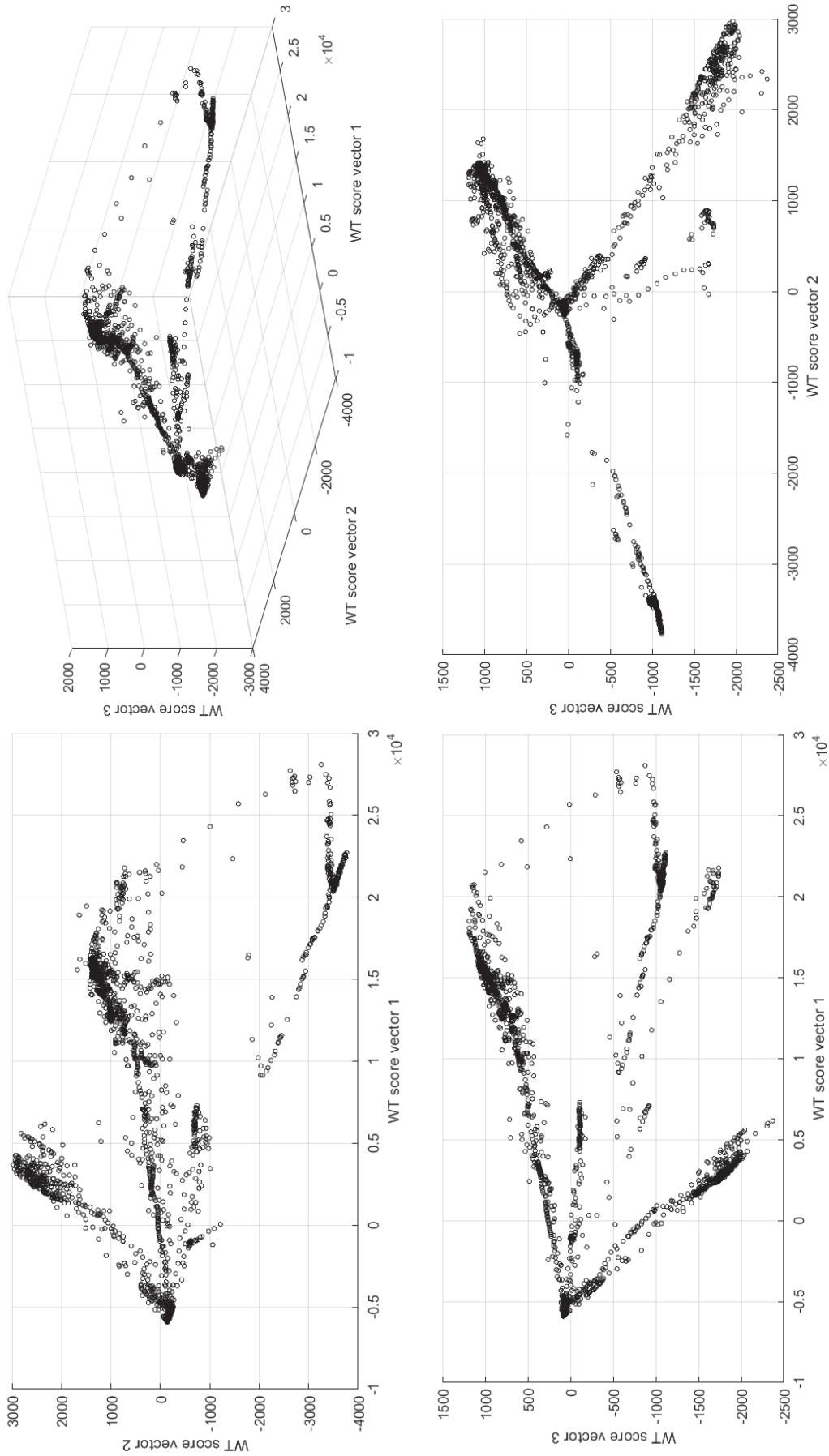


Figure 3.24 –Scatter plot for wavelet-transformed score vectors

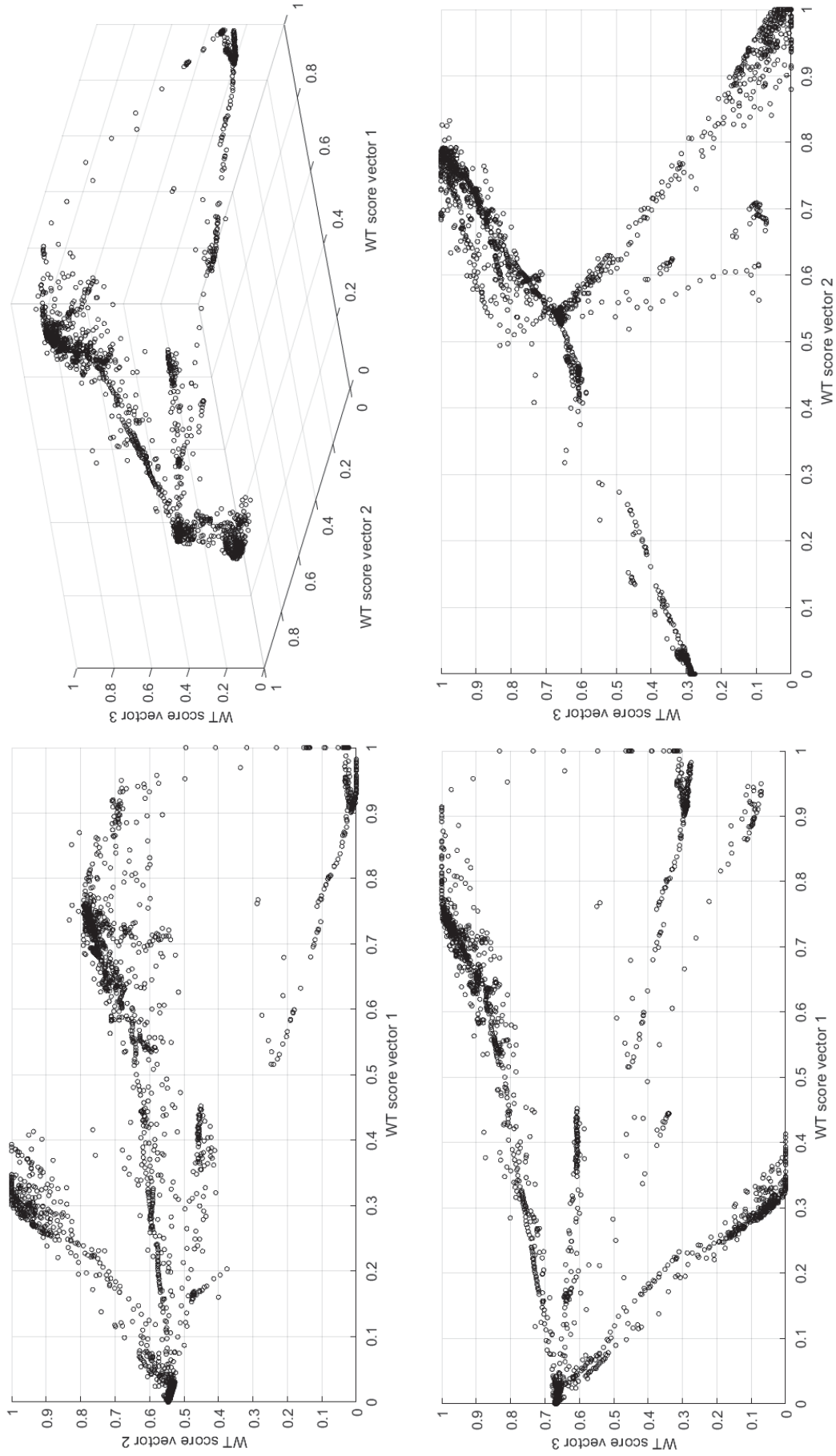


Figure 3.25 – Rescaled and contrast-stretched scatter plot for wavelet-transformed score vectors

This contrast-stretching operation, already mentioned at the end of the previous paragraph §3.3.2, saturating the bottom and top 1% of all pixel values of each *wt score vector* allows to significantly increase the distance between clusters, making them even more recognizable, on the scatter plot, by the human eye. Appendix 3 – Rescaling and contrast-stretching explains with a practical example how rescaling and contrast stretching does not modify the clusters naturally present in the data, but improves the distinguishability of some of them.

3.4.2 Visualization of results

The “data abstraction” step (ref. section §2.5.2.1), necessary for human interpretation of the results, is a very easy task if the dataset is an image: each cluster is associated to a number from 0 to 1 and the results are plotted in the shape of an image with a high contrast colormap (Figure 3.29).

The improvement obtained through the pre-processing is confirmed by the several results collected in Figure 3.28, where a side-by-side presentation of the groupings obtained shows how in general the image on which distance-enhancement pre-processing was applied is better classified for almost all the metrics considered.

Among the distance metrics available in MATLAB[®], given the configuration shown in Figure 3.25 where most of the clusters seems to be reasonably compact, the simple Euclidean distance could be more than enough for the purpose. Metrics like the Chebychev distance (that considers the maximum coordinate distance), the Cosine distance (referring to the cosine of the included angle between points), or the Hamming distance (considering the percentage of coordinates that differ) seems inappropriate for the application, as well as some other metrics, like the Mahalanobis distance (considering the covariance matrix) and the Correlation distance (considering correlation between points), seem unnecessarily complicated. Therefore, the Euclidean distance metric seems the most appropriate. Setting the metric to ‘euclidean’, the algorithm can be run several times considering the options available for the linkage metric. All the options available in MATLAB[®] are compatible with Euclidean distance. Some options can be excluded by simple reasoning, like the Simple linkage, considering the inter-cluster distance to be equal to the shortest distance between two points of the two clusters, or the Complete linkage, that considers the furthest distance, being too simplistic to define properly inter-cluster dissimilarity. Median, centroid or weighted linkages seem to be more appropriate, nevertheless the results are not the best. Another promising option is the Ward’s linkage, that considers as an inter-cluster “distance” the increase in the within-cluster variance after merging, and links the two clusters that minimize this objective function. In practice, this option seem to give the best results in term of object borders more polished and domain for each object compact enough (in the sense that the object does not contain points classified as a material different from the object’s material

itself). The only problem that seems to be present is a misclassification of the shadowing surrounding some objects more than others, as happens for the original image as well. This issue can be easily visualized through a scatter plot (Figure 3.27, left) where points corresponding to a specific group from Figure 3.27 (right) have the same colour as the clustering result image. It is clear that the “yellow” group is heavily dispersed, since just part of those yellow points in the scatter plot actually correspond to the small rectangular object coloured in yellow in Figure 3.27 (right), the others being the misclassified blurred borders of some other objects in the picture. The problem arises also because hierarchical clustering is a *hard* clustering method (ref. §2.5.2.1), meaning that it must necessarily put a point in one of the groups, and if the blurred borders are not similar enough to the bulk material, a solution could be to increase the number of clusters (cutting the hierarchical tree lower), hoping that the shadowing will be classified as the 7th material. Figure 3.26 shows the dendrogram (Euclidean, Ward) cut after 6 clusters, and the two crosses indicate the links that would be sequentially broken if the dendrogram cut-off was 7 or 8 clusters. It is clear how the specific sequence of these cut is determined by a very small difference in the height of the two bridges. In both cases, they link two clusters that are quite near to one another, so it is not obvious that the very first “new” cut will lead to a separation of the cluster corresponding to the rectangular yellow object in Figure 3.27 from the yellow border of other objects. Figure 3.30 shows how this trick to try to classify the shadows as an additional “extra” material does not work properly for this problem, even increasing the number of clusters to 8. Therefore, before proceeding to the next step, that is building a classifier, some morphological operation on the border will be needed, to try to build a classifier able to give attribute labels even better than the clustering ones.

It is important to notice also how the “blue” material in Figure 3.27 corresponds to a cluster practically impossible to find by only human eye inspection of the scatter plot in Figure 3.24 and Figure 3.25. Similarly, it is impossible to associate one specific yellow area from the scatter plot Figure 3.27 (left) to the rectangular yellow object in Figure 3.27 (right). Remembering how the k-means clustering method starts from initial arbitrary centroids, there is a substantial likelihood of missing the yellow object or the blue object if these initial seeds are chosen randomly by the algorithm, and before finding all the objects a significant number of runs with random seeds could be necessary, *de facto* eliminating or significantly diminishing the supposed computational advantage of using a partition method. On the other hand, it is possible to give as input to the method initial centroids carefully chosen in the areas corresponding to each object, that could be identified checking the three “coordinates” of few points for each object identified by human inspection on the image. Nevertheless, this trick would make meaningless the concept of “unsupervised learning”, since the human intervention would be crucial for the success of the clustering task. Conversely, the

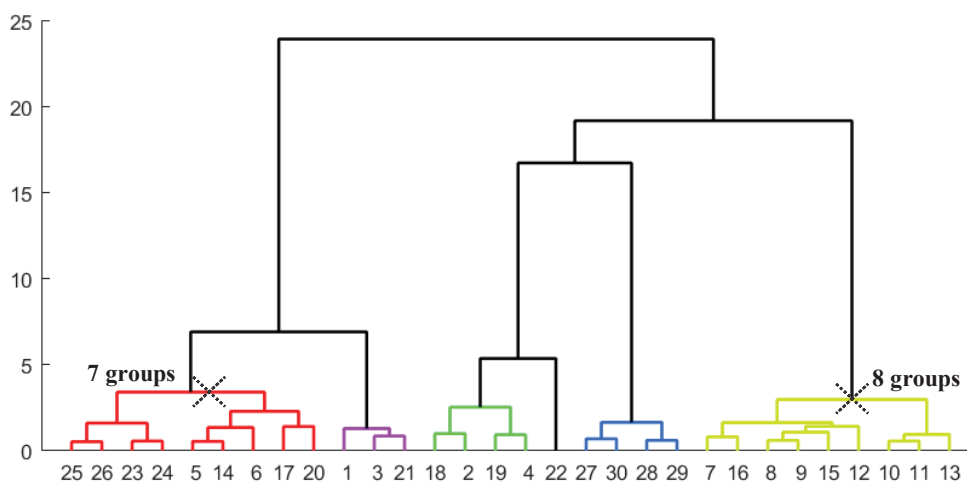


Figure 3.26 – Dendrogram for the clustering obtained with Euclidean metric and Ward’s linkage, cut-off after 6 clusters. With an x are marked the links that would break with a cut-off of 7 or 8 clusters. The colours are random and do not correspond to the colormap from Figure 3.29.

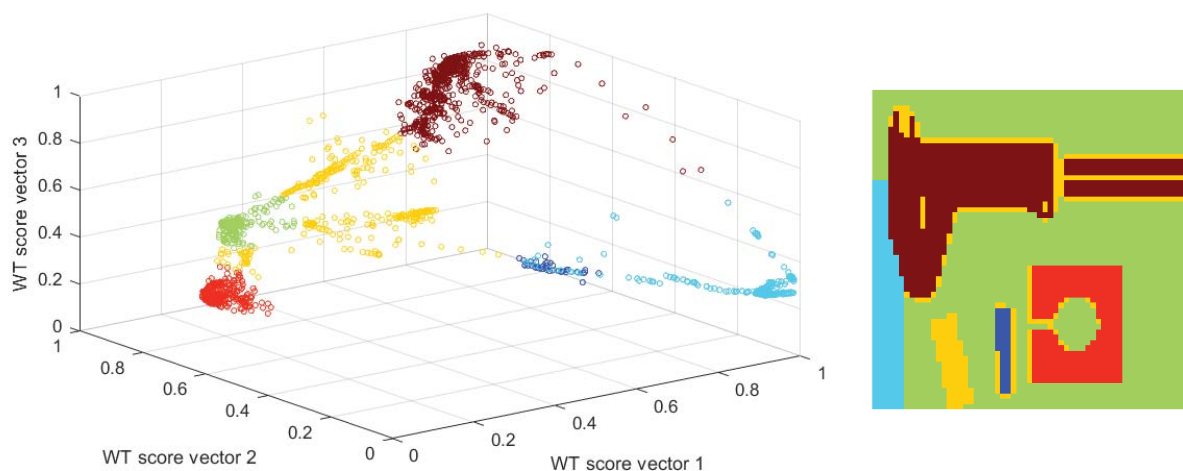


Figure 3.27 – (left) Scatter plot for the wavelet-transformed score vectors, rescaled and stretched. Colours corresponding to clusters on the (right) image.

hierarchical method is able to obtain these results without external intervention, confirming its robustness.

For comparison purpose, the Mahalanobis distance metric was also tested. Some of the linkage options are compatible only with the Euclidean metric, and of the remaining given the nature of the distance metric related to the covariance, the Complete or the Single linkages do not work properly, and the best results are obtained with the Weighted linkage, considering a weighted average distance between the covariance of the points as the inter-cluster distance.




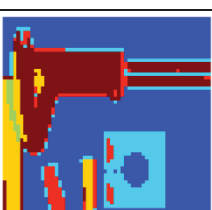
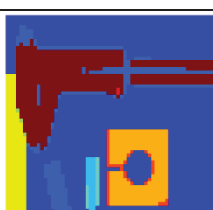
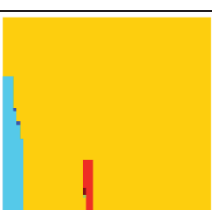
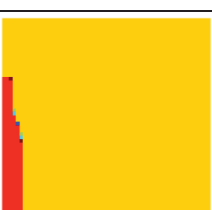
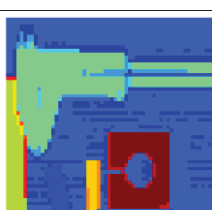
<i>Applied to the WT Image</i>	<i>Applied to the WT Image Rescaled and Contrast-Stretched</i>	<i>Applied to the WT Image</i>	<i>Applied to the WT Image Rescaled and Contrast-Stretched</i>
 <p><i>Euclidean Average</i> $c_h = 0.9448$</p>	 <p><i>Euclidean Average</i> $c_h = 0.9229$</p>	 <p><i>Euclidean Ward</i> $c_h = 0.9130$</p>	 <p><i>Euclidean Ward</i> $c_h = 0.9124$</p>
 <p><i>Euclidean Centroid</i> $c_h = 0.9442$</p>	 <p><i>Euclidean Centroid</i> $c_h = 0.9187$</p>	 <p><i>Euclidean Weighted</i> $c_h = 0.9324$</p>	 <p><i>Euclidean Weighted</i> $c_h = 0.9174$</p>
 <p><i>Euclidean Complete</i> $c_h = 0.9302$</p>	 <p><i>Euclidean Complete</i> $c_h = 0.9096$</p>	 <p><i>Mahalanobis Weighted</i> $c_h = 0.8333$</p>	 <p><i>Mahalanobis Weighted</i> $c_h = 0.9270$</p>
 <p><i>Euclidean Median</i> $c_h = 0.8887$</p>	 <p><i>Euclidean Median</i> $c_h = 0.8782$</p>	 <p><i>Mahalanobis Weighted 10 Clusters</i></p>	 <p><i>Mahalanobis Weighted 10 Clusters</i></p>
 <p><i>Euclidean Single</i> $c_h = 0.7221$</p>	 <p><i>Euclidean Single</i> $c_h = 0.7102$</p>	 <p><i>Mahalanobis Weighted 15 Clusters</i></p>	 <p><i>Mahalanobis Weighted 15 Clusters</i></p>

Figure 3.28 – Clustering results with different “distance” and “inter-cluster distance” metrics

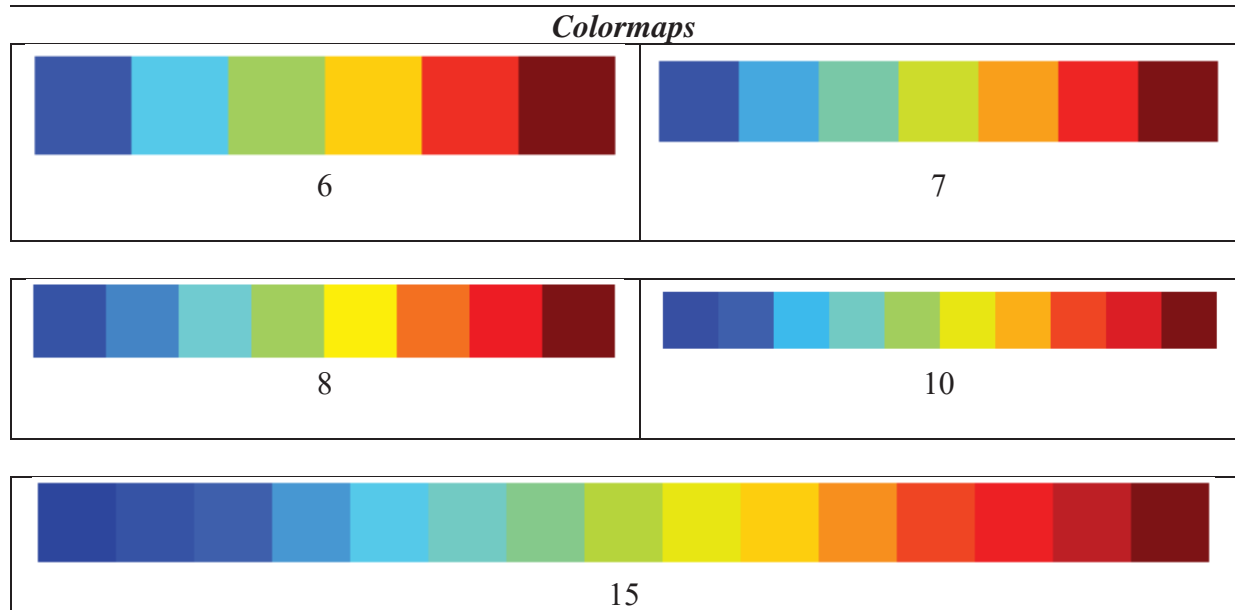


Figure 3.29 – Colormaps for the clusters in Figure 3.28. Each colour corresponds to a different group.

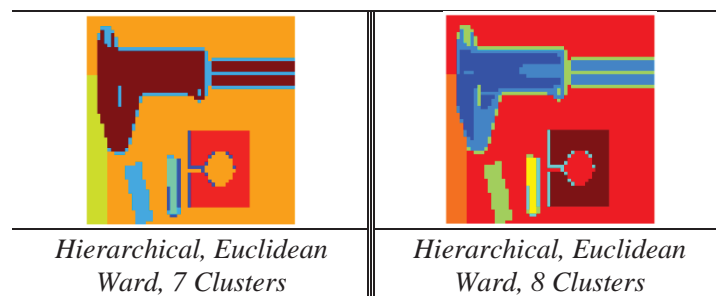


Figure 3.30 – Clustering results increasing the number of clusters required.

The results for the latter option, shown in Figure 3.28, seem to be way less than optimum, especially for the non-rescaled dataset, since an entire object “disappears” from the picture. Increasing the number of “natural” clusters requested, the object “reappears” only for the case of 10 total clusters starting from the rescaled dataset, and 15 total clusters starting from a non-rescaled dataset, and in the latter is considered part of the same group of the background waves (ref. background inspection in section §3.1), even if they clearly are not made of the same material. Considering the Mahalanobis weighted case with 15 clusters, the improvement in the classification of the stretched image with respect to the non-stretched dataset is even more evident than for the Euclidean metric cases, being the Mahalanobis metric not the best.

It is of considerable importance to notice how the cophenetic coefficient is very low, even on the stretched image, for the Single linkage, as expected due to the poor results, but surprisingly high for the Complete linkage on the non-stretched image, even higher than the

accepted better clustering (i.e. Euclidean Ward on stretched image), despite of the evident low quality of the results that can be discovered by simple eye inspection. Therefore, if a high cophenetic coefficient can contribute to confirm the meaningfulness of the results, it cannot be used as the sole indicator of a good clustering.

3.5 Classifier

After obtaining classes labels with the clustering operations, it is possible, finally, to build a classifier not only able to label the high resolution image, but also useful for future applications.

3.5.1 Pre-processing

As pointed out in the previous section, the labels matrix resulting from clustering operations can contain some misclassified points. In this case, misclassified points are represented by yellow points on the borders of objects that are actually made of materials different from the “yellow” one. Nevertheless, some morphological processing may improve the quality of this labelling. In fact, as shown in Figure 3.31 (a) the actual output from the clustering operation is a matrix with labels from “1” to “6”, that can be rescaled to a greyscale image simply dividing by 6. Therefore, after the division, the label matrix is de facto a greyscale image, and greyscale morphological operations can be applied. The color representation is obtained only via colormap editing, and so it is just a different way of representing the same intensity matrix. For this purpose, it is important to know the greyscale value of the label of each group, that is reported in Figure 3.31 next to the colormap. For the red and the orange objects eliminating the yellow border through dilation operation is not difficult, being their greyscale value higher than the “yellow” material one (Figure 3.31 – a). For the blue object a problem is posed, because it is so small that it will be “buried” by the yellow contour, being “blue” equivalent to the lowest greyscale intensity, as shown in Figure 3.31 (b). A good trick is to interchange the two labels, labelling as group 4 the blue object and group 1 all the yellow points, including the unwanted ones at the borders, as shown in Figure 3.31 (c). Through this operation, it is possible to apply dilation to cover the “dark” (i.e. low intensity) borders and expand the “light” (i.e. high intensity) areas corresponding to objects, and then apply erosion to get back the original size of the elements. The sequential application of these two operations is also known as *closing*, and in fact, among the consequences, it also tends to close small holes like it happens for the open bridge in the orange object and the holes in the red one (Figure 3.31 – d). The rectangular object originally labelled as material 4 is bigger than the rectangle originally labelled as material 1, therefore after the change of classes even

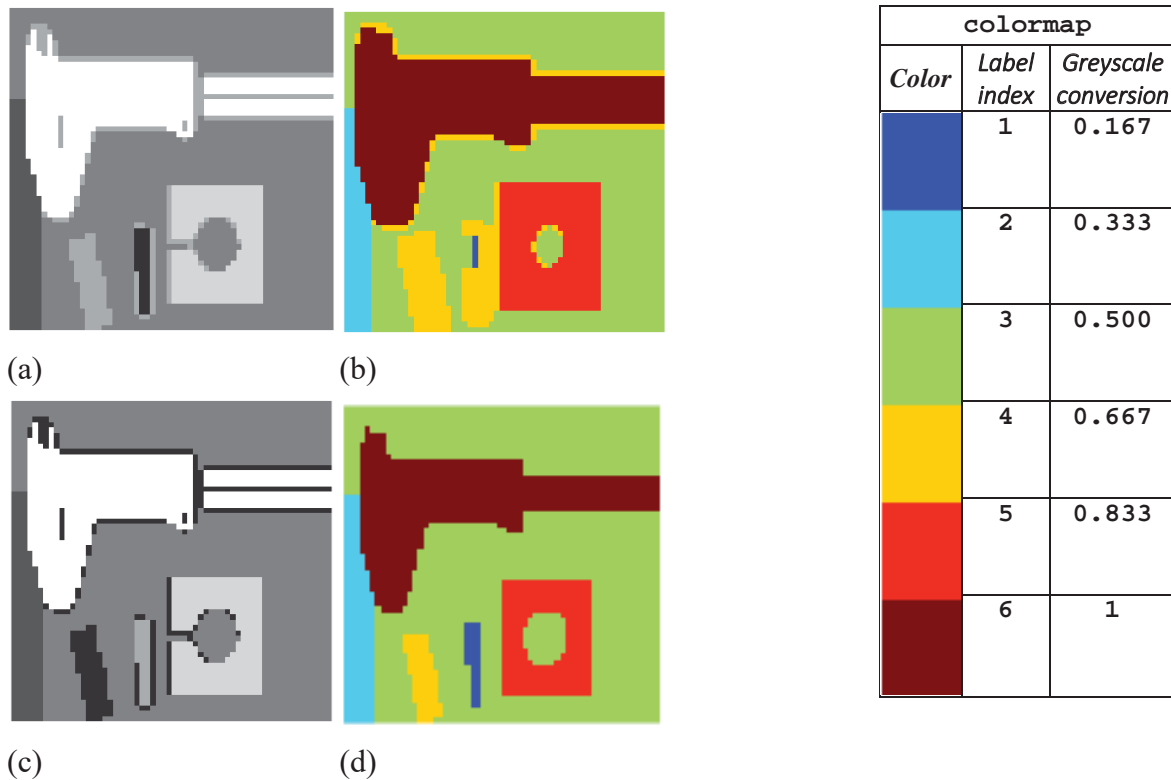


Figure 3.31 – (a) Greyscale clustering result; (b) Dilation on greyscale clustering result; (c) Labels interchange between groups 1 and 4; (d) Closing operation on the image with interchanged labels

if there was anyway an object of “dark” material, it was big enough to not disappear because of the dilation operation, and the subsequent erosion brought it back to its original size.

Notice that, after these operations, it is possible to give back the original “colors” to the groups, i.e. the original labels for the materials. However, this step is not strictly necessary since those labels are an arbitrary sequence of numbers created during the clustering operations, and keeping the same labels over all the analysis is only a consistency matter, but in reality for this specific problem interchanging the labels would not have significant consequences overall because the labels are not yet associated with real materials names.

3.5.2 Results

The first step to build a classification model is training the classifier. For this scope, MATLAB[®] provides a function specific for discriminant analysis, that is `fitcdiscr`. As mentioned in paragraph §2.6.3.2, the training dataset consists of: (1) the data points of wavelet-transformed low-resolution image (not enhanced), and (2) the labels obtained from the contrast-enhanced *wt*-image and subsequent morphological transformation of the clustering results. In fact, for better classification results, as mentioned, it is advisable to have a training dataset with labels as correct as possible. The results of the training are assessed

applying the classifier on the training dataset itself, as a test to check the quality of the produced labels with respect to “real” classes. To predict labels with a trained classifier, the MATLAB[®] function `predict` can be applied. Observing the results obtained training the classifier on a non-pre-processed dataset (from now on “*NPP classifier*”), where several points on the object borders were incorrectly labelled, they appear to be significantly worse than results obtainable using processed labelling (from now on “*PP classifier*”), as shown in Figure 3.32, where the “yellow” misclassified border around objects is significantly reduced and mostly substituted with classification as “green” background material. Also comparing the scatter plot representing the groups after clustering (Figure 3.27) to the *PP classifier* scatter plot in Figure 3.34, the extension of the “green” area appears clearly. Nevertheless, some misclassification with respect to the *PP* dataset still occurs, as shown by the crosses marking the wrongly classified points in Figure 3.34. In order to have a clear view of the number of points that have erroneous labelling, with respect to the *PP* dataset labels, it is possible to build a *confusion matrix* (Figure 3.33), i.e. a matrix in which each element (i, j) is



Figure 3.32 – Results for classifier trained on pre-processed data (left) and on non-pre-processed (right) compared. The colormap indicated the labels corresponding to each color

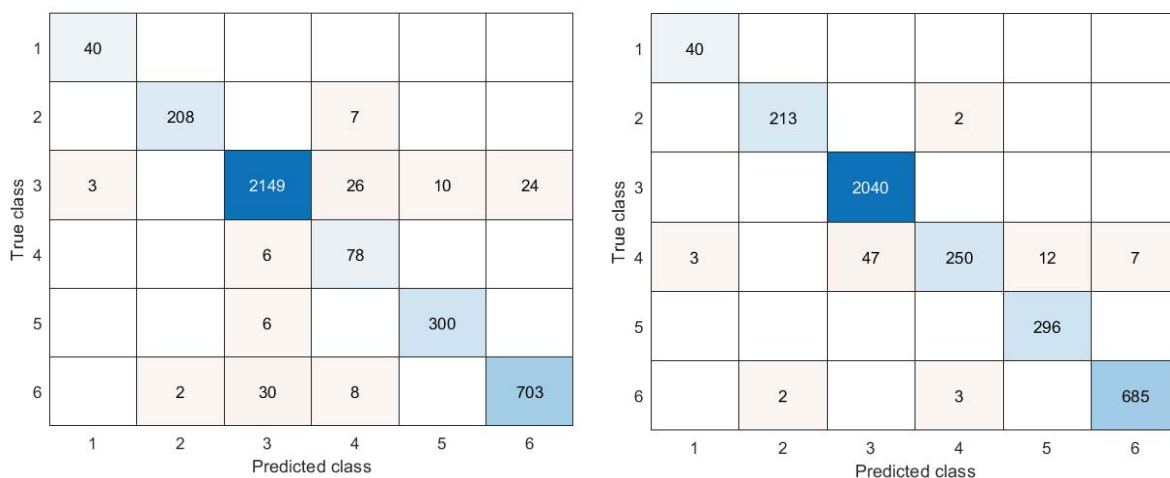


Figure 3.33 – Confusion matrix in the case of a pre-processed (left) or non-pre-processed (right) dataset

the number of samples whose true class is i while the predicted class is j . Therefore, rows and column indexes represent class indexes, and the diagonal elements, for which the true and the predicted class are equal, represents the correctly classified points. Comparing the confusion matrix for *NPP* and *PP classifier*, it may seem that the *NPP classifier* is somehow better because it commits less mistakes compared to the training dataset labels, but it is important to remember that the *NPP* dataset already contains mistakes with respect to what is the natural and true classification of the materials because of incorrect border labelling, therefore the classifier thus obtained is adherent to wrong labels and so it is inherently wrong anyway. In fact, when the *PP classifier* is applied to the *wt*-data, more points are assigned to a group different from the one in the *PP* labels matrix, and this happens because, as can be deduced observing Figure 3.31 (left), some of the holes in the objects have disappeared in the labels matrix during the morphological operations, but in the real data there is a significant difference in intensity between points that represent an object and points corresponding to holes, and as a consequence those “holes” are labelled by the classifier with group indexes different from the object ones, and fortunately they are labelled mostly as background, as in reality those holes show exactly the background. In conclusion, the best classifier is definitely the one trained on the dataset pre-processed with morphological operations.

Applying the *PP-classifier* to the PCA scores image, the results are quite satisfying (Figure 3.35). The shadowed borders, instead of being classified erroneously as “yellow” material, are mostly labelled as background or as the corresponding object. All the materials are correctly labelled, and the classifier can be used for future applications. In fact, in general, a misclassification to the background is better than a misclassification to another material

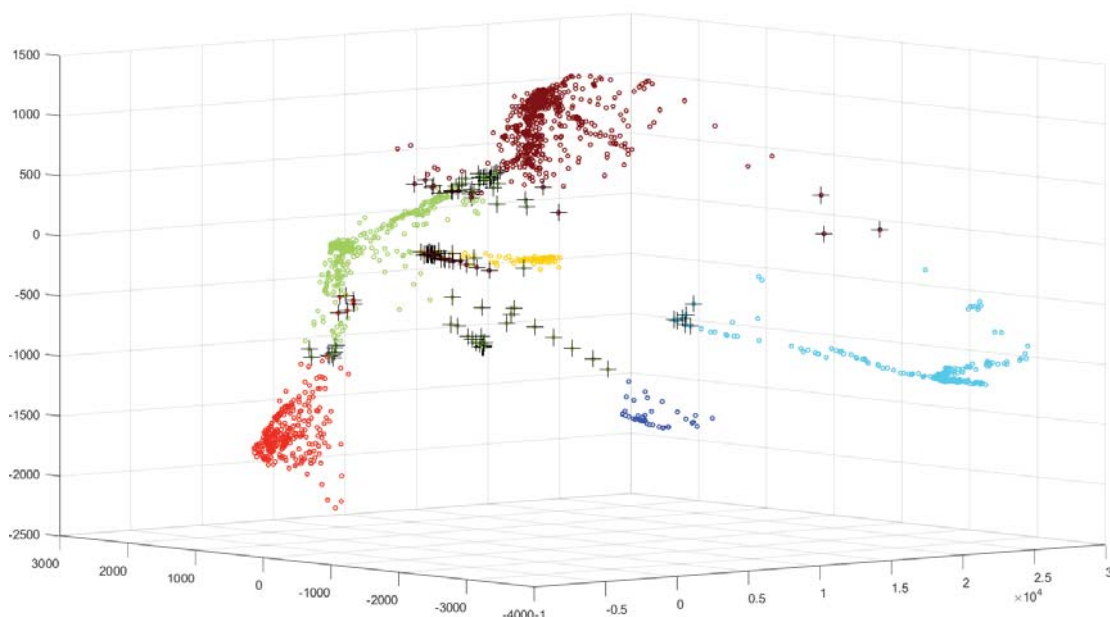


Figure 3.34 – Scatter plot showing the groupings obtained through the classifier. The misclassified datapoints are marked with a cross (+)

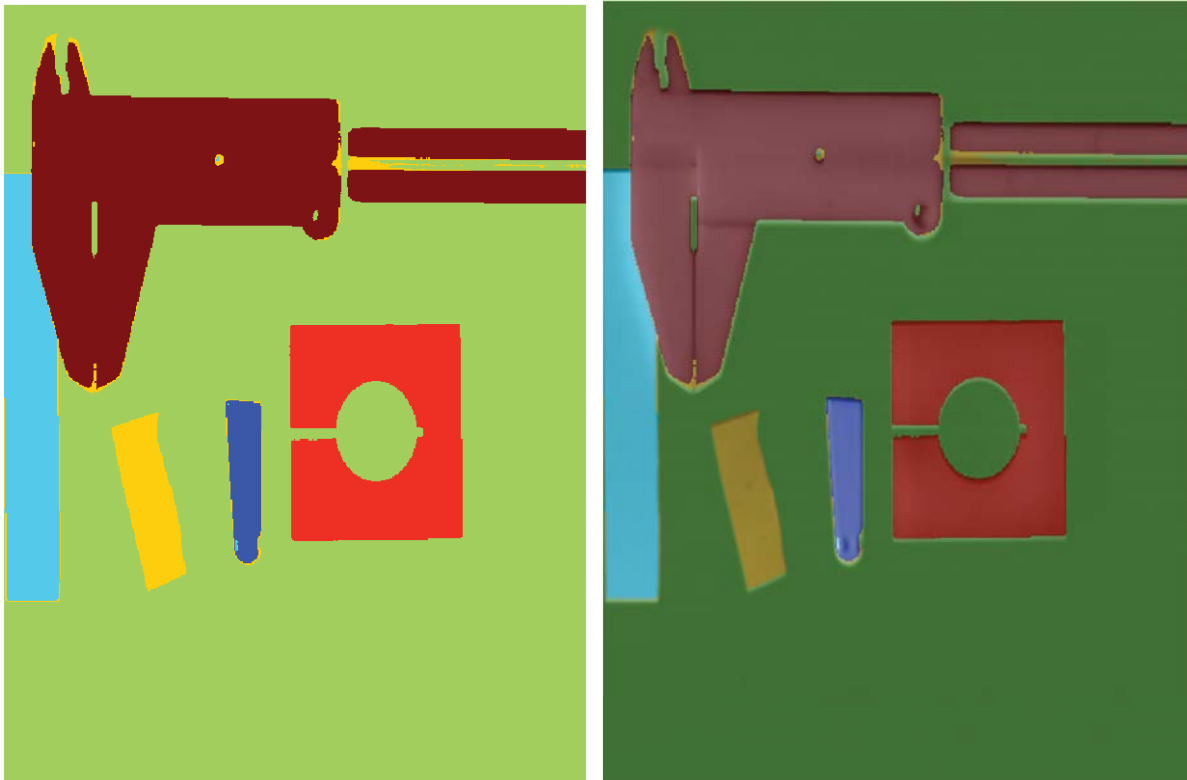


Figure 3.35 – Classification of the PCA scores image using the *PP-classifier* (left) and superimposition of the results on a greyscale version of the original image (right)

3.6 Problem dimension, time complexity and execution time

This paragraph is meant to make a point about how, sometimes, apparently intensive and long pre-processing before a specific task can, overall, be more advantageous than performing the same task on unprocessed data.

3.6.1 Introduction to the problem

In this specific work, the main objective was to label different materials in a hyperspectral image that collects intensity responses for all the materials of interest in the near-infrared spectrum range. If a classifier able to distinguish the different materials is also build, the side quest of classifying future images containing the same materials is also solved. The main challenge is undoubtedly the dimension of the dataset. Concerns from the classification point of view arise also due to the fact that the materials constitute objects of different sizes and shapes that are photographed all together with a non-vertical light source, creating significant shadowing. Nevertheless, shadowing and contemporaneous presence of several materials can

actually be a real scenario of future applications of the classifier itself, therefore a set of perfect pictures of different materials would be less adequate to build a classification model to be used in images of real objects.

Labelling different materials in an “unsupervised” way, i.e. leaving to computer calculations the grouping task given a method and a distance measure, can be a very expensive task from the computational point of view, as illustrated in section §2.5.2.2. The computational cost or time complexity is an estimation of the number of elementary operations performed by an algorithm, in terms of order of magnitude³¹. Some (very few) algorithms may require the same amount of operations independently on the size of the dataset on which they are performed, but some others may depend hugely on the number of elements in the dataset, and this is the case for clustering algorithms, that need to execute similarity calculations involving each object in the dataset. If the clustering task was to be performed on the original image using a hierarchical method the computational cost would be prohibitive, as shown in Table 3.3. Therefore, the use of other simplified methods would be necessary, with all the disadvantages related. In fact, as already mentioned in §2.5.2.2, partitional algorithms are faster than hierarchical ones, as shown also in Table 3.3, where it is assumed that 100 iterations are more than enough to be able to cluster the data into 6 groups through k-means method. In fact, as an order of magnitude, the total number of iterations l is usually less than the number of points n (Subbalakshmi *et al.*, 2014), and in general it is assumed to be relatively low to ensure the computational advantage over the hierarchical method. But lower execution time comes at a cost: the algorithm has to be run more than once because the final results depend on the position of the initial “seeds” (the random initial centroids), reproducibility is affected by the fact that each run gives different results, the only parameter is the distance between points (hierarchical clustering allows to choose also the parameter “distance between clusters”) and fewer options are compatible with the algorithm, etc. In general, hierarchical algorithm is more rigorous, and it is preferable when applicable, even if it is more expensive, since usually the quality of the results is better. To be able to use a hierarchical method, it is necessary to check if somehow it is possible to reduce the dimension of the dataset keeping just “significant” points, i.e. points retaining meaningful information to build a classifier, that could be used for this case and for future applications. Within this frame, dimension reduction through principal component analysis and subsequent application of the wavelet transform, could give a cut to the size of the problem. If the cost of these pre-processing operations is significantly lower than the cost of applying the clustering algorithm to the original image, the “game” is actually “worth the candle”. Table 3.3 shows how

³¹ For an introduction to computational complexity and some basic examples refer to the online material made available by the University of Wisconsin <http://pages.cs.wisc.edu/~vernon/cs367/notes/3.COMPLEXITY.html> (last access: 07/2019)

computational cost reduction is significant with the decreasing of the dataset size, being the dependence higher than parabolic.

		224-bands hyperspectral image	PCA scores image	Wavelet transform result (3 decomposition levels)
Number of observations		$480 \cdot 640$ $= 3.07 \cdot 10^5$	$480 \cdot 640$ $= 3.07 \cdot 10^5$	$60 \cdot 60$ $= 3.6 \cdot 10^3$
Matrix size		$480 \cdot 640 \cdot 224 =$ $6.88 \cdot 10^7$	$480 \cdot 640 \cdot 3 =$ $9.2 \cdot 10^5$	$60 \cdot 60 \cdot 3 =$ $1.08 \cdot 10^4$
Hierarchical Algorithm	Computational Cost $\mathcal{O}(n^2 \log n)$	$1.16 \cdot 10^{14}$	$1.55 \cdot 10^{12}$	$2.13 \cdot 10^8$
	Execution Time	$4.64 \cdot 10^4$	$6.21 \cdot 10^2$	$8.53 \cdot 10^{-2}$
Partitional (K- Means) Algorithm	Computational Cost $\mathcal{O}(nkl)$ (With $K=6, L=100$)	$4.13 \cdot 10^{10}$	$5.53 \cdot 10^8$	$6.48 \cdot 10^6$
	Execution Time	$1.65 \cdot 10^1$	$2.21 \cdot 10^{-1}$	$2.59 \cdot 10^{-3}$

Table 3.3 – Calculations and estimations of problem dimension, computational cost and execution time (based on equation (3.6.1-2) with a clock frequency of 2.5 GHz)

Execution time. Besides the computational cost, Table 3.3 includes an estimation of a parameter indicated as “execution time”. In computer science, the execution time required by a program is computed as follows

$$\frac{\text{time}}{\text{program}} = \frac{\text{number of clock cycles}}{\text{program}} \cdot \frac{1}{\text{clock frequency}} \quad (3.6.1-1)$$

where the clock frequency is a measure of the rate at which the CPU executes the instructions of a computer program. The number of clock cycles required by a program is the number of elementary steps needed to perform a task. These elementary steps must not be confused with the “elementary operations” considered in the calculation of the computational cost. For example, a single multiplication is an “elementary operation” but may require from the CPU several clock cycles to be executed. Nevertheless, by construction, the minimum possible is that each task (or elementary operation) requires exactly one clock cycle. Under this simplifying assumption, an estimate of the time required to execute an algorithm can be

obtained, keeping in mind that each elementary operation can require a different number of clock cycles and that the result obtained can be simply multiplied by the average value of the number of cycles to obtain a more accurate estimation of the time required. From equation (3.6.1-1) multiplying both sides for the number of tasks per clock cycle:

$$\begin{aligned} \frac{\text{time}}{\text{program}} \cdot \frac{\text{tasks}}{\text{nr of cycles}} \\ = \frac{\text{tasks}}{\text{nr of cycles}} \cdot \frac{\text{nr of cycles}}{\text{program}} \cdot \frac{1}{\text{clock frequency}} \end{aligned}$$

Substituting $\text{tasks}/(\text{nr of cycles}) = 1$, and simplifying on the right side

$$\frac{\text{time}}{\text{program}} \cdot 1 = \frac{\text{tasks}}{\text{nr of cycles}} \cdot \frac{\text{nr of cycles}}{\text{program}} \cdot \frac{1}{\text{clock frequency}}$$

We obtain

$$\frac{\text{time}}{\text{program}} = \frac{\text{tasks}}{\text{program}} \cdot \frac{1}{\text{clock frequency}} \quad (3.6.1-2)$$

Considering that the number of tasks is by definition the computational cost, knowing the clock frequency of the CPU on which the calculations will be executed, an estimation of the execution time required can be easily calculated. The clock frequency of the machine used for this work is 2.50 GHz³², and results in Table 3.3 assume to run the CPU at maximum speed performing only the calculations required by the algorithm. In reality, the cycles required by a single task can be several, and the CPU is running several programs at the same time (e.g. the MATLAB[®] user interface itself), therefore the time required can be one order of magnitude higher if not more. The actual time required by the machine in the specific situation of several programs running can be measured through the `tic toc` function³³ in MATLAB[®], and depends on the current number of programs the user is executing in the background (e.g. music player running!).

3.6.2 Overall computational cost

To check the feasibility of the approach proposed, involving heavy preprocessing before clustering operations, it is important to consider the computational cost of each algorithm involved. For the principal component analysis, the two main tasks involved (Povey *et al.*,

³² AMD A12-9700P RADEON R7, 10 COMPUTE CORES 4D+6G 2.50 GHz

³³ For more information refer to https://it.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html (last access: 07/2019)

2010) are the computation of the covariance matrix, that requires $\mathcal{O}(\lambda^2 n)$ elementary calculations (where λ is the number of variables and n , as usual, the number of points) and the eigen-vector decomposition, that is $\mathcal{O}(\lambda^3)$. The computational cost of a wavelet transform may be estimated as $\mathcal{O}(4N^2 \log N)$ (Bhaskaran *et al.*, 1995), where $N \cdot N$ is the dimension of the square matrix than undergoes the transformation. Finally, as seen, hierarchical clustering requires $\mathcal{O}(n^2 \log n)$ operations. As illustrated in Table 3.4, the overall cost is dominated by the number of steps required by the PCA, but also clustering gives a significant contribution. Up to this point, with or without pre-processing, the result is a labelled data set that can be used to train a linear classifier, but the overall cost differs of six order of magnitudes, as displayed in Table 3.4 under the voice “subtotal cost”.

If the only objective was to classify the current image, hierarchical clustering already gives the wanted information if applied directly on the original hyperspectral data. Conversely, if hierarchical clustering is applied on the low-resolution image, an additional cost of building a classifier has to be considered. According to Cai *et al.* (2008), if linear discriminant analysis is performed on a dataset counting a number of samples (or “points”) greater than the number of features (or “variables”), the number of operations needed to build the classifier is $\mathcal{O}(\lambda^2 n + \lambda^3)$, where λ is the number of features and n the number of samples. Once the classifier is trained, the cost of application is negligible since the hyperplanes dividing the groups have already been defined. As shown in Table 3.4, the final cost having a linear classifier does not change perceivably for the procedure with pre-processing, since building a classifier is relatively inexpensive for a small dataset, while if the number of points and of features increase significantly, the cost will skyrocket. Notice how building a classifier using the original data as a training set is as expensive as performing PCA on the original image. Nevertheless, if a classifier is needed for future applications, the final cost of clustering applied on the original image surpasses anyway the cost of LDA, and makes heavily inconvenient following this path. In conclusion, the appeal of a classifier built on a pre-processed low-resolution dataset is undeniable. The last entry in Table 3.4 gives an estimation of the execution time according to the formulation proposed in equation (3.6.1-2) and considering a clock frequency of 2.5 GHz, that is the one of the CPU used for this work. In reality, as mentioned in the previous section, not the full clock frequency is available for the calculations, since other necessary tasks involving the execution of the operative system and of MATLAB[®] itself are also utilizing a portion of the CPU calculation power. Moreover, the actual cost of an elementary operation can be significantly higher than just one clock cycle. These and other related issues make the results of equation (3.6.1-2) just an estimation of the approximate order of magnitude of the actual execution time. In fact, the few seconds estimated for the execution of the whole sequence of procedures to build the “low-resolution” classifier translates into few minutes of real execution time on the commercial laptop used for this work. The actual time required to execute the procedure on the original dataset can only

be left to imagination because of RAM limitations addressed in the next section (§3.6), where computations carried out at several resolutions are compared.

<i>Method</i>	<i>Computational cost formula</i>	<i>Result with pre-processing steps</i>	<i>Results without pre-processing</i>
<i>PCA</i>	$\mathcal{O}(\lambda^2 n + \lambda^3)$	$1.54 \cdot 10^{10}$	n.a.
<i>Wavelet transform</i>	$\mathcal{O}(4N^2 \log N)$	$8.07 \cdot 10^6$	n.a.
<i>Hierarchical clustering</i>	$\mathcal{O}(n^2 \log n)$	$2.31 \cdot 10^8$	$1.16 \cdot 10^{14}$
<i>Subtotal cost</i> <i>(without linear classifier)</i>		$1.56 \cdot 10^{10}$	$1.16 \cdot 10^{14}$
<i>LDA</i>	$\mathcal{O}(\lambda^2 n + \lambda^3)$	$3,24 \cdot 10^4$	$1.54 \cdot 10^{10}$
<i>Total cost</i> <i>(with linear classifier)</i>		$1.56 \cdot 10^{10}$	$1.16 \cdot 10^{14}$
<i>Estimated execution time</i>		<i>6,36 seconds</i>	<i>$4.64 \cdot 10^4$ seconds</i> <i>(more than 12 hours)</i>

Table 3.4 – Overall computational cost with and without pre-processing (n.a. = non-applicable). The execution time is estimated as in equation (3.6.1-2) with a clock frequency of 2.5 GHz

3.7 Refining the model

As mentioned in the previous Chapter, after a working base case is defined, it is time to explore the consequence of varying the spatial and spectral resolution.

3.7.1 Spectral resolution

As seen in Figure 3.9 and on Table 3.1, the value of the cumulative explained variation becomes, more or less, steady after the first three principal components are considered for the new basis. This means that retaining less than three components could cause difficulty in distinguishing clusters, but retaining more than three could be a useless additional computational burden from the point of view of clustering and classification. Figure 3.36 collects results for both scenarios. If the dataset is projected only on one or two principal components, the separation between clusters is insufficient, and a situation similar to the one presented in Figure 2.16 (c) arises. Observing clustering results in the case of one PC retained, even forcing the algorithm to find six clusters, it is clear that the natural grouping of the scores on PC1 is such that objects (2) and (4) are considered made of the same “orange” material and objects (3) and (5) are labelled as “cyan” material. Similarly, if two principal components are considered, it is hard to distinguish the materials of objects (1) and (4).

As presented in the previous paragraphs, retaining three components allows finally for a good separation among the six materials. Furthermore, more components do not improve the

classification results, as shown in Figure 3.36. Both cases of four and five principal

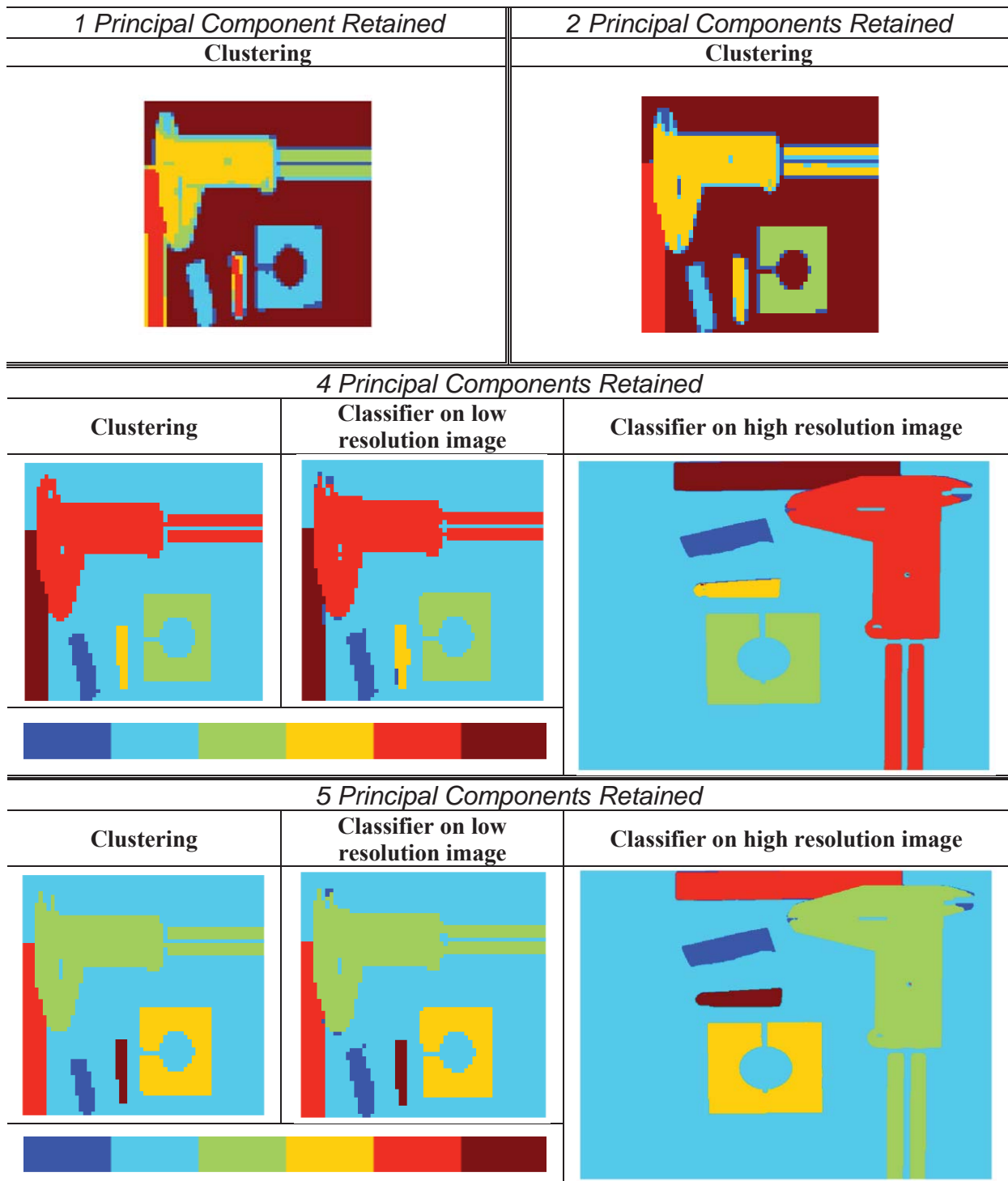


Figure 3.36 – Unsupervised and supervised classification results in the case of one, two, four or five principal components retained (wavelet transform applied up to the third level of decomposition)

components classify the points with an accuracy comparable to the results obtained with three components, since the shadowed areas of object (1) are still classified as background, so adding these components will just increase the computational effort adding little to no

information, as already discussed considering Table 3.1. Polishing clustering results, misclassification of small groups of points at the border of object (1) can be corrected, as seen for the case of three PC in paragraph §3.5.1.

3.7.2 Spatial resolution

When the wavelet transform is applied for training the classifier, spatial resolution varies in such a way that, at each step, both length and width of the image are halved. Therefore, for a square image of side equal to 480 pixels there are very few levels of decomposition available before the image content is completely degraded, as demonstrated in Figure 3.37. Remembering that level 0 is the original image, after three level of decomposition the resolution (60×60) is still high enough that the smallest objects can be distinguished and properly labelled. When the size is halved, the smallest object is represented by less than ten points, therefore its proper classification is almost impossible, considering also how similar its spectrum is to the response of other materials. As a matter of fact, as shown in Figure 3.36, if not enough components are considered, that object's material is erroneously classified as coincident with the material of another object. After five levels of decomposition, small objects cannot be distinguished anymore, and forcing the creation of six clusters has the only consequence of obtaining misclassified borders for the bigger objects.

On the contrary, considering a higher resolution image obviously could allow to distinguish the objects considered even more easily, from the point of view of unsupervised learning, and could give to the classifier a bigger dataset on which to train. Nevertheless, if the performance of the trained classifier is not improved there is no reason to request the increased computational effort obviously related to the processing of a bigger image (remember that the computationally most expensive operation is the preliminary clustering; therefore, the spatial resolution should be tuned carefully, as it rules the overall computational burden of the proposed segmentation procedure). Figure 3.37 illustrates all the step up to the final labelling of the high-resolution image for the case of decomposition up to the second level before clustering. It shows how the results are perfectly equivalent to the ones produced using the third level of decomposition, since anyway some small regions at the borders get misclassified and again some shadowed areas are labelled as background, therefore higher cost in this case comes with no tangible advantage.

For the first level of decomposition the original image is only halved in size. Therefore, the clustered dataset has a spatial resolution of 240×240 and considering that each of these 240×240 points has three coordinates (if three principal components are retained), evaluating the distance for *all* the pairs of points would be extremely expensive. For how it is built, MATLAB[®] stores this matrix into the RAM memory up to when the distances calculations

are completed. Nominally, the RAM of the personal computer used for this study is 12 GB, of which 0.6 GB are necessary for running the Operative System (Windows® 10 – 64 bit). If the pairwise distance matrix calculation is started, MATLAB® automatically stops the execution of the program informing the user that the requested RAM (12.4 GB) exceeds the RAM

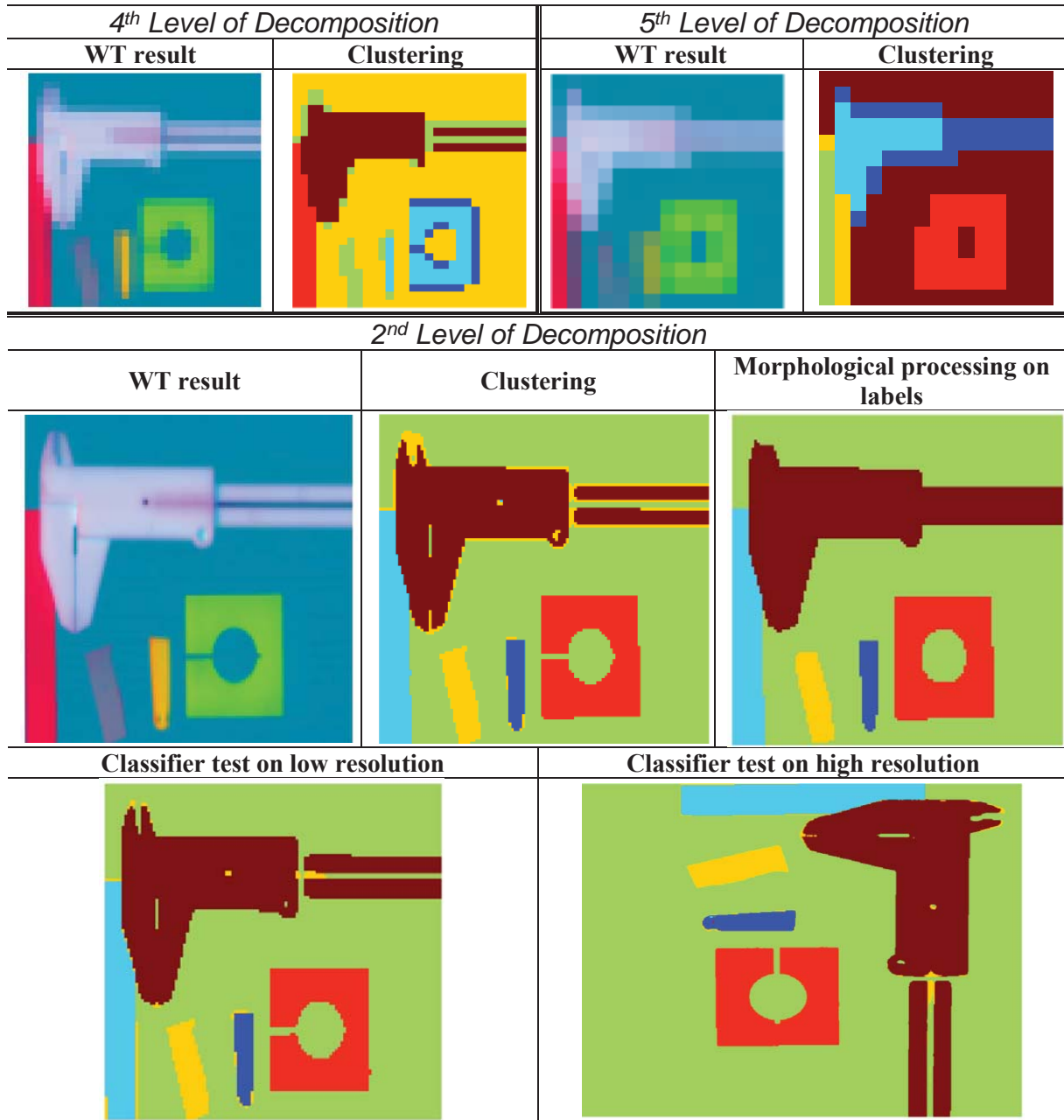


Figure 3.37 – Unsupervised and supervised classification results in the case of two, four and five level of decomposition (three principal components retained)

available for calculations³⁴. Very big dataset cannot be processed in the usual way. MATLAB[®] online documentation provides some tools³⁵ implemented specifically for dealing with big data. In this situation, since the second or third level of decomposition already produce satisfactory results, there is no need to further explore these complex scenarios involving big data processing. Moreover, applying these techniques is against the objective of this work of finding a simple and computationally affordable way to label a big dataset with a classifier trained on a smaller dataset.

Finally, notice that, as shown in Table 3.5, if the number of principal components retained is sufficiently low, the main computational burden of the whole procedure remains the PCA itself, as was for the base case, since the final total cost remains around $1.6 \cdot 10^{10}$. In practice, also the real execution time is very similar for all the cases, and stays of the order of few minutes, while the one predicted through equation (3.6.1-2) is of few seconds because of the approximation considering one clock cycle sufficient to perform an elementary operation (see paragraph §3.6). The cost of the PCA obviously does not depend on the number of components considered because when the method is applied the full 224-components decomposition is evaluated.

If instead the number of components retained is fixed to three, varying the decomposition depth has a significant effect on clustering because the number of points for which pairwise distance has to be calculated is divided by 2^d for the decomposition level d , while varying the number of PC retained only increases the number of elements of the image matrix by n additional points for each PC. As a consequence, the variation of the computational cost with the spatial resolution is significant (Table 3.6). Remembering again that equation (3.6.1-2) introduces the assumption of one clock cycle for each elementary operation, comparing measured and theoretical execution time for the sole clustering operation shows how this assumption can barely give an approximated order of magnitude for the execution time, while the time actually required will always be greater. Anyway, for lower resolution images the computational cost of clustering remains lower than a second. Starting from the second level of decomposition, the time spent for the execution increases steeply, not only because

³⁴ Notice that the RAM actually available for calculation is not the 11 GB of memory remaining after the OP is started. For more details, it is possible to ask the execution of the MATLAB script `memory` in the Command Window. For this specific case:

```
>> memory
Maximum possible array:      9175 MB (9.620e+09 bytes) *
Memory available for all arrays: 9175 MB (9.620e+09 bytes) *
Memory used by MATLAB:      10932 MB (1.146e+10 bytes)
Physical Memory (RAM):      11664 MB (1.223e+10 bytes)

* Limited by System Memory (physical + swap file) available.
```

³⁵ See <https://it.mathworks.com/help/matlab/large-files-and-big-data.html> (last access: 07/2019)

elementary operations could require way more than one clock cycle, but likely also because the RAM was almost filled by the distance matrix and the calculations were significantly slowed down by the fact that also other programs had to access the remaining RAM. For the first level of decomposition, requiring an amount of RAM superior to the physical memory installed on the personal computer used, of course no information is available about the actual execution time of the algorithm. This shows how a fast CPU is not enough if the RAM is completely filled.

In conclusion, from both the point of views of quality of the results and affordability of the computational cost, the base case with three principal components retained and resolution lowered up to the third level of decomposition seems to be the optimum.

<i>Method</i>	<i>Computational Cost – 3rd level of decomposition for the WT</i>				
	<i>1 PC retained</i>	<i>2 PC retained</i>	<i>3 PC retained</i>	<i>4 PC retained</i>	<i>5 PC retained</i>
<i>PCA</i>	$1.54 \cdot 10^{10}$				
<i>Wavelet T.</i>	$2.47 \cdot 10^6$	$5.22 \cdot 10^6$	$8.07 \cdot 10^6$	$1.10 \cdot 10^7$	$1.40 \cdot 10^7$
<i>Clustering</i>	$4.61 \cdot 10^7$	$9.22 \cdot 10^7$	$1.38 \cdot 10^8$	$1.84 \cdot 10^8$	$2.30 \cdot 10^8$
<i>LDA</i>	$3.60 \cdot 10^3$	$1.44 \cdot 10^4$	$3.24 \cdot 10^4$	$5.77 \cdot 10^4$	$9.01 \cdot 10^4$
<i>Total cost</i>	$1.55 \cdot 10^{10}$	$1.55 \cdot 10^{10}$	$1.56 \cdot 10^{10}$	$1.56 \cdot 10^{10}$	$1.57 \cdot 10^{10}$
<i>Execution time</i>	6.2 seconds				

Table 3.5 – Computational cost for several values of the spectral resolution, i.e. for different number of PC retained

<i>Method</i>	<i>Computational Cost – 3 Principal Components retained</i>				
	<i>Dec. level: 1</i>	<i>Dec. level: 2</i>	<i>Dec. level: 3</i>	<i>Dec. level: 4</i>	<i>Dec. level: 5</i>
<i>PCA</i>	$1.54 \cdot 10^{10}$				
<i>Wavelet T.</i>	$8.07 \cdot 10^6$				
<i>Clustering</i>	$1.58 \cdot 10^{10}$	$8.62 \cdot 10^8$	$4.61 \cdot 10^7$	$2.39 \cdot 10^6$	$1.19 \cdot 10^5$
<i>LDA</i>	$5.18 \cdot 10^5$	$1.30 \cdot 10^5$	$3.24 \cdot 10^4$	$8.13 \cdot 10^3$	$2.05 \cdot 10^3$
<i>Total cost</i>	$3.12 \cdot 10^{10}$	$1.65 \cdot 10^{10}$	$1.55 \cdot 10^{10}$	$1.54 \cdot 10^{10}$	$1.54 \cdot 10^{10}$
<i>Execution time</i>	68.7 seconds	9.6 seconds	6.4 seconds	6.2 seconds	6.2 seconds
<i>Ex. time for clustering</i>	6.32 seconds	0.345 seconds	0.018 seconds	0.001 seconds	$1 \cdot 10^{-4}$ seconds
<i>Measured ex. time for clustering</i>	n.a.	28 minutes	0.89 seconds	0.87 seconds	0.24 seconds

Table 3.6 – Computational cost for several values of the spatial resolution, i.e. for different levels of decomposition for the wavelet transform

Conclusions

In this work, a technique for segmentation of hyperspectral images was proposed. Since, for its very nature, hyperspectral data tends to be huge in size, operations with a more-than-linear dependence on the size of the dataset can end up requiring an unbearable computational effort. Unsupervised classification of an image for its segmentation is one of the most expensive basic operations in image processing, hence the necessity of techniques to avoid the direct application of clustering on a huge dataset. The approach proposed consists in the application of unsupervised classification to a low-resolution version of the original image, training then a classifier to label the high-resolution image. Application of the method to a typical segmentation problem is carried out to illustrate quantitatively the performances of the method proposed. Size reduction in terms of spectral dimension is achieved through application of principal component analysis, that projects the *observations* (i.e. the points of the image) onto a new set of basis vectors defined in such a way that variability of the projected data is maximized. For the case under examination, retaining only the projections (or *scores*) capturing the maximum amount of variability, it is possible to preserve more than 99% of the total variability of the data. Spatial resolution of the image is lowered applying the Haar wavelet transform, that at each step halves the size of the image, performing operations of downsampling and averaging between adjacent points. The resolution is lowered up to a level of coarseness retaining sufficient detail for objects discrimination. After, hierarchical clustering operation is finally performed on a relatively small dataset, calculating Euclidean distance between points and using Ward's formula for the inter-group distances. The best distance definition depends on the nature of the dataset, but in general Euclidean distance is appropriate for image clustering. The labelled data thus obtained is then used to train a linear classifier, defining hyperplanes separating the clusters from one another. The final step of the method is then to apply the classifier to the high-resolution image, that in this way is labelled without directly using costly clustering algorithms. If the resolution of the reduced dataset is not too coarse, the final results on the high-resolution image are more than satisfying, and the computational effort decreases of several orders of magnitude (four o.o.m. for the specific case under examination). After a base case is solved, calculations for different values of the image resolutions are carried out, to try to achieve the better compromise between quality of the results and computational effort. Since a low-resolution image is obviously fast to be processed but inaccurate in the results, a trade-off can be identified varying the number of principal components retained and the level of coarseness of the low-resolution image obtained after the wavelet transform. Finally, it is important to notice that, if

MATLAB® and its built-in toolboxes are employed to perform the clustering operation, the computational cost for the CPU is not the only limitation to the processing of large datasets, since real issues arise if large data matrices have to be computed and stored in the RAM, that can even get completely filled.

The quality of the results obtained for the sample dataset makes this procedure quite promising not only for situations when a faster response is needed, but also when computing power is limited. In the future, the limits of this procedure can be explored by testing it on more complex datasets, for example objects with complicated textures, like vegetation.

Bibliography

Questa bibliografia è dedicata a Umberto Eco e alla sua Appendice a “Come Presentare un Catalogo d’Arte”, dalla raccolta “Come Viaggiare con un Salmone”

This bibliography is dedicated to Umberto Eco and to his Appendix to the essay “How to Write an Introduction”, from the book “How to Travel with a Salmon and Other Essays”

Backer S.D., Kempeneers P., Debruyne W., (2005) “A band selection technique for spectral classification”, IEEE Geoscience Remote Sensing Letters, vol. 2, no. 3

Barry M. Wise, Neal B. Gallagher (1996). The process chemometrics approach to process monitoring and fault detection, Journal of Process Control, Volume 6, Issue 6

Bellman R., (1961). Adaptive Control Processes: A Guided Tour, Princeton University Press.

Bhaskaran V., Konstantinides K. (1995). Image and video compression standards: algorithms and architectures. Kluwer, Boston.

Bro, R., Smilde, A. K. (2003), Centering and scaling in component analysis. J. Chemometrics, 17: 16-33. doi:10.1002/cem.773

Bro, R., Smilde, A. K. (2014). Principal component analysis. Analytical Methods, 6(9), 2812-2831. <https://doi.org/10.1039/c3ay41907j>

Byrnes, James (2009). Unexploded Ordnance Detection and Mitigation. Springer. pp. 21–22

Cai, D., He, X., Han, J. (2008) "Training Linear Discriminant Analysis in Linear Time" IEEE 24th International Conference on Data Engineering, Cancun, 2008, pp. 209-217.

Cattell, R. B. (1966). The Scree Test for the Number of Factors. Multivariate Behavioral Research, 1, 245-276

Ceamanos, X., Valero, S. (2016). Chapter 4 - Processing Hyperspectral Images, Pages 163-200, Optical Remote Sensing of Land Surface, Editor(s): Nicolas Baghdadi, Mehrez Zribi, Elsevier

Damez, J.L., Clerjon, S. (2008). Meat quality assessment using biophysical methods related to meat structure. Meat Science. 80(1): 132–149.

Elmasry, G., Barbin, D. F., Sun, D. W., Allen, P. (2012). Meat Quality Evaluation by Hyperspectral Imaging Technique: An Overview, Critical Reviews in Food Science and Nutrition, 52:8, 689-711, DOI: 10.1080/10408398.2010.507908

Gao B.C., Montes M.J., Davis C.O., (2009) “Atmospheric correction algorithms for hyperspectral remote sensing data of land and ocean”, Remote Sensing of Environment, vol. 113, no. 1, pp. S17–S24

- Gonzalez, R.C., Richard, E.W., Steven, L.E., (2010). Digital Image Processing Using MATLAB®, Second Edition, Mc Graw Hill Education
- Gowen, A.A., O'Donnell, C.P., Cullen, P.J., Bell, S.E.J., (2008). Recent applications of Chemical Imaging to pharmaceutical process monitoring and quality control, European Journal of Pharmaceutics and Biopharmaceutics, Volume 69, Issue 1
- Hastie, T., Tibshirani, R., Friedman, J. (2001). The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc..
- Hefferon, Jim (2017). Linear Algebra. <http://joshua.smcvt.edu/linearalgebra/>
- Iqbal, M., (1983). "An Introduction to Solar Radiation", Academic Press, Chapter 3
- Jackson, J.E. (1991). A User's Guide to Principal Components. New York: Wiley
- Jain, A. K., Murty, M. N., Flynn, P. J, (1999). Data clustering: a review. ACM Comput. Surv. 31, 3 (September 1999), 264-323. DOI=<http://dx.doi.org/10.1145/331499.331504>
- Jain, R., Kasturi, R., Schunck, B. G. (1995). Machine Vision. McGraw-Hill series in computer science. McGraw-Hill, Inc., New York, NY.
- Kaushik, M., Mathur, B. (2014). Comparative Study of K-Means and Hierarchical Clustering Techniques. International journal of Software and Hardware Research in Engineering. 2. 93-98.
- Kurita, T. (1991). An efficient agglomerative clustering algorithm using a heap. Pattern Recogn. 24, 3 (1991), 205–209.
- Land, E. H., McCann, J., (1971). Lightness and Retinex Theory. Journal of the Optical Society of America. 61. 1-11. 10.1364/JOSA.61.000001.
- Mallat, S. G. (1989) A theory for multiresolution signal decomposition: the wavelet representation, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 674-693, July 1989.
- Megahed, A. I., Monem Moussa, A., Elrefaie, H. B., Marghany, Y. M. (2008). Selection of a suitable mother wavelet for analyzing power system fault transients. 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century.
- Mudrová, M., Procházka, A. (2005). Principal Component Analysis in image processing, In: Proceedings of MATLAB® Technical Computing Conference, Prague
- Ngui, W. K., Leong, M. S., Hee, L. M., Abdelrhman, A. M. (2013). Wavelet Analysis: Mother Wavelet Selection Methods. Applied Mechanics and Materials, 393, 953–958.
- Povey D. and Rastrow, A. (2010). "Computational Issues In Principal Components Analysis", ICASSP
- Rasmussen, E. (1992). Clustering algorithms. In Information Retrieval: Data Structures and Algorithms, W. B. Frakes and R. Baeza-Yates, Eds. Prentice-Hall, Inc., Upper Saddle River, NJ, 419–442.

- Reis, M. S., Saraiva, P. M., Bakshi, B. R., (2008), Multiscale statistical process control using wavelet packets. *AIChE J.*, 54: 2366-2378. doi:10.1002/aic.11523
- Reis, M., Bakshi, B., Saraiva, P.M., (2010). Denoising and Signal to Noise Enhancement: Wavelet Transform and Fourier Transform. *Comprehensive Chemometrics: Chemical and Biochemical Data Analysis*. 2. 25-55. 10.1016/B978-044452701-1.00099-
- Reis, M., Bauer, A.. (2009). Wavelet texture analysis of on-line acquired images for paper formation assessment and monitoring. *Chemometrics and Intelligent Laboratory Systems*. 95. 129-137. 10.1016/j.chemolab.2008.09.007
- Subbalakshmi, C., Rao, P. V., Rao, S. K. M. (2014) "Performance Issues on K-Mean Partitioning Clustering Algorithm," *Int. Journal of Computer (IJC)*, vol. 14, no. 1, pp. 41-51
- Tan, Y. (2016) *Gpu-Based Parallel Implementation of Swarm Intelligence Algorithms*, Chapter 11, Editor(s): Ying Tan, Morgan Kaufmann
- Torrence, C., Compo, G.P., (1998) *A Practical Guide to Wavelet Analysis*. *Bull. Amer. Meteor. Soc.*, 79, 61–78

Web sites

Chapter 1

https://www.nasa.gov/directorates/heo/scan/spectrum/txt_electromagnetic_spectrum.html (last access: 07/2019)

Chapter 2

<http://www.specim.fi/products/specim-fx17/> (last access: 07/2019)

<http://www.specim.fi/fx17-state-of-the-art-in-industrial-hyperspectral-imaging/> (last access: 07/2019)

https://en.wikipedia.org/wiki/Iris_flower_data_set
(last access: 07/2019)

<https://it.mathworks.com/help/stats/kmeans.html> (last access: 07/2019)

<https://it.mathworks.com/help/stats/pdist.html> (last access: 07/2019)

Chapter 3

<https://statweb.stanford.edu/~wavelab/> (last access: 07/2019)

<http://pages.cs.wisc.edu/~vernon/cs367/notes/3.COMPLEXITY.html>
(last access: 07/2019)

https://it.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html (last access: 07/2019)

<https://it.mathworks.com/help/matlab/large-files-and-big-data.html> (last access: 07/2019)

<https://it.mathworks.com/help/wavelet/gs/choose-a-wavelet.html>
(last access: 07/2019)

<https://statweb.stanford.edu/~wavelab/> (last access: 07/2019)

Symbols

Symbol convention used throughout the work, unless differently stated

\mathbf{v} vector (minuscule, cursive, bold)

\mathbf{M} matrix (majuscule, cursive, bold)

`function` MATLAB® function

List of tables and figures

Table 2.1 – Scanner measurement report	22
Table 3.1 – Fraction of total variation explained and cumulative fraction for the first 7 components.....	64
Table 3.2 – Correspondence between colours and principal components on which the scores are higher.....	71
Table 3.3 – Calculations and estimations of problem dimension, computational cost and execution time (based on equation (3.6.1-2) with a clock frequency of 2.5 GHz)	94
Table 3.4 – Overall computational cost with and without pre-processing (n.a. = non-applicable). The execution time is estimated as in equation (3.6.1-2) with a clock frequency of 2.5 GHz	97
Table 3.5 – Computational cost for several values of the spectral resolution, i.e. for different number of PC retained.....	102
Table 3.6 – Computational cost for several values of the spatial resolution, i.e. for different levels of decomposition for the wavelet transform	102
Figure 1.1 – Electromagnetic spectrum. Schematic representation by NASA (black and white conversion of the original colour image)	15
Figure 1.2 – Schematic illustration of the method proposed (left) as an alternative to direct clustering application (right).	19
Figure 2.1 – Electromagnetic spectrum from UV rays to NIR radiation	21
Figure 2.2 – Typical experimental assembly: Specim Lab Scanner 40x20 and Specim Hyperspectral Camera FX 17 (from the SpecimSpectral YouTube Channel).....	22
Figure 2.3 – Greyscale picture of the objects to be analysed in png format	23
Figure 2.4 – Greyscale conversion of the 150 th wavelength range acquired.....	23
Figure 2.5 – Unfolding hyperspectral image into 2D matrix.	24
Figure 2.6 – Example of dataset for which variables are not completely independent.....	25
Figure 2.7 – (left) Approximation \mathbf{X} of the original dataset \mathbf{X} along the first principal component; (right) Second principal component and example of projection of a point.....	27
Figure 2.8 – Grayscale conversion of the picture corresponding to the 150 th wavelength band. Brightness increased from 50% (reference value for the default brightness) to 80%.....	29
Figure 2.9 – Example of Haar wavelet: (left) mother wavelet; (right) wavelet obtained from the mother wavelet imposing $\mathbf{j} = \mathbf{1}$ in equation (2.4.2-2)	31

Figure 2.10 – Haar scaling function (or “father wavelet”).....	32
Figure 2.11 – Schematic representation of the wavelet expansion	33
Figure 2.12 – Bidimensional Haar scaling function and wavelets.....	35
Figure 2.13 – Decomposition resulting after wavelet transform.....	36
Figure 2.14 – Example of dendrogram.....	41
Figure 2.15 – Example of a multivariate gaussian distribution (+) with some disturbance points (*) that alter the sample average and the covariance matrix.	50
Figure 2.16 – Example of reduction of the spatial dimension – (a) and (b) – and of the spectral dimension – (c) and (d). Objects naturally belonging to different groups are represented with markers of different shapes.	54
Figure 3.1 – Spectral response of different points in the picture. Continuous line for the point of coordinates (row,col)=(100,300), dashed line for the point (300,300).....	57
Figure 3.2 – Identification numbers for the objects in the picture	57
Figure 3.3 – Greyscale conversion of the response for bands number 1 (left), 150 (middle), 224 (right).....	59
Figure 3.4 – Intensity of the response of row 90 (up) and column 155 (down) for the 150 th band.....	59
Figure 3.5 – Base of the Specim Lab Scanner 40x20 (from the SpecimSpectral YouTube Channel)	60
Figure 3.6 – Eigenvalues corresponding to each principal component (first 15 components) 63	
Figure 3.7 – Eigenvalues corresponding to each principal component (zoom on components 2-15).....	63
Figure 3.8 – Fraction of variation explained by each principal component (first 15 components).....	64
Figure 3.9 – Cumulative explained variation by the number of principal components retained (from 1 to 15 retained)	65
Figure 3.10 – Mean of the absolute error as function of the number of principal components 65	
Figure 3.11 – Score vector for the first principal component against pixel indexes.....	67
Figure 3.12 – Score vector for the first principal component against pixel indexes (first 8000 points).....	67
Figure 3.13 – Greyscale conversion of data corresponding to (1) a random wavelength, (2) score vector for the first principal component, (3) score vector for the second principal component, (4) score vector for the third principal component (images are ordered from left to right and from top to bottom).....	68
Figure 3.14 – (left) Combination of the first three components into a RGB picture; (right) Contrast enhancement for the RGB conversion.....	69
Figure 3.15 – Score vector for the first principal component versus the one for the second principal component.....	70

Figure 3.16 - Loading vector for the first three principal components for each spectral band	72
Figure 3.17 – Scatter plot for the loading vectors of the first two components	72
Figure 3.18 – (left) Residual matrix for the first principal component; (right) representation of the three residual error matrices as a colour image	73
Figure 3.19 – Decomposition resulting from the application of a two-dimensional wavelet transform to the score matrix of the first component (a), the second (b), the third (c) and a joint representation of the three score matrices in rgb color space (data rescaled and contrast stretched)	75
Figure 3.20 – Example of wavelet expansion using the scaling function in (a) and the wavelet function in (b). In order: (c) original function, (d) approximation, (e) detail, (f) reconstruction from approximation and detail	77
Figure 3.21 – Intensity profiles of row 11 (up) and column 20 (down) for the 1 st pc scores after wavelet transform application, followed by rescaling and contrast-stretching.....	78
Figure 3.22 – Approximated image obtained after wavelet transform application, followed by rescaling and contrast-stretching. Respectively: three decomposition stages (left) and five decomposition stages (right). The images have been zoomed.	79
Figure 3.23 – Decomposition resulting from the application of a two-dimensional wavelet transform to the score matrix of the first component. Results are saturated after 1 and below 0.....	79
Figure 3.24 –Scatter plot for wavelet-transformed score vectors	81
Figure 3.25 – Rescaled and contrast-stretched scatter plot for wavelet-transformed score vectors	82
Figure 3.26 – Dendrogram for the clustering obtained with Euclidean metric and Ward’s linkage, cut-off after 6 clusters. With an x are marked the links that would break with a cut-off of 7 or 8 clusters. The colours are random and do not correspond to the colormap from Figure 3.29.	85
Figure 3.27 – (left) Scatter plot for the wavelet-transformed score vectors, rescaled and stretched. Colours corresponding to clusters on the (right) image.....	85
Figure 3.28 – Clustering results with different “distance” and “inter-cluster distance” metrics	86
Figure 3.29 – Colormaps for the clusters in Figure 3.28. Each colour corresponds to a different group.....	87
Figure 3.30 – Clustering results increasing the number of clusters required.....	87
Figure 3.31 – (a) Greyscale clustering result; (b) Dilation on greyscale clustering result; (c) Labels interchange between groups 1 and 4; (d) Closing operation on the image with interchanged labels.....	89
Figure 3.32 – Results for classifier trained on pre-processed data (left) and on non-pre-processed (right) compared. The colormap indicated the labels corresponding to each color	90

Figure 3.33 – Confusion matrix in the case of a pre-processed (left) or non-pre-processed (right) dataset.....	90
Figure 3.34 – Scatter plot showing the groupings obtained through the classifier. The misclassified datapoints are marked with a cross (+).....	91
Figure 3.35 – Classification of the PCA scores image using the <i>PP-classifier</i> (left) and superimposition of the results on a greyscale version of the original image (right)	92
Figure 3.36 – Unsupervised and supervised classification results in the case of one, two, four or five principal components retained (wavelet transform applied up to the third level of decomposition).....	98
Figure 3.37 – Unsupervised and supervised classification results in the case of two, four and five level of decomposition (three principal components retained).....	100

Appendix

Appendix 1 – Hyperspectral camera datasheet



SPECIM FX17

SMALL, FAST & AFFORDABLE HYPERSPECTRAL CAMERA
SPECIFICALLY DESIGNED FOR INDUSTRIAL MACHINE VISION



FLEXIBILITY

Free wavelength selection from 224 bands within the camera coverage

HIGH SPEED

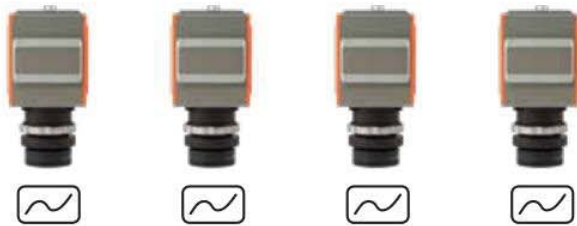
More FPS than you ever need

Example 1: Full range (224 bands) selected = 670 FPS

Example 2: Three specific regions selected (4 bands): > 15 000 FPS

PLUG 'N' PLAY

Every Specim FX unit gives identical results



FAST OPTICS

HIGH SENSITIVITY

Enables good signal with short integration times

HIGH SIGNAL TO NOISE RATIO

Enables better detection accuracy on high speeds

INTEGRATION

Acquisition software & SDK: LUMO Toolkit
Camera interface: GigE Vision or CameraLink



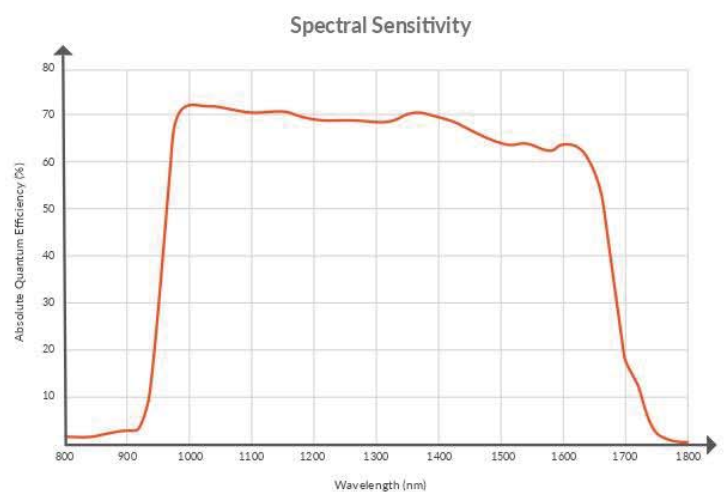
SMALL SIZE

Much smaller footprint than traditional hyperspectral cameras



Spectral Range	900 – 1700 nm *
Spectral Bands	224
Spectral FWHM	8 nm
Spatial Sampling	640 px
Frame Rate	670 FPS full frame 15 000 FPS with 4 bands selected
FOV	38°
F-number	F/1.7
Camera SNR (Peak)	1000:1
Camera Interface	GigE Vision or CameraLink
Dimensions	150 x 85 x 75 mm
Weight	1.56 kg
Integrated shutter	

* The precise range is determined by sensor material characteristics. A typical sensitivity curve of InGaAs sensor is shown in figure.



Appendix 2 – Camera report

Lumo - Scanner measurement report

<http://www.specim.fi>

Sample name:

Date: Wednesday, September 19, 2018

Time: 09:49:17

Error:

Description

Teste1

Camera

Attributes	Value
sensor type	FX17e
sensor id	4200037
frame rate	25
integration time	5
samples	640
active bands	224
spatial binning	1
spectral binning	1
frames recorded	480
frames dropped	0
data format	BIL
calibration pack	C:/Program Files/Specim/4200037/Calibrations/4200037_20180629_FX_calpack.scp

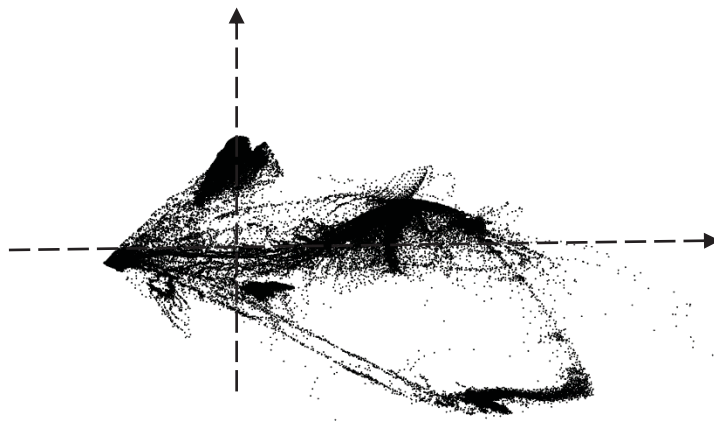
Metadata

Attributes	Value
operator	Vitor Medeiros
description	Teste1
sensor temperature	30
sensor prefix	NIR-UC_DEQ

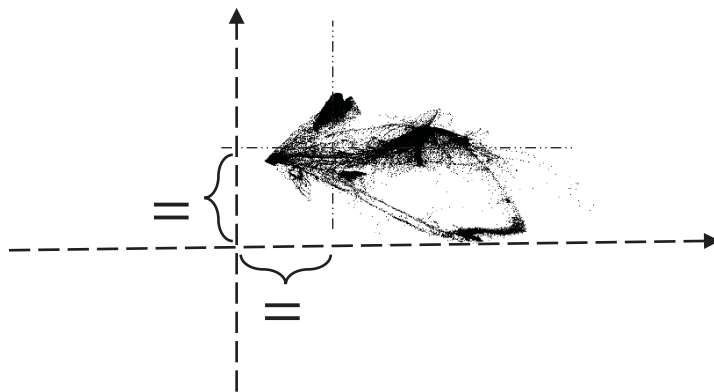
Appendix 3 – Rescaling and contrast-stretching

Visualization of a picture generated by concatenating scores matrices for several principal components further transformed through rescaling and contrast stretching

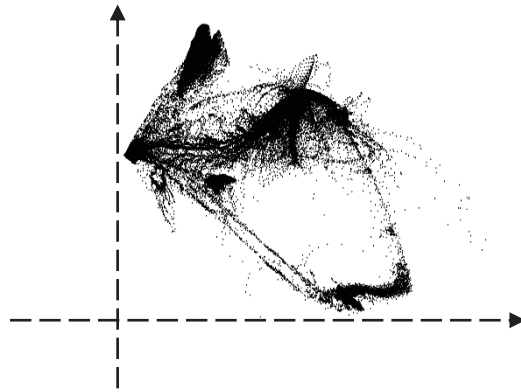
Scatter plot of the scores on PC1 and PC2



Subtracting the absolute minimum and dividing by the difference between absolute maximum and absolute minimum, the point distribution is qualitatively identical, because the points are all divided by the same constant, and it appears translated with respect to the PC axes



Expansion of the narrow input range into a wider output range (contrast-stretching)



The different clusters appear to be easier to distinguish after the stretching procedure, as represented in Figure 3.15 in Chapter 3.