**UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA**

**ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA**

**PROYECTO FIN DE CARRERA**

# *Identificación de tumores cerebrales mediante el estudio de la forma y la composición de los tejidos usando imágenes hiperespectrales*

**TITULACIÓN: Ingeniero de Telecomunicación**

**TUTOR/ES: Gustavo Iván Marrero Callicó**
**Inmaculada García Dópido**

**AUTOR: Miguel Ángel Tejedor Hernández**

**FECHA: Septiembre 2015**

**UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA**

**ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA**

**PROYECTO FIN DE CARRERA**

# *Identificación de tumores cerebrales mediante el estudio de la forma y la composición de los tejidos usando imágenes hiperespectrales*

| | | |
|---|---|---|
| Presidente: | Secretario: | Vocal: |
| Tutores: | | Autor: |

**NOTA:**

**TITULACIÓN: Ingeniero de Telecomunicación**

**TUTOR/ES: Gustavo Iván Marrero Callicó**
**Inmaculada García Dópido**

**AUTOR: Miguel Ángel Tejedor Hernández**

**FECHA: Septiembre 2015**

# RESUMEN

La presente memoria resume el trabajo de investigación realizado por Miguel Ángel Tejedor Hernández con motivo de su proyecto fin de carrera (PFC). En concreto, el presente trabajo describe una comparativa de diferentes cadenas de procesamiento de imágenes hiperespectrales de tumores cerebrales humanos. El documento sigue la estructura clásica de un trabajo de investigación en dicho campo, presentando en primer lugar las motivaciones y objetivos que han motivado la comparativa de diferentes técnicas de análisis hiperespectral respondiendo a una necesidad claramente existente en este campo de estudio, pues actualmente no existen mecanismos fiables para delimitar el tejido tumoral con el fin de extraerlo en su totalidad y de forma precisa durante una operación de neurocirugía. A continuación se realiza un estudio en profundidad del estado del arte en dicho campo, desde el concepto de píxel hiperespectral hasta los algoritmos existentes que fundamentan la base de este estudio. Posteriormente se detallan los módulos de pre-procesado y clasificación que se han combinado en forma de diferentes cadenas de procesamiento orientadas a clasificar datos hiperespectrales de forma supervisada. En este sentido, el núcleo del presente trabajo viene dado por la comparativa de las cadenas de procesamiento consideradas en el marco de dos casos de estudio centrados en la utilización de imágenes hiperespectrales de tejido tumoral, obtenidas mediante sensores hiperespectrales durante intervenciones quirúrgicas y adquiridas de las biopsias cerebrales extraídas durante dichas intervenciones. Como resultado del estudio cuantitativo y comparativo realizado al analizar los resultados de clasificación obtenidos utilizando diferentes cadenas de procesamiento en relación con información de referencia (muestras etiquetadas) disponible para dichas imágenes, se ofrecen una serie de conclusiones y recomendaciones generales acerca del mejor uso posible de los módulos de pre-procesado y clasificación que integran dichas cadenas. Dichas recomendaciones suponen un aspecto innovador en la literatura especializada dedicada al análisis de datos hiperespectrales, y pensamos que serán de gran utilidad para los usuarios de este tipo de datos interesados en aplicaciones relacionadas con la clasificación supervisada de los mismos.

# Contents

# MEMORIA

# Chapter 1
# Introduction

## 1.1. Objectives

The main objective of this dissertation is to analyze different types of processing chain for hyperspectral imaging that allow to obtain accurate and efficient results for this type of data, combining the advantages of each technique applied while minimizing the disadvantages associated with the separate application of each technique. For this, the next overall objective is proposed below: **to study, to evaluate and to compare the different existing techniques for hyperspectral classification process and to draw some conclusions with regard to the efficiency approaches for human brain cancer detection.** In order to achieve this general objective, several specific objectives have also been accomplished:

O1. To develop a state of the art study on hyperspectral imaging in the medical field (data format and representation), acquiring the necessary knowledge of the hyperspectral analysis.

O2. To gain knowledge of tools and programs management to work with this type of processing data.

O3. To analyze the images used to better understand their behavior in certain applications (source, characteristics, special properties, types of tissue...).

O4. To compare the most important data processing techniques currently in the literature for hyperspectral imaging and to establish those that provide better results or more accurately.

O5. To perform a comprehensive study regarding supervised algorithm SVM, one of the main techniques for supervised classification.

O6. To implement a set of processing chains using the chosen techniques and classification testing.

O7. To evaluate the proposed experiments focusing on the influence of using different types of pre-processes before the classification process.

O8. To design and develop a final system for hyperspectral image classification in order to detect tumor samples.

O9. To validate a final classification system using real hyperspectral images of human brain tumors.

O10. To draw conclusions from quantitative and comparative study performed, and propose possible future work.

## 1.2. Context and motivations

The work developed in this dissertation is the analysis and study of different automatic processing techniques for hyperspectral classification focused on of human brain tumour samples using supervised methods. This work is included in the actual research lines funded by the European Commission through the FP7 FET Open programme ICT-2011.9.2, the European Project HELICoiD "HypErspectral Imaging Cancer Detection" under Grant Agreement 618080.

Hyperspectral images are an extension of the concept of digital image, in the sense that each pixel in a hyperspectral image is not only formed by a single discrete value, but by a wide range of values for different spectral measurements recorded by a sensor or measuring instrument in different wavelength values. The collection of all the wavelength values (one per spectral band) which are associated to a given pixel is called a *spectral signature*. As a result, we can understand a hyperspectral image as a collection of spectroscopic measurements that provide very detailed of information on the properties of the materials appearing on the scene.

This fact involves the provision of a large amount of information with a high level of detail. The scientific community dedicated to the analysis of hyperspectral data has identified the need to interpret such data properly and obtain relevant information for different fields with little effort [1]. Therefore, the basis for defining and testing a flexible chain of collection and processing of hyperspectral data to produce efficient results should be set.

In the literature, there are a variety of methodologies and techniques applicable, which may be considered as combinable functional blocks within the entire processing chain. Considering the main purpose of processing as the classification and characterization of hyperspectral pixels to produce a thematic map that identifies different classes or regions of interest in the image, the steps which are executed a priori and a posteriori are known as pre-processing and post-processing techniques respectively. This work has taken into account large part of the research for this discipline and is intended to increase or deepen the understanding of how the choice of a particular pre-processing affects, besides the choice of the classification algorithm, to obtain high levels of accuracy and make an approach to the most appropriate sequence of applications to define a general standard processing chain or a processing chain applicable to a given case.

The number and variety of processing tasks in hyperspectral imaging is enormous [2]. However, this dissertation is mainly focused on the integration of three of those techniques:

- Classification consists of assigning a label (class) to each pixel of a hyperspectral data cube [2].
- Dimensionality reduction consists of reducing the dimensionality of the input hyperspectral scene to facilitate subsequent processing tasks [3].
- Spectral unmixing consists of estimating the fraction of the pixel area covered by each material present in the scene [4].

In the present work, these techniques will be applied to two kinds of samples: *ex vivo* samples and *in vivo* samples, in which it is intended to classify different types of tissues using a supervised architecture called Support Vector Machine (SVM). Finally, it should be noted that experiments performed represent a part of the whole process suffered by a hyperspectral imaging since it is collected by the sensor until the user takes advantage of the interpretation made. Therefore, the accuracy that can be obtained will always be conditioned by the transformations carried out on the data previously and effectiveness in the definition of the parameters which apply.

## 1.3. Petitioner

The development of this Master Thesis is carried out as a request from the Faculty of Telecommunications Engineering and Electronics (EITE) of the University of Las Palmas de Gran Canaria (ULPGC) as an official requirement for obtaining the title of Engineer in Telecommunications, once the student has passed all the credits that comprise this degree.

## 1.4. Memory organization

This dissertation is organized into six chapters as follows.

- Chapter 2 provides an introduction and review of the state of the art in the context of hyperspectral imaging in the medical field, paying particular attention to the supervised classification techniques in this field and different preprocessing methods, which are described in more detail in chapter 3.
- Chapter 4 presents the different process considered in this research.
- Chapter 5 performs a comprehensive experimental validation of the results obtained after applying the different processing chains considered to hyperspectral data of tumor tissues. This chapter also includes a general discussion of the results obtained by the different processing chains in different study cases, extrapolating conclusions about the performance of each processing chain in terms of the accuracy obtained in the classification.
- Chapter 6 provides the conclusions about the studies conducted and future research lines.

- The paper concludes with the bibliographic references taken into account in the drafting of the report and other references enabling the extension of the concepts presented in this work.

To conclude this chapter, Table 1.1 provides a list of the acronyms used throughout the dissertation document. Hereinafter, these acronyms will be used instead of the full terms for simplicity.

| **Acronyms** | |
|---|---|
| AA | Average Accuracy |
| AMEE | Automatic Morphological Endmember Extraction |
| ANC | Abundance Non-negativity Constraint |
| ASC | Abundance Sum-to-one Constraint |
| FastICA | Fast Independent Component Analysis |
| FCLSU | Fully Constrained Linear Spectral Unmixing |
| HELiCoiD | HypErspectraL Imaging Cancer Detection |
| HSI | Hyperspectral imaging |
| HySime | Hyperspectral Subspace Identication by Minimum Error |
| ICA | Independent Component Analysis |
| JADE | Joint Diagonalization of Eigenmatrices |
| K | Kappa coefficient |
| KPCA | Kernel Principal Component Analysis |
| LIBSVM | Library of SVM [Online: https://www.csie.ntu.edu.tw/~cjlin/libsvm/] |
| LPP | Locality Preserving Projections |
| LSU | Linear Spectral Unmixing |
| MIR | Mid-infrared |
| ML | Machine Learning |
| MNF | Minimum Noise Fraction |
| NIR | Near-infrared |
| NPE | Neighborhood Preserving Embedding |
| OA | Overall Accuracy |
| OSP | Orthogonal Subspace Projection |
| PCA | Principal Component Analysis |
| RBF | Gaussian Radial Basis function |
| SPP | Spatial Pre-Processing |
| SSEE | Spatial Spectral Endmember Extraction |
| SVM | Support Vector Machine |
| UV | Ultraviolet |
| VCA | Vertex Component Analysis |
| VD | Virtual Dimensionality |
| VIS | Visible |

*Table 1.1: List of acronyms used in this dissertation.*

# Chapter 2

# 2. Hyperspectral Imaging

## 2.1. Hyperspectral imaging concept

The observation of a particular object is based on the capture of electromagnetic radiation from the interaction between the object and the radiation source by a measuring instrument or sensor. Electromagnetic radiation received several names depending on the wavelength that characterizes it, as shown in Figure 2.1. For measuring the emitted or reflected radiation by a given surface is necessary to quantify the amount of energy flux that proceeds from the same. For this the radiance measure is used, which depends on factors such as the perceived brightness, reflectance, viewing angles, among others. [5]

Spectral detection techniques are based on the fact that all materials in the real world reflect, absorb and emit electromagnetic energy differently in different wavelengths. [6]

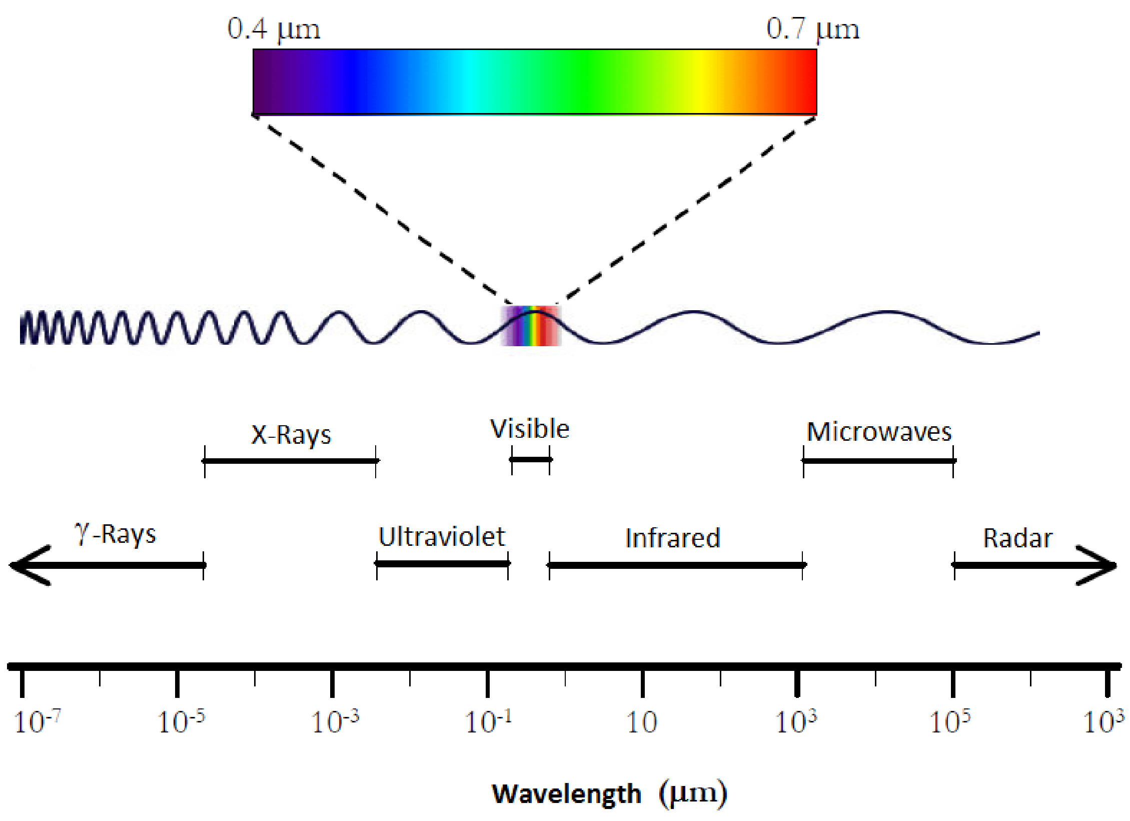


*Figure 2.1: Electromagnetic spectrum.*

Currently, there is a wide range of instruments or sensors capable of measuring spectral singularities in different wavelengths [7]. The availability of these instruments has provided a redefinition of the digital image concept through the extension of the idea of pixel.

It is important to remember that the associated value with each pixel is defined by a numerical value called digital level. This is a numeric value, not visual, but can easily be translated into a visual intensity or gray level by any digital-analog converter. Thus, in a purely spatial schema one pixel is constituted by a single discrete value, while in a spectral schema one pixel consists of a set of values. These values may be understood as N-dimensional vectors [8], where N is the number of spectral bands in which the sensor measures information.

Extending the concept of pixel results in what is known as multidimensional image. The order of magnitude of N allows a distinction when talking about multidimensional images. Thus, when the value of N is small, typically a few spectral bands [9], one speaks of multispectral images, whereas when the order of N is hundreds of bands [10], there is talk of hyperspectral imaging, which are also known as hypercubes.

Hyperspectral imaging (HSI), like other spectral imaging, collects and processes information from across the electromagnetic spectrum. The goal of hyperspectral imaging is to obtain the spectrum for each pixel in the image of a scene, with the purpose of finding objects, identifying materials, or detecting processes. [11][12]

Much as the human eye sees visible light in three bands (red, green, and blue), spectral imaging divides the spectrum into many more bands. This technique of dividing images into bands can be extended beyond the visible as shown in figure [13]. In hyperspectral imaging, the recorded spectra have fine wavelength resolution and cover a wide range of wavelengths.

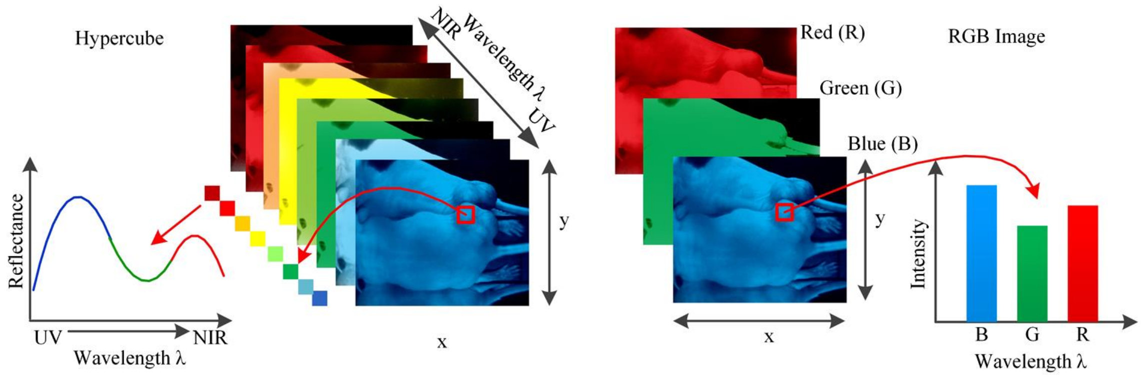


*Figure 2.2: Comparison between hypercube and RGB image. Hypercube is a three-dimensional dataset of a two-dimensional image on each wavelength. The lower left is the reflectance curve (spectral signature) of a pixel in the image. RGB color image only has three image bands on red, green, and blue wavelengths respectively. The lower right is the intensity curve of a pixel in the RGB image.*

Hyperspectral sensors look at objects using a vast portion of the electromagnetic spectrum. Certain objects leave unique *fingerprints* in the electromagnetic spectrum. If the difference between each of these wavelengths is a few nanometers, it can be obtained for each pixel an "almost continuous" spectral response of the photographed material at that pixel. This response is called spectral signature which is unique to each material. Known as spectral signatures, these *fingerprints* enable identification of the materials that make up a scanned object and it can be used to identify specific substances in an image. For example, a spectral signature for tumor tissue helps surgeons find tumor areas.

Most hyperspectral image pixels are mixed pixels. This is because the spatial resolution of the sensor cannot separate different materials in a pixel. The denotation of mixed pixel is given by the presence of pixels composed by the combination of several pure spectral signatures. In the process of spectral unmixing we find an efficient possibility to express the composition of each mixed pixel in proportion to the pure pixels present. [14][15]

With the advantage of acquiring two-dimensional images across a wide range of electromagnetic spectrum, HSI has been applied to numerous areas, engineers have built hyperspectral sensors and processing systems for applications in astronomy, agriculture, biomedical imaging, geosciences, physics, and surveillance. [13]

The primary disadvantages of these techniques are cost and complexity. Fast computers, sensitive detectors, and large data storage capacities are needed for analyzing hyperspectral data. Significant data storage capacity is necessary since hyperspectral cubes are large, multidimensional datasets, potentially exceeding hundreds of megabytes. All of these factors greatly increase the cost of acquiring and processing hyperspectral data.

## 2.2. Medical hyperspectral imaging

### 2.2.1. Brain tumours

Due to the increase in the incidence and mortality from brain tumor in world population in recent decades, the number of research papers related to its diagnosis has grown exponentially. An early diagnosis of such diseases can be vital to prevent a benign tumor evolves into a more aggressive cancer.

Brain tumors are due to abnormal growth of cells derived from brain components in the case of primary tumors or tumor cells localized elsewhere in the organism in the case of metastases. According to evolution, size and dimension can be grouped into four degrees of danger. This classification is ranging from grade I tumors (benign tumors of slow growth) to grade IV tumors (malignant cancers with very rapid growth). [16]

The 77% of malignant brain cancers belongs to the group of tumors called gliomas. These tumors affect glial cells responsible for supporting neurons and information brain processing. Depending on the type of glial cells affected (astrocytes, oligodendrocytes, ependymal cells, etc.) are different types of gliomas, being multiform glioblastoma (astrocytoma grade IV) the most aggressive and less likely to survive [17][18]. Figure 2.3 shows magnetic resonance imaging of a patient afflicted with a multiform glioblastoma.

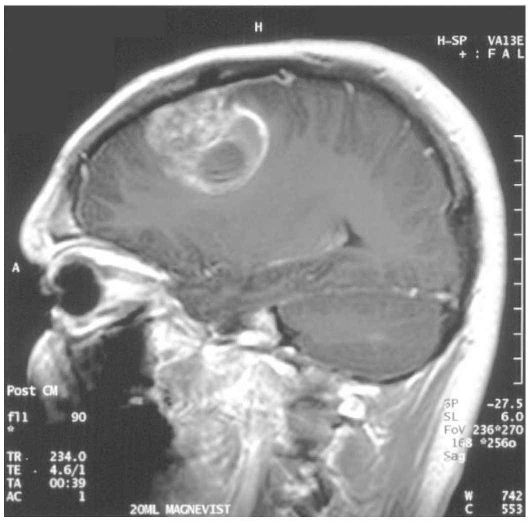


*Figure 2.3: Magnetic resonance of patient suffering from multiform glioblastoma.*

Symptoms and treatment of these tumors depend largely on the patient's age, the type of tumor and its location in the brain. However, these tumors tend to infiltrate into healthy brain tissue, so that surgery is complex and sometimes impossible. That is why it is vitally important research and study of non-invasive techniques that provide the most accurate diagnosis possible.

Proper removal (resection) of brain tumor eliminates malignant tissue and prevents the tumor is reproduced. Increase resection area with an additional safety margin maximizes the chances of patient survival, but in the case of brain tumors, excessive resection can lead to significant damage of all kinds (motor, cognitive, visual, etc.). For this reason, a technique to determine accurately and minimally invasive glioma limits is necessary.

### 2.2.2. Hyperspectral imaging applied to tumour detection

Spectral imaging is a technology that integrates conventional imaging and spectroscopy methods to obtain both spatial and spectral information from an object. Although this technology was originally developed for remote sensing, it has been extended to the biomedical engineering field as a powerful analytical tool for biological and biomedical research [19].

Hyperspectral imaging field is an emerging imaging modality for medical applications. It offers great potential for noninvasive disease diagnosis and surgical guidance. Light delivered to biological tissue undergoes multiple scattering from inhomogeneity of biological structures and absorption primarily in hemoglobin, melanin, and water as it propagates through the tissue [20][21]. One of the important advantages of this technique is that it can acquire reflectance, absorption, or fluorescence spectrum for each pixel in the image. It is assumed that the absorption, fluorescence, and scattering characteristics of tissue change during the progression of disease [22], which can be used to detect the biochemical changes of objects that cannot be identified with traditional gray or color imaging methods [19]. Therefore, the reflected, fluorescent, and transmitted light from tissue captured by HSI carries quantitative diagnostic information about tissue pathology [22-25].

In recent years, advances in hyperspectral cameras, image analysis methods, and computational power make it possible for many exciting applications in the medical field. These applications mainly cover the ultraviolet (UV), visible (VIS), and near-infrared (near-IR or NIR) regions. Table 2.1 defines the spectral range from UV to mid-IR (200 to 25,000 nm) [26].

| Short name | Full name | Spectral range (nm) |
|---|---|---|
| UV | Ultraviolet | 200 to 400 |
| VIS | Visible | 400 to 780 |
| NIR/near-IR | Near-infrared | 780 to 2500 |
| MIR/mid-IR | Mid-infrared | 2500 to 25.000 |

*Table 2.1: Spectral range definitions.*

Visible light penetrates only 1 to 2 mm below the skin and thus obtains information from the sub-papillary [27], while light in the NIR region penetrates deeper into the tissue than VIS or mid-IR radiation [28]. NIR light is preferred for surgical guidance due to its deep penetration into the tissue, which can help the surgeon see through connective tissue for visualizing critical anatomical structures of interest that are not visible and detecting molecules with detectible spectra [13]. By expanding light beyond the visual spectrum, additional information can be obtained to further characterize the cells of interest [29].

HSI acquires a three-dimensional dataset called hypercube, with two spatial dimensions and one spectral dimension. Figure 2.4 [19] shows the concept of the hypercube data captured by a spectral imaging system. These dates can be visualized as a three-dimensional (3-D) cube or a stack of multiple two-dimensional (2-D) images because of its intrinsic structure, in which the cube face is a function of the spatial coordinates and the depth is a function of wavelength [19]. Spatially resolved spectral imaging obtained by HSI provides diagnostic information about the tissue physiology, morphology, and composition [13].

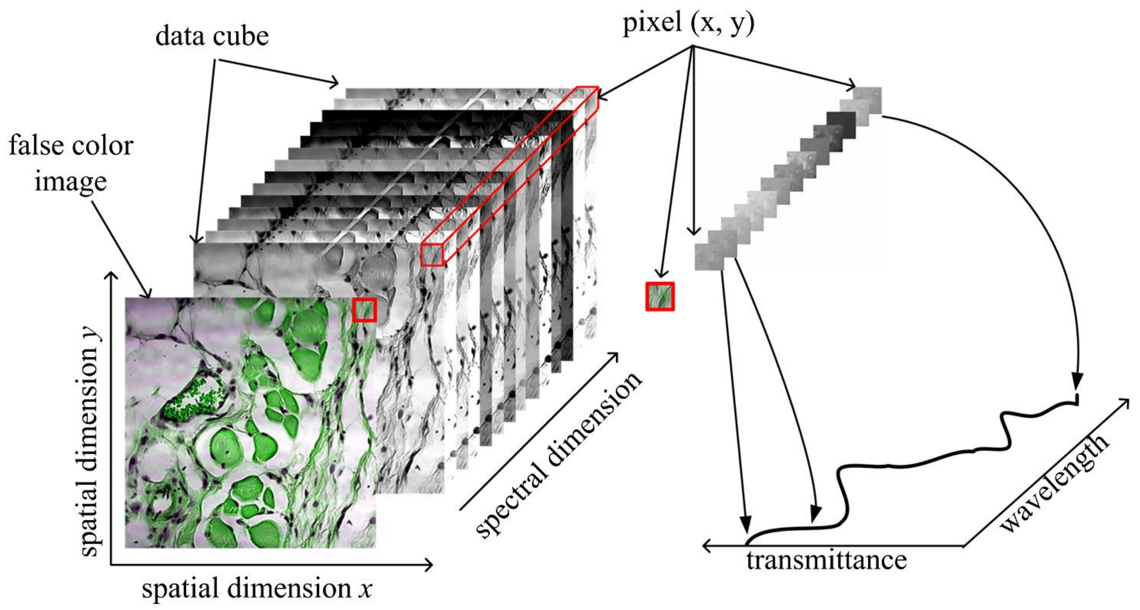


*Figure 2.4: The concept of spectral data cube. The data cube contains two spatial dimensions (x and y) and one spectral dimension, in which the cube face is a function of the spatial coordinates and the depth is a function of wavelength.*

According to the electromagnetic theory, different biochemical constituents commonly have different spectral signatures [30]. These signatures are usually generated by the interactions between materials and electromagnetic waves, such as electron transition, atomic and molecular vibration or rotation. The biological and pathological changes in tissues and organs also have a close relationship with the spectra. Spectral characteristics in different wavelength regions yield a distinguishable spectral signature, making pathological changes distinguishable. Therefore, the spectral imaging technology also can be extended to the biomedical engineering field to estimate the physiological status of biological tissues, since it can take advantage of the spatial relationships among the different spectra in a neighborhood. This technology opens new prospects for life science by which scientists can identify and quantify the relationships among biologically active molecules, observe living organisms noninvasively, perform histopathological and fluorescent analyses, and enhance biological understanding of diseases [19].

Image analysis enables the extraction of diagnostically useful information from a large medical hyperspectral dataset at the tissue, cellular, and molecular levels and is, therefore, critical for disease screening, diagnosis, and treatment. Hypercube with high spatial and spectral resolution may potentially contain more diagnostic information. However, high spatial and spectral dimensions also make it difficult to perform automatic analysis of hyperspectral data. In particular, it is complex in many aspects: (1) high data redundancy due to high correlation in the adjacent bands, (2) variability of hyperspectral signatures, and (3) curse of dimensionality [7]. With abundant spatial and spectral information available, advanced image classification methods for hyperspectral datasets are required to extract, unmix, and classify relevant spectral information. The goal is not only to discriminate between different tissues (such as healthy and malignant tissue) and provide diagnostic maps, but also to decompose mixtures into the spectra of pure molecular constituents and correlate these molecular fingerprints (biomarkers) with disease states. Although hyperspectral image analysis methods have been intensively investigated in the remote sensing area, their development and application in medical domain lag far behind. The relationships between spectral features and underlying biomedical mechanisms are not well understood. The basic steps for hyperspectral image analysis generally involve preprocessing, feature extraction and feature selection, and unmixing and/or classification [13].

### 2.2.3. Hyperspectral imaging data sets

Throughout this dissertation two different data are used: *ex vivo* samples and *in vivo* samples. In the first stage of this work, all the experiments were performed with *ex vivo* samples. In the second stage, once the *in vivo* samples were obtained, the main processing chains were repeated using such data. This section explains how datasets used along this work were obtained.

### 2.2.3.1. *Ex vivo* samples

The *ex vivo* samples are formed by living tissues which were removed from a real patient in a surgery. Then, several pictures were taken with a hyperspectral camera from different angles. The camera used has 1040 spectral bands ranging from 380 to 1028 nanometers. For each picture the hyperspectral data cube is formed from the files generated by the hyperspectral camera. In this regard, we have 11 hyperspectral data cube, which have 3 different angles and can contain up to 3 different tissue types: healthy, tumor and necrosis.

The process to obtain the samples is detailed below:

1. The hyperspectral data cube is formed from the files generated by the hyperspectral camera.

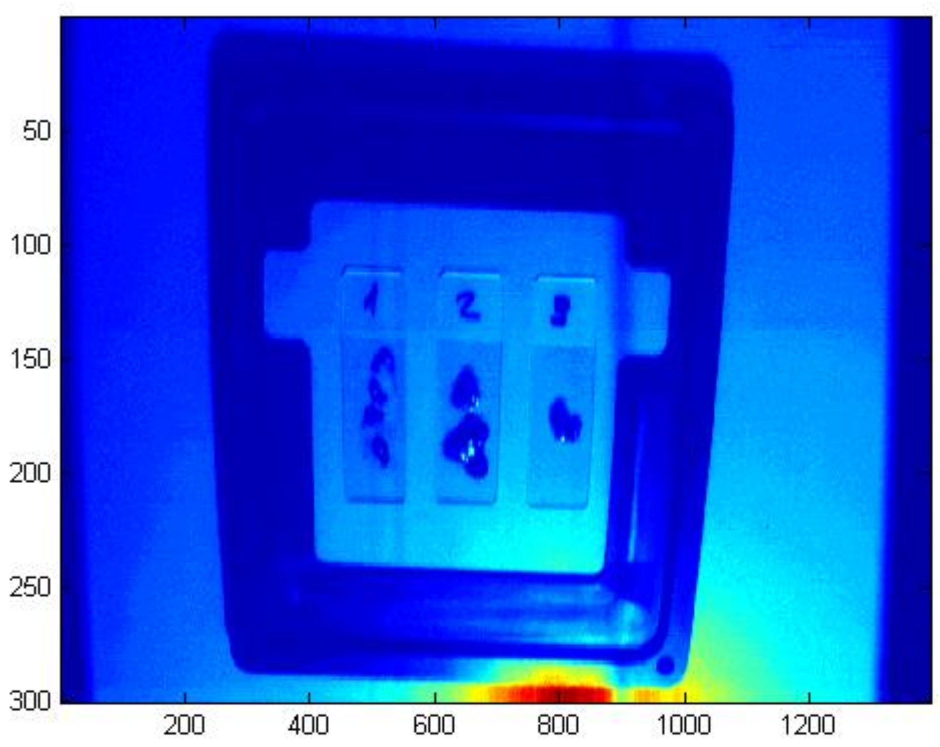


*Figure 2.5: Hypercube 2 band 350.*

2. The image is cropped in order to it can be processed more easily thanks to its reduced size and for separating the different samples (healthy, tumor and necrosis) in each image.

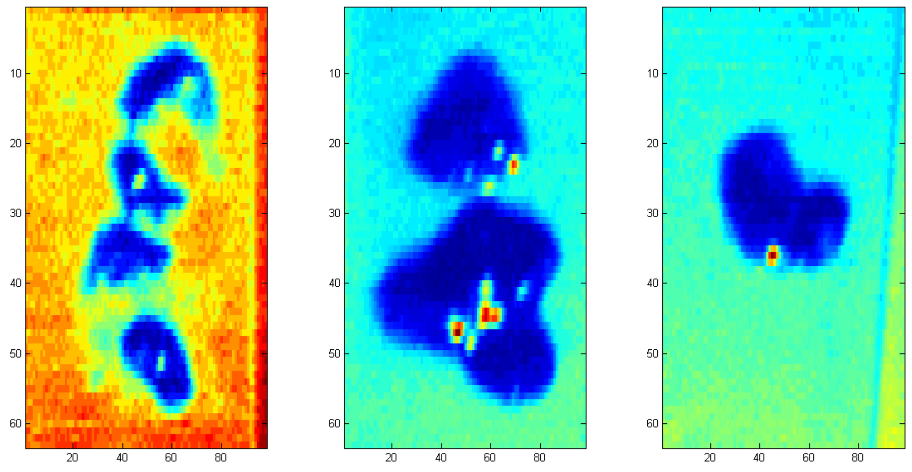


Figure 2.6: Hypercube 2 band 350 separated by classes: healthy, tumor and necrosis.

3. The samples are filtered because these contain a lot of noise. The filters used are Hyperspectral Subspace Identification by Minimum Error (HySime) and smooth, and these are explained in detail in the following sections.
4. Finally the contour of the sample is determined by removing the rest of the image, in order to determine the number of samples of each type at our disposal. The contour functions are explained in detail in the following sections.

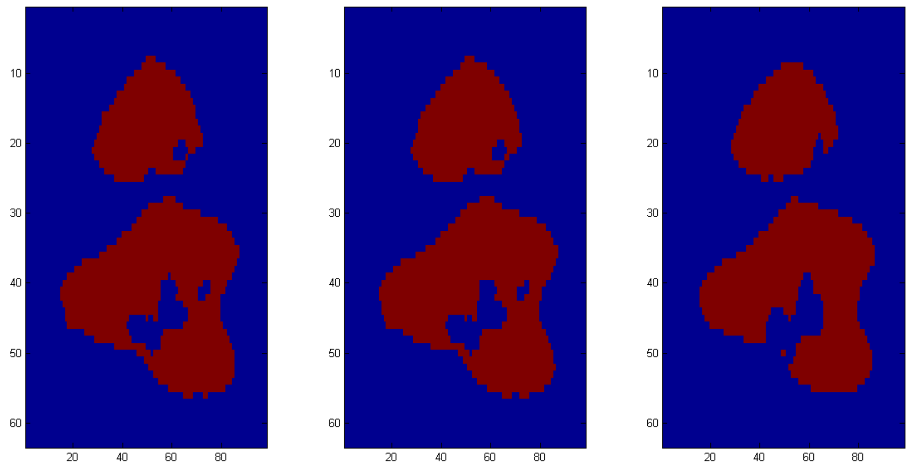


Figure 2.7: Tumor sample hypercube 2 without filter, HySime filter and smooth filter.

In Table 2.2 is presented the number of samples from each hypercube:

| | Healthy | | | Tumor | | | Necrosis | | |
|---|---|---|---|---|---|---|---|---|---|
| **Number of hypercube** | No filter | HySime filter | Smooth filter | No filter | HySime filter | Smooth filter | No filter | HySime filter | Smooth filter |
| **1** | 972 | 990 | 1020 | 1956 | 1978 | 1925 | 565 | 572 | 531 |
| **2** | 902 | 918 | 924 | 1723 | 1741 | 1619 | 563 | 572 | 531 |
| **3** | 1043 | 1063 | 1042 | 1807 | 1824 | 1714 | 402 | 411 | 341 |
| **4** | 811 | 807 | 741 | 1787 | 1785 | 1789 | - | - | - |
| **5** | - | - | - | - | - | - | 441 | 443 | 423 |
| **6** | 802 | 803 | 742 | - | - | - | - | - | - |
| **7** | - | - | - | 1928 | 1921 | 1939 | - | - | - |
| **8** | - | - | - | 1911 | 1906 | 1895 | - | - | - |
| **9** | - | - | - | - | - | - | 421 | 428 | 392 |
| **10** | - | - | - | - | - | - | 439 | 436 | 421 |
| **11** | - | - | - | - | - | - | 604 | 605 | 591 |

*Table 2.2: Samples in each hypercube.*

In this table we can see the samples extracted from each hypercube, which containing samples of some of the possible classes available in one of the possible inclinations. In this sense, a "-" represents that a given hypercube has no samples of a particular class.

The samples obtained from each hypercube are properly labeled and mixed regardless of its inclination, resulting in a dataset with 3 different classes: healthy, tumor and necrosis, whose spectral signatures are shown in Figure 2.8.

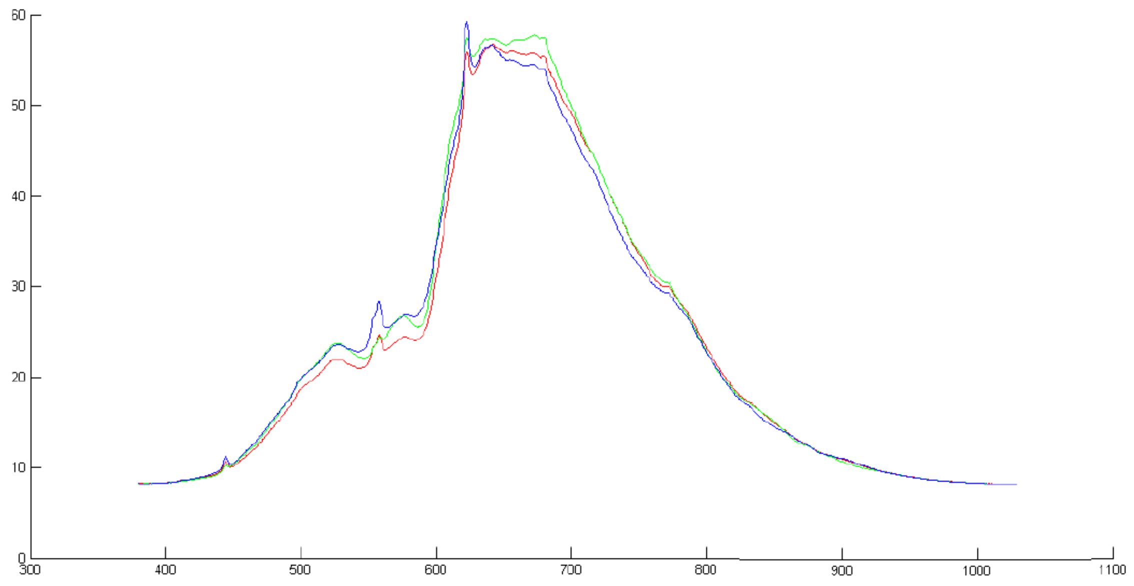


*Figure 2.8: Spectral signatures: healthy (green), tumor (red) and necrosis (blue).*

### 2.2.3.2. *In vivo* samples

In this case the hyperspectral images were taken during the surgery. Subsequently, some well-known areas of the image were selected, calibrated and properly labeled, forming an image dataset. In this set of samples we get 3 classes: healthy, tumor and vein.

In fact dissertation two cameras have been used to obtain *in vivo* samples from surgeries:

- Visible and near-infrared (VNIR): whose portion of the electromagnetic spectrum has wavelengths between approximately 380 and 1000 nanometers. This camera has 826 bands.
- Near-infrared (NIR): whose portion of the electromagnetic spectrum has wavelengths between approximately 900 and 1700 nanometers. This camera has 172 bands.

Table 2.3 describes the number of samples from each patient associated to each camera:

| | Healthy | | Tumor | | Vein | |
|---|---|---|---|---|---|---|
| **Number of patient** | **VNIR** | **NIR** | **VNIR** | **NIR** | **VNIR** | **NIR** |
| **4** | 256 | 20 | 408 | 14 | - | - |
| **5** | 252 | 30 | 156 | 27 | 30 | 4 |
| **7** | 214 | 49 | 558 | 38 | 393 | 56 |
| **8** | 581 | - | 1270 | - | 169 | - |
| **10** | 63 | 12 | 216 | 15 | - | - |

*Table 2.3: Samples from each patient.*

A "-" means no samples of a particular class to one of the cameras.

## 2.3. Hyperspectral analysis techniques

The hyperspectral analysis is based on the ability of hyperspectral sensors to acquire digital images in a large number of spectral channels very close to each other, obtaining for each pixel a characteristic spectral signature of each material that will be used in the analysis process [7]. These images can be represented as a data cube, with two dimensions to represent the spatial location of a pixel, and a third dimension representing the spectral singularity of each pixel at different wavelengths. This process facilitates the identification and quantification of the materials present in the scene [31][32]. Figure 2.9 illustrates the process of hyperspectral analysis using a simple diagram.

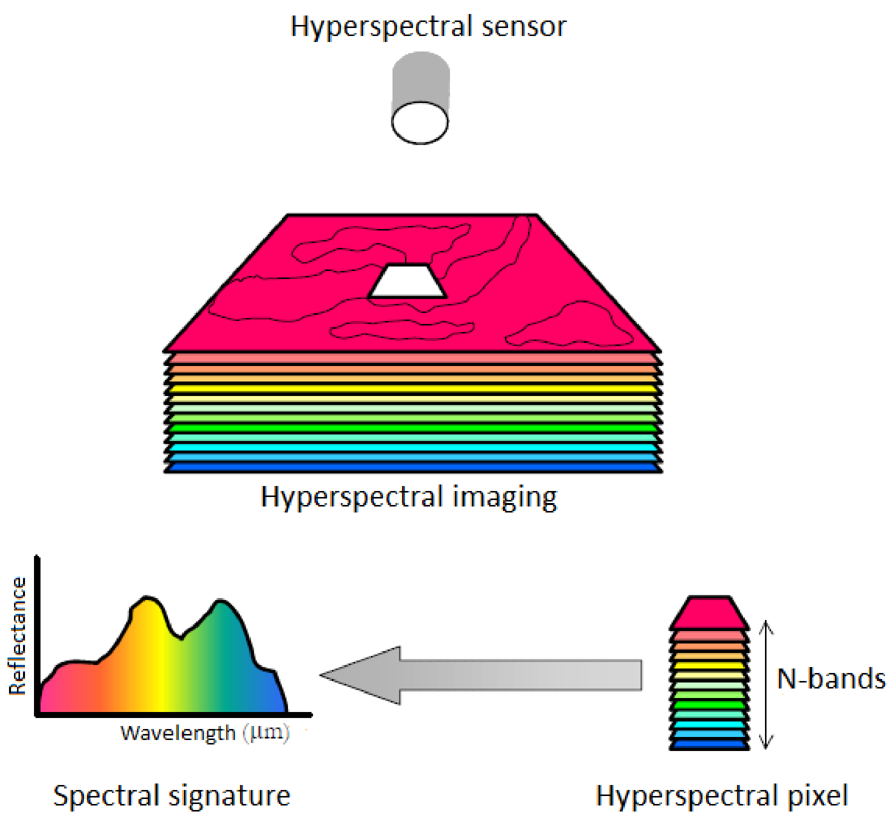


*Figure 2.9: Hyperspectral analysis procedure.*

It is known that the potential of these images is the large amount of information and that allows distinguishing classes and objectives in more detail. But this advantage also becomes a disadvantage when it does not have computing power enough to process and store these hundreds of bands. Then, we are faced with problems of high dimensionality of the data and data redundancy. The high dimensionality can be appreciated if we get an idea of the total size of an image like these, multiplying the pixel size in bits per the size of an image or single band per the total number of bands. The redundancy of information means the repetition of many spectral patterns and it can become quite significant, resulting in many cases inconvenient when statistical classification methods are wanted to use. For this reason, approximations or geometric and non-parametric techniques are most appropriate in many cases.

Therefore, it is necessary to perform a set of activities and processing techniques both hardware and software capable of handling effectively the inherent complexity of the hyperspectral data (high dimensionality) [33]. Although the processing chain for hyperspectral data is not an easy procedure to define consistently, within the framework of this dissertation a set of recommendations for the definition of an appropriate processing chain for hyperspectral data have been introduced [34].

The first part of this processing chain is a specific process to obtain the data hypercube and provide noise-free samples, giving the data ready for its processing and target identification. It is in turn divided into the following steps:

- Data hypercube is formed from the image obtained by the camera.
- Spatial reduction, in order to reduce de image size for processing it more easily.
- Data calibration.
- De-noising filtering.
- Samples extraction from data hypercube using contour functions.
- Samples normalization.

Once the data of interest have been obtained and preprocessed to be used correctly, there is the need to extract relevant information from the collected data sets. Then, the second part of the data processing chain is further divided into the following additional steps:

- Feature extraction in order to reduce the high dimensionality and study the nature of each tissue.
- Classification.

It is important to emphasize that any data processing chain in any scientific field has to be flexible and adapt not only to its application on different scenarios, but also the various types of resolutions that provide different spectral and spatial variations of the instruments.

The next section explains what the main goal in this processing is: the hyperspectral data classification. The processing chains used before will be described in detail in the next chapter.

## 2.4. Hyperspectral data classification

The simplest way to address the problem of pixels classification in a hyperspectral image is to consider that the pixels of interest are composed of a single material, using conventional pattern classification techniques [35] but more accurately, because of the high number of spectral bands available.

In practice, the use of hyperspectral sensors allow a better determination of the internal composition of each pixel, which rarely will consist of a single material, since the phenomenon of the mixture is very common in the real world, regardless of the spatial scale considered [36].

There exists a set pattern classification techniques that perform the interpretation of a scene on the basis of assigning a label or individual classification for each of the pixels of the scene. These techniques offer interesting results in certain applications, particularly those highlighted below.

- **Thematic classification:** Classification techniques have been used successfully in applications that aim to obtaining a thematic map in which each pixel in the hyperspectral image is properly labeled as belonging to a particular class [37]. There may be an additional class called “background” or “others” representing the pixels that are not classified in any of the previous classes. The ideal result is achieved when all classes are mutually exclusive of each other, including the class “background”. The key task in this type of application is usually determining the number of classes and its characterization in terms of training data or ground-truth information.
- **Targets detection:** Classification techniques have also been used very extensively in targets detection applications in hyperspectral imaging [38]. In these applications, the main objective is the identification of a specific material or object (called target in the literature) between all pixels of the image.

Conceptually, the two problems mentioned can be considered a binary classification problem:

- In the thematic classification, there are several possible classes associated with different objects. The goal is to ultimately determine the presence or absence of each of the objects considered in each pixel, situation that can be expressed as a binary classification problem [39].

- In the targets detection the pixels are classified into two classes called "target" and "background", depending on whether they contain the target sought or not.

The binary classification problem can be formulated mathematically as follows. Let $R$ the $N$-dimensional space formed by all pixels in the hyperspectral image. Given a $N$-dimensional array $\boldsymbol{u} = (u_1, u_2, \dots, u_N)^T$ associated with a specific pixel. The binary classification consists of dividing the space $R$ into two regions, $R_o$ and $R_f$, such that $\boldsymbol{u}$ is classified as "target" if $\boldsymbol{u} \in R_o$ and "background" if $\boldsymbol{u} \in R_f$. This problem can be illustrated graphically using a scatterplot between two bit correlated bands of hyperspectral imaging, as shown in Figure 2.10.

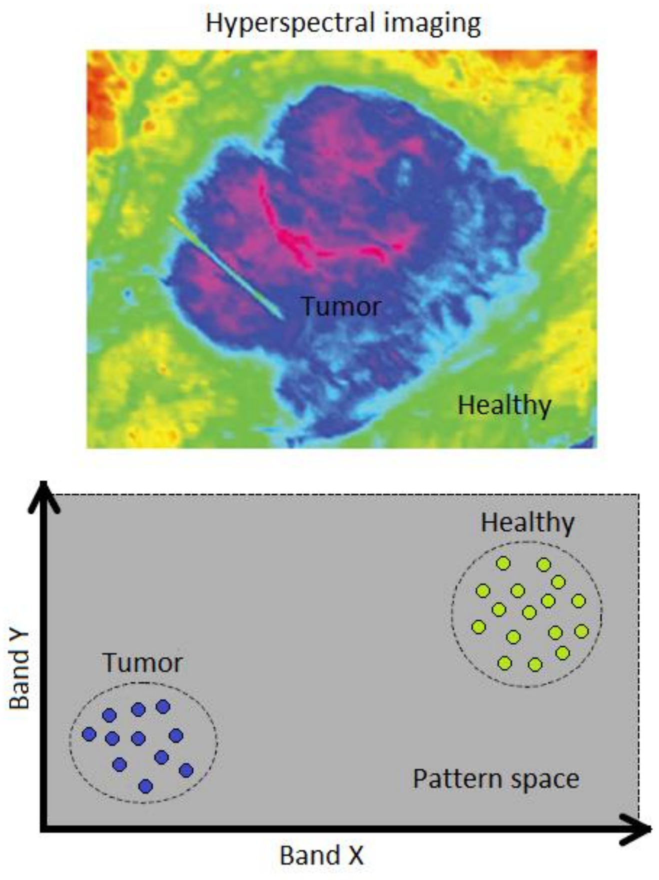


*Figure 2.10: Graphic illustration of the classification problem in hyperspectral imaging.*

As shown in Figure 2.10, the ideal situation in a classification problem occurs when the separation between target and background is clearly defined in clearly distinguishable clusters.

The hyperspectral image classification algorithms can be divided into two broad categories [37]: supervised algorithms and unsupervised algorithms:

- **Unsupervised algorithms:** These algorithms assume that there is no a priori knowledge about existing classes. The aim of these techniques is to identify automatically classes or groups of pixels using a similarity metric.
- **Supervised algorithms:** It start with some knowledge about existing classes, from which it can be derived some classification criteria. This approach tends to be given by a preliminary step in which spectral signatures of the existing classes are selected.

## 2.4.1. Unsupervised classification algorithms

The pixels classification techniques in an unsupervised way for hyperspectral imaging are in full development phase [7]. Among the existing techniques highlights the *K-Means* method [40], which supposes the existence of *K* classes (parameter to be determined a priori) and performs a grouping of the pixels of the image in such classes, using purely statistical methods of vector quantization based on the average spectra of these classes.

Moreover, the *ISODATA* method [41] also requires initialization of a *K* parameter relating to the number of desired classes, prior to the execution of the algorithm. In addition, this method requires information on the minimum number of signatures belonging to one class. If the initial value of *K* is low, the dispersion between classes can be very high. Conversely, if the initial value of *K* is high, the distance between classes can be very small, causing the partitioning of the same class in several classes similar to each other.

Overall, the recent literature shows that the results obtained by these two techniques have not been very satisfactory, except in very specific applications [42].

## 2.4.2. Supervised classification algorithms

In supervised classification techniques highlights the *matched filters* and *Spectral Angle Mapper* (SAM) method, both based on first-order statistics [43]. Within this category may also be other classifiers as the *nearest neighbor*, *minimum distance*, *parallelepiped* or *maximum likelihood* techniques. However, this dissertation is focus on the *Support Vector*

*Machine* (SVM) technique which has demonstrated excellent performance when working with high-dimensional data such as hyperspectral data.

Once presented the most common techniques of hyperspectral image classification, this section is concluded highlighting some techniques used to evaluate the performance of these algorithms.

### 2.4.3. Evaluation metrics used in classification algorithms

The large number of existing techniques as well as the continued proliferation of new methodologies makes clear the need for comparative schemes or metrics to qualitatively analyze the performance of new methodologies rose, contrasting its results with those provided by the existing ones. Most of the evaluation techniques of digital image analysis algorithms are based on the concept of ground-truth [44]. Ideally it is possible to define the concept of ground-truth as the optimal classification or interpretation result which should get an algorithm [45][46].

The ground-truth usually comes characterized by relevant information about the properties in the real world of a set of targets that are desired to identify or characterize. This information is usually obtained by measurements made directly in the study area covered by the image [47], although it is also possible to obtain ground-truth information by applying algorithmic techniques [6]. In any case, the first alternative is the more reliable, but it can be expensive due to the need to label all the samples obtained [48].

Assuming the existence of ground-truth information, there are several methodologies to compare that information with the results provided by an image analysis algorithm. This section provides a brief description of the different metrics that can be applied to assess the ability of a computer algorithm in terms of classification and identification of targets of interest in a digital image. In particular, it will highlight one of the most widely used approaches called the confusion matrix, that will serve for the subsequent comparative study of this dissertation and from which other metrics are derived as the percentage of success in the classification.

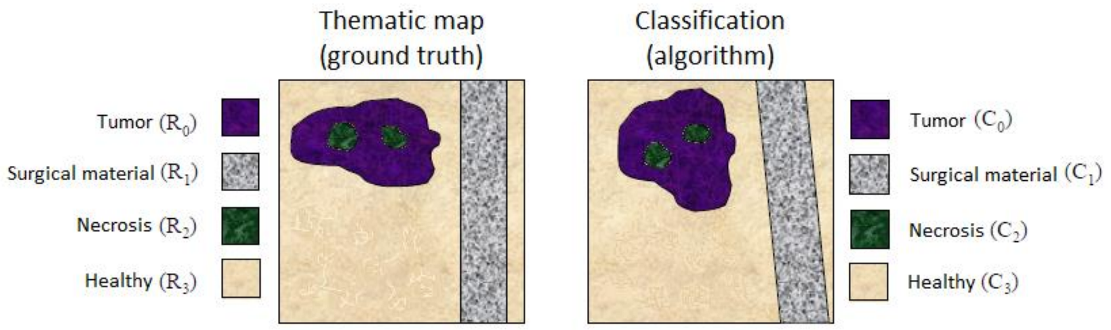
The confusion matrix [49] is a technique to assess the accuracy of classification algorithms of digital images. This technique assumes that the ground-truth information is expressed as a thematic map [50][51] characterized by the following properties:

1. Each pixel is labeled as belonging to a particular class, so as to have N classes or reference regions $\{R_i\}_{i=1}^{N}$.
2. The reference regions are mutually exclusive of each other, ie, two different regions have no pixel in common: $R_i \cap R_j = \emptyset, \forall i \neq j$.

Assuming that each pixel *i* of the image to evaluate *I* is assigned by the algorithm as belonging to a certain class $C_i$, so as to have *N* classes. $C_i$ sets are a partition of the image to be evaluated, that is, the union of all sets resulting in the image and two different sets have no element in common:

$$\bigcup_{i=1}^{N} C_i = I \ and \ C_i \cap C_j = \emptyset, \forall i \neq j \tag{2.1}$$

Given the above considerations, Figure 2.11 shows an example of the process of constructing a confusion matrix. In the figure, the thematic map of ground-truth classification associated with the image to classify, the classification result provided by a given algorithm to that image, and the confusion matrix that quantifies the accuracy of the algorithm in the classification task are shown. As shown, the matrix entries are expressed in the form of $a_{jk}$ where $a_{jk} = cardinal\{C_j \cap R_k\}$, that is, the number of pixels of the resulting region when performing the intersection between a $C_j$ class obtained by the algorithm and a ground-truth class $R_k$ [49].


Thematic map (ground truth)
Classification (algorithm)
Tumor ($R_0$)
Surgical material ($R_1$)
Necrosis ($R_2$)
Healthy ($R_3$)
Tumor ($C_0$)
Surgical material ($C_1$)
Necrosis ($C_2$)
Healthy ($C_3$)

Confusion matrix

| | $R_0$ | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|---|
| $C_0$ | $a_{00}=\|C_0 \cap R_0\|$ | $a_{01}=\|C_0 \cap R_1\|$ | $a_{02}=\|C_0 \cap R_2\|$ | $a_{03}=\|C_0 \cap R_3\|$ |
| $C_1$ | $a_{10}=\|C_1 \cap R_0\|$ | $a_{11}=\|C_1 \cap R_1\|$ | $a_{12}=\|C_1 \cap R_2\|$ | $a_{13}=\|C_1 \cap R_3\|$ |
| $C_2$ | $a_{20}=\|C_2 \cap R_0\|$ | $a_{21}=\|C_2 \cap R_1\|$ | $a_{22}=\|C_2 \cap R_2\|$ | $a_{23}=\|C_2 \cap R_3\|$ |
| $C_3$ | $a_{30}=\|C_3 \cap R_0\|$ | $a_{31}=\|C_3 \cap R_1\|$ | $a_{32}=\|C_3 \cap R_2\|$ | $a_{33}=\|C_3 \cap R_3\|$ |

*Figure 2.11: Example of constructing of a confusion matrix.*

From the confusion matrix can be derived some generic accuracy measures [52] as:

- **Overall accuracy (OA):** Percentage of pixels correctly classified in all classes:

$$OA = \frac{\sum_i^N a_{ii}}{\sum_{ij}^N a_{ij}} \times 100 \tag{2.2}$$

- **Average accuracy (AA):** Means percentage of classification accuracy per class for all classes:

$$AA = \frac{a_{ii}}{\sum_j^N aA_i} \times 100 \tag{2.3}$$

  AA can also be calculated only for a particular class instead of as the mean of all the classes. In that case it is known as the Average accuracy per class.

OA and AA are percentages should approach 100%, which would be the ideal classification. When the reference set is not well defined, the OA will not be representative regarding the true performance of the classifier. For example, if a class has very few pixels of reference, its influence on the computation of OA will be very low, while the AA will become more important, since it is an average done with the number of classes not with the total number of pixels. If the differences arising between these two measures are high, then it may indicate that there has been a poor classification for a specific class, which can affect the overall results of classification.

Moreover, there are two additional metrics really important in the medical field: sensitivity and specificity.

Let us imagine a study evaluating a new test that screens healthy and tumor samples. Each sample taking the test either has or does not have the disease. The test outcome can be positive (predicting that the sample is a tumor sample) or negative (predicting that the sample is not a tumor sample and therefore it is healthy sample). The test results for each sample may or may not match the sample's actual status. In that setting:

- True positive: Tumor sample correctly diagnosed as tumor.
- False positive: Healthy sample incorrectly identified as tumor.
- True negative: Healthy sample correctly identified as healthy.
- False negative: Tumor sample incorrectly identified as healthy.

In general, Positive = identified and negative = rejected. Therefore:

- True positive = correctly identified.
- False positive = incorrectly identified.
- True negative = correctly rejected.
- False negative = incorrectly rejected.

The four outcomes can be formulated in a confusion matrix as follows:

| | **Condition (ground-truth)** | |
|---|---|---|
| **Total population** | **Condition positive** | **Condition negative** |
| **Test outcome positive** | True positive | False positive (Type I error) |
| **Test outcome negative** | False negative (Type II error) | True negative |

*Table 2.4: Confusion matrix.*

Sensitivity and specificity are statistical measures of the performance of a binary classification test:

- **Sensitivity (true positive rate):** Relates to the test's ability to correctly detect patients who do have a condition. Consider the example of a medical test used to identify a disease. Sensitivity of the test is the proportion of samples known to have the disease, which test positive for it. Mathematically, this can be expressed as:

$$Sensitivy = \frac{True\ Positives}{True\ Positives + False\ Negatives} =$$
$$= \frac{True\ Positives}{Total\ positive\ in\ the\ population\ or\ in\ the\ Ground\ Truth}$$
$$= probability\ of\ a\ positive\ test\ given\ that\ the\ patient\ is\ ill \qquad (2.4)$$

  A test with high sensitivity has a low type II error rate.

- **Specificity (true negative rate):** Relates to the test's ability to correctly detect patients without a condition. Consider the example of a medical test for diagnosing a disease. Specificity of a test is the proportion of healthy samples known not to have the disease, which will test negative for it. Mathematically, this can also be written as:

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} =$$
$$= \frac{True\ Negatives}{Total\ negative\ in\ the\ population\ or\ in\ the\ Ground\ Truth}$$
$$= probability\ of\ a\ negative\ test\ given\ that\ the\ patient\ is\ well \qquad (2.5)$$

  A test with a high specificity has a low type I error rate.

For any test, there is usually a trade-off between the measures.

A perfect predictor would be described as 100% sensitive (e.g., all sick are identified as sick) and 100% specific (e.g., all healthy are not identified as sick); however, theoretically any predictor will possess a minimum error bound known as the Bayes error rate.

Finally, it is defined the Kappa coefficient (K), that is a statistical measure to assess the reliability of an agreement between a fixed number of evaluators to assign categorical classifications to a number of elements which are classified by these evaluators. This measure calculates the degree of agreement in the classification what would be expected by random effect, i.e., when evaluators are not absolutely sure and just venture an answer.

$$k = \frac{correct\ classification\ probability - agreement\ by\ chance\ probability}{1 - agreement\ by\ chance\ probability} \quad (2.6)$$

The kappa coefficient can be interpreted considering the following table:

| Kappa coefficient (K) | Classification can be regarded as |
|---|---|
| Below 0.4 | Poor |
| 0.41-0.60 | Moderate |
| 0.61-0.75 | Good |
| 0.76-0.80 | Excellent |
| 0.81 and above | Almost perfect |

*Table 2.5: Interpretation of Kappa coefficient.*

If evaluators are in full agreement, then $k = 1$. If no agreement among evaluators (other than what would be expected by chance) then $k = 0$ [53].

There are other metrics such as the commission errors and omission errors that are not considered in this study.

## 2.5. Summary

Hyperspectral imaging is a very powerful tool that can be used to provide a solution in many different fields. In this dissertation it is intended to apply hyperspectral imaging in the medical field, particularly in the brain cancer detection.

For this purpose, a study of main existing techniques for hyperspectral imaging preprocessing is performed because it is necessary to properly process hyperspectral data with the ultimate goal of obtaining the best possible classification. In order to evaluate the classification process, a set of standard evaluation metrics will be used to subsequently to draw conclusions with regard to the efficiency approaches for brain cancer detection.

# Chapter 3
# Image processing techniques

Image processing is the processing of images using mathematical operations in any form of signal processing for which the input is an image, such as a photograph, video frame or hyperspectral imaging; the output of image processing may be either an image or a set of characteristics or parameters related to the image [54]. Namely, image processing is the set of techniques applied to images in order to improve quality or facilitate the search for information.

The processing chain in this project can be divided into five different stages, as we can observe in Figure 3.1:

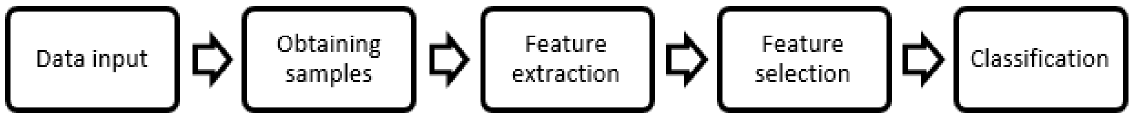


*Figure 3.1: Stages of hyperspectral data processing chain followed in this project.*

The classification system is composed by the preprocessing stage and the classification algorithms.

Firstly, the initial set of samples is studied in order to delete the anomalous samples taken during the surgery. In this stage, the hyperspectral cubes are spatially reduced in size, in order to process the region of interest.

On the one hand, in the case of *ex vivo* samples, its contour is obtained with the purpose of extracting the samples from the rest of the image (table, test plates, shadows and glow in the samples), and to prepare the data for the classification process in response to the inclination of the samples (no inclination, 23.58 degrees and 19.10 degrees) and according to the kind of filter used. For this reason, in this stage we use two types of filter: HySime and Smooth, as well as a function to obtain the contour.

On the other hand, in the case of *in vivo* samples, the images are formed by several elements, labeled directly on the image thereby forming the *in vivo* dataset. Then, these samples are processed as the *ex vivo* samples.

Features are functions of the original measurement variables that are useful for classification and/or pattern recognition [55].

The following step of the processing chain, **feature extraction**, is the process of defining a set of features, or image characteristics, which will most efficiently or meaningfully represent the information that is important for analysis and classification. In this stage there are several applicable methodologies, of which will be applied in this dissertation, mainly dimensionality reduction and unmixing techniques.

The next stage of the chain relies on the feature selection, also important in reducing the high dimensionality of the data. The fact of choosing a set of features whose dimensionality is the most appropriate and reasonable is a hot topic in the scientific literature, so efficient and fast algorithms that perform the process of combining bands or characteristics for a given problem are needed. This is a complex process that cannot be defined with a unique approach. There are several techniques of statistical indices for feature selection that will be seen in detail in the corresponding section.

Finally, once the data have been reduced and/or processed, and with some analyzes already performed, they will be classified.

As noted in the previous chapter, classification is generally a process in which the individual elements or items are differentiated into groups, based on quantitative information of one or more inherent characteristics of the elements, usually by items previously labeled using sets or training patterns. Finally, and as an option, the resulting data from the classification may be post-processed (for example using spatial techniques) to improve the coherence of them.

## 3.1. Pre-processing techniques for hyperspectral images

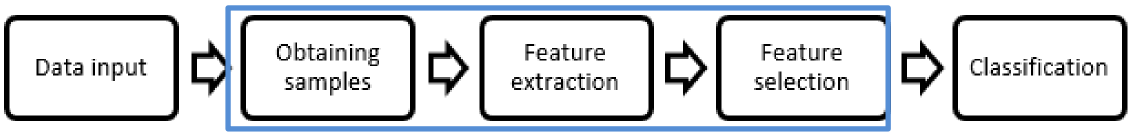


*Figure 3.2: Stages of hyperspectral data processing chain.*

This section describes the preprocessing methods discussed previously. These methods are used in the present dissertation as building blocks of processing chains considered.

### 3.1.1. De-noising filtering

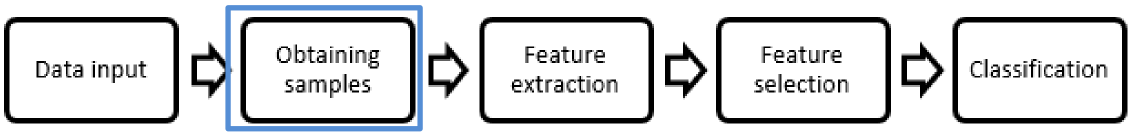


*Figure 3.3: Stages of hyperspectral data processing chain.*

Noise reduction is the process of removing noise from a signal.

All recording devices, both analog and digital, have traits which make them susceptible to noise. Noise can be random or white noise with no coherence, or coherent noise introduced by the device's mechanism or processing algorithms.

The following sections describe the filtering algorithms used in this dissertation: HySime filter and Smooth filter.

#### 3.1.1.1. HySime filter

Hyperspectral subspace identification by minimum error (HySime) is an eigen-decomposition based technique and does not depend on any tunable parameters. HySime initializes by determining the signal and noise correlation matrices and then representing the subspace by minimizing the mean square error between the signal projection and the noise projection, estimating the number of spectrally distinct signal sources in hyperspectral dataset. The result is an estimate of the number of spectrally distinct signal sources or the inherent dimensionality of the dataset [56][57][58]. This method was proposed in [59] and it is eigen-decomposition based i.e. it decomposes or reduces the original signal into subsets of eigen-vectors. The subspace obtained by HySime optimally represents the original signal with minimum error. HySime uses multiple regressions for the estimation of the noise and signal covariance matrices and is adaptive, i.e. it does not require any tuning parameters. Also it makes no assumptions about the noise being independent and identically distributed (i.i.d.) and the subspace dimensions.

The difficulty in getting reliable noise estimation from these eigenvalues is that these eigenvalues are still representing the mixtures of the signal sources and the noise present in the data. When the signal sources are too weak their contribution towards the computation of eigenvalues is very less, which can be observed if there is no sudden drop in eigenvalues distribution [60]. HySime instead finds the subset of eigenvectors and the corresponding eigenvalues by minimizing the mean square error between the original signal and the noisy projection of it.

HySime [59] starts with the noise estimation step in which the noise correlation matrix of the data is computed. Then it calculates the signal correlation matrix and computes the eigenvectors by performing the eigen-decomposition of the signal correlation matrix. The signal subspace is then derived by minimizing the sum of projection error power and noise power, which are decreasing and increasing functions of the subspace dimensions respectively.

Let us assume that the observed spectral vectors, $Y \in \boldsymbol{R}^L$, for the given hyperspectral scene are given by:

$$y = x + n \tag{3.1}$$

where $\mathrm{x}$ and $\mathrm{n}$ are L-dimensional vectors for signal and noise, and $\mathrm{L}$ is the number of bands.

The assumption here is that the signal vectors reside in an unknown p–dimensional subspace such that,

$$x = Ms \tag{3.2}$$

Where: $p < L$,

$M$ is a $L \times p$ matrix, whose columns represent the image endmembers

And: $s$ is the abundance fraction of the endmembers.

### 3.1.1.1.1 Noise Estimation

Estimation of noise from a dataset is a challenging task in image processing. HySime uses a multiple regression based approach for noise estimation from hyperspectral images, which performs better than others methods because of the high correlation exhibited between adjacent spectral bands.

Let Y be an L×N matrix, where N is the number of observed spectral vectors and L is the number of bands. Then define a matrix, $Z = Y^T$, which is an N×L matrix, a N×1 vector, $z_i = [Z]_{:,i}$, where $[Z]_{:,i}$ is the ith column of Z, i.e. $z_i$ contains the data read by the hyperspectral sensor at the ith band for all image pixels, and the N×(L-1) matrix $Z_{\partial i} = [z_1, \dots, z_{i-1}, z_{i+1}]$.

Now if $z_i$ is given by the linear regression equation,

$$z_i = Z_{\partial i}\beta_i + \xi_i \tag{3.3}$$

where, $Z_{\partial i}$ - data matrix of dimensions N×(L - 1)

$\beta_i$ - regression vector of size (L – 1)×1

$\xi_i$ - noise vector of size N×1

The least square estimate for the regression vector $\beta_i$ is given by the equation,

$$\hat{\beta}_i\text{=}(Z_{\partial i}^T Z_{\partial i})^{-1} Z_{\partial i}^T z_i \tag{3.4}$$

The noise estimates, $\hat{\xi}_i$, are given by the equation,

$$\hat{\xi}_i = z_i - Z_{\partial i}\hat{\beta}_i \tag{3.5}$$

and the estimated noise correlation matrix, $\hat{R}_n$, is given by,

$$\hat{R}_n = [\hat{\xi}_1, \dots \dots, \hat{\xi}_N]^T\, [\hat{\xi}_1, \dots \dots, \hat{\xi}_N]/N \tag{3.6}$$

### 3.1.1.1.2 Signal Subspace Identification

Signal subspace estimation starts by computing the noise and signal correlation matrices. A subset of the eigenvectors of the signal correlation matrix is used to represent the subspace. This signal subspace is determined by minimizing the mean square error between the original signal, x, and the noisy projection of it i.e. the observed spectral vector. The signal correlation matrix is given by, $\hat{R}_x$

$$\hat{R}_x = [\hat{x}_1, \dots\dots, \hat{x}_N][\hat{x}_1, \dots\dots, \hat{x}_N]^T/N \tag{3.7}$$

where, $\hat{x}$ - signal estimates obtained after subtracting the noise estimates from the original data.

The eigenvectors can be obtained by performing the eigen-decomposition of the signal correlation matrix as given by,

$$\hat{R}_x = E \sum E^T \tag{3.8}$$

Where: $E = [e_1, \dots\dots, e_L]$ is the eigenvector matrix of $\hat{R}_x$,

And $\sum$ - eigenvalues matrix of the signal correlation matrix, with the diagonal values ordered in decreasing magnitude.

Now let the space $\mathbf{R}^L$ be decomposed into two orthogonal subspaces, the k-dimensional subspace, $\langle E_k \rangle$, be represented by $E_k \equiv \left[e_{i_1}, \dots\dots, e_{i_k}\right]$ and $\langle E_k \rangle^\perp$ be the orthogonal component of subspace $\langle E_k \rangle$, spanned by $E_k^T \equiv \left[e_{i_{k+1}}, \dots\dots, e_{i_L}\right]$.

Let $U_k = E_k E_k^T$ represent the projection matrix onto the subspace $\langle E_k \rangle$, then the projection of the observed spectral vectors, y, or the noisy projection of x, onto the subspace $\langle E_k \rangle$ is given by,

$$\hat{x}_k \equiv U_k y \tag{3.9}$$

The first order moment of $\hat{x}_k$ given x is,

$$\mathbb{E}[\hat{x}_k | x] = U_k \mathbb{E}[y|x] = U_k \mathbb{E}[x + n|x] = U_k x = x_k \tag{3.10}$$

where, $x_k$ is the projection of the signal vectors onto the subspace $\langle E_k \rangle$.

And the second order moment of $\hat{x}_k$ given x is,

$$\mathbb{E}[(\hat{x}_k - x_k)(\hat{x}_k - x_k)^T|x] = \mathbb{E}[(U_k y - U_k x)(U_k y - U_k x)^T|x] = \mathbb{E}\big[\big(U_k n n^T {U_k}^T\big)|x\big]$$
$$= U_k \hat{R}_n {U_k}^T \qquad (3.11)$$

The mean square estimation between $x$ and $\hat{x}_k$ is given by,

$$mse(k|x) = \mathbb{E}[(x - \hat{x}_k)^T(x - \hat{x}_k)|x] = \mathbb{E}[(x - x_k - U_k n)^T(x - x_k - U_k n)|x]$$
$$= (x - x_k)^T(x - x_k) + \big(U_k \hat{R}_n {U_k}^T\big)^T \qquad (3.12)$$

Since, $U_k x = x_k$, implies that $x - x_k = {U_k}^{\perp}$, which is the orthogonal component of the signal projection vector onto the subspace $\langle E_k \rangle$. Thus by using the projection matrix properties i.e. $U = U^T$, $U^2 = U$ and $U^T = I - U$, we have

$$mse(k) = \mathbb{E}\left[\big({U_k}^{\perp} x\big)^T\big({U_k}^{\perp} x\big)\right] + \big(U_k \hat{R}_n {U_k}^T\big)^T = \big({U_k}^{\perp} R_x\big)^T + \big(U_k \hat{R}_n\big)^T$$
$$= \big({U_k}^{\perp} \hat{R}_y\big)^T + 2\big(U_k \hat{R}_n\big)^T + c \qquad (3.13)$$

where c is an irrelevant constant. The signal subspace $\langle E_k \rangle$ is inferred by the minimization of the mean square error given by the next equation with respect to all the permutations $\pi = \{i_1, \dots\dots, i_L\}$ and is given by the expression,

$$\big(\hat{k}, \hat{\pi}\big) = argmin\left\{\big({U_k}^{\perp} \hat{R}_y\big)^T + 2\big(U_k \hat{R}_n\big)^T\right\} \qquad (3.14)$$

where $\hat{k}$ is the estimate of the subspace and the subspace is spanned by $E_k \equiv [e_{i_1}, \dots\dots, e_{i_k}]$. Now the term $\big({U_k}^{\perp} \hat{R}_y\big)^T$ in the equation corresponds to the projection error power which is a decreasing function of subspace dimension and the term $2\big(U_k \hat{R}_n\big)^T$ corresponds to the noise power and is increasing function of subspace dimension. As mentioned above $U_k = E_k E_k^T$ is a projection matrix and from matrices properties we know that $(AB)^T = (BA)^T$, then the minimization equation can be written as

$$\big(\hat{k}, \hat{\pi}\big) = argmin\left\{c + \sum_{j=1}^{k}\big(-p_{ij} + 2\sigma_{ij}^2\big)\right\} \qquad (3.15)$$

where c is a constant and

$$p_{ij} = e_{ij}^T \hat{R}_y e_{ij} \qquad (3.16)$$

$$\sigma_{ij}^2 = e_{ij}^T \hat{R}_n e_{ij} \qquad (3.17)$$

The term on the right hand side $\left(-p_{ij} + 2\sigma_{ij}^2\right)$ is represented by $\delta_{ij}$ and by including all the negative terms of $\hat{\delta}_i$, for $i = 1, \dots \dots, L$, in the sum, the minimization of mean square error between projection power error and noise power is obtained. The estimate of the subspace dimension is given by the number of negative terms in $\hat{\delta}_i$.

The subspace dimension, $\hat{k}$, for this subset is obtained by minimizing the mean square error between the signal projection power and the noise projection. The output of the signal subspace estimation step is a set of matrices containing the eigenvalues and the eigenvectors spanning the signal subspace.

#### 3.1.1.1.3 HySime Components

The output of the HySime gives us the signal subspace estimates and the corresponding eigenvectors spanning the subspace i.e. $E_k \equiv \left[e_{i_1}, \dots \dots, e_{i_k}\right]$, sorted in descending order of their relevance. The HySime components, with the first component representing the maximum variance, corresponding to these eigenvectors can be computed by multiplying the eigenvector matrix, $E$, (can also referred to as transformation matrix or projection matrix) with the original image i.e. by projecting the original image by $E$. The components which are obtained will be ordered according to the decreasing variability. Each column of the eigenvector matrix produces a component image. The transformation can be achieved by the following expression [61],

$$newBV_{i,j,p}(Y^{HySime}) = \sum_{i=1}^{n} \left(E_{k,p} BV_{i,j,k}\right) \tag{3.18}$$

where, $E_{k,p}$ - Eigenvector matrix

$BV_{i,j,k}$ - Brightness value of pixel at ith row, jth column of the band k of original image

$newBV_{i,j,p}$ - Brightness value of pixel at ith row, jth column of the pth HySime component

$Y^{HySime}$ - HySime component image

#### 3.1.1.1.4 Inverse HySime for Hyperspectral image restoration

Once the HySime components have been obtained and the noise segregated the Hyperion data can be restored to its original spectral space without noise. The noise free original spectral space consisting of the noise less signals only can be achieved by performing an inverse HySime transform. As the original image data $Y$ was transformed into HySime components, $Y^{HySime}$ in the HySime space, the inverse transformation can be achieved by inverting the projection matrix $E$ and multiplying it with$Y^{HySime}$. The expression for restoration of signals to original spectral space is as follows,

$$Z = (E^T)^{-1} Y^{HySime} \tag{3.19}$$

where, $Y^{HySime} = \left(y_1^{HySime}, \dots\dots, y_p^{HySime}, 0, \dots\dots, 0\right)$ - is the HySime component image

*p* - the number of 1st components selected for restoring to the original spectral space and $Z$ - the restored image

### 3.1.1.2. Smooth filter

In many experiments in physical science, the true signal amplitudes (*y*-axis values) change rather smoothly as a function of the *x*-axis values, whereas many kinds of noise are seen as rapid, random changes in amplitude from point to point within the signal. In the latter situation it may be useful in some cases to attempt to reduce the noise by a process called smoothing. In smoothing, the data points of a signal are modified so that individual points that are higher than the immediately adjacent points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased, thus is achieved that the signal is more homogeneous. This naturally leads to a smoother signal. As long as the true underlying signal is actually smooth, then the true signal will not be much distorted by smoothing, but the noise will be reduced [62].

When processing a signal by the Smooth filter, the goal is to create an approximate function that attempts to capture important signal patterns, leaving out the noise. The Smooth filter smooths the signal by using a moving average filter.

In the moving average filters each output value is obtained as the average of a subset of the original data. Its purpose is to highlight the significant pattern i.e. it reduce the noise smoothing the fluctuations in short periods, thus highlighting trends or long periods cycles [62].

Moving average filters [63] are the most common filters used in signal processing, mostly because it is the easiest to understand and use digital filter. Despite its simplicity, it is optimal for the task of reducing random noise. This makes them the best filters to signals in the time domain. Instead, moving average filters are the worst for signals encoded in the frequency domain, as they have little ability to separate frequency bands from each other [64].

### 3.1.1.2.1 Smoothing algorithms

Most smoothing algorithms are based on the "*shift and multiply*" technique, in which a group of adjacent points in the original data are multiplied point-by-point by a set of numbers (coefficients) that defines the smooth shape, the products are added up to become one point of smoothed data, then the set of coefficients is shifted one point down the original data and the process is repeated. The simplest smoothing algorithm is the *rectangular* or *unweighted sliding-average smooth*; it simply replaces each point in the signal with the average of *m* adjacent points, where *m* is a positive integer called the *smooth width*. For example, for a 3-point smooth (*m* = 3):

$$S_j = \frac{Y_{j-1} + Y_j + Y_{j+1}}{3} \tag{3.20}$$

for j = 2 to n-1, where $S_j$ the jth point in the smoothed signal, $Y_j$ the jth point in the original signal, and n is the total number of points in the signal. Similar smooth operations can be constructed for any desired smooth width, *m*. Usually *m* is an odd number. If the noise in the data is "white noise" (that is, evenly distributed over all frequencies) and its standard deviation is *s*, then the standard deviation of the noise remaining in the signal after the first pass of an unweighted sliding-average smooth will be approximately *s* over the square root of *m* (*s/sqrt(m)*), where m is the smooth width.

The *triangular smooth* is like the rectangular smooth, above, except that it implements a *weighted* smoothing function. For a 5-point smooth (*m* = 5):

$$S_j = \frac{Y_{j-2} + 2Y_{j-1} + 3Y_j + 2Y_{j+1} + Y_{j+2}}{9} \tag{3.21}$$

for j = 3 to n-2, and similarly for other smooth widths.

It is often useful to apply a smoothing operation more than once, that is, to smooth an already smoothed signal, in order to build longer and more complicated smooths. For example, the 5-point triangular smooth above is equivalent to two passes of a 3-point rectangular smooth. *Three* passes of a 3-point rectangular smooth result in a 7-point "*pseudo-Gaussian*" or *haystack* smooth, for which the coefficients are in the ratio 1 3 6 7 6 3 1. The general rule is that *n* passes of a *w*-width smooth results in a combined smooth width of *n*w-n+1*. For example, 3 passes of a 17-point smooth results in a 49-point smooth. These multipass smooths are more effective at reducing high-frequency noise in the signal than a rectangular smooth.

In all these smooths, the width of the smooth *m* is chosen to be an odd integer, so that the smooth coefficients are symmetrically balanced around the central point, which is important because it preserves the x-axis position of peaks and other features in the signal. (This is especially critical for analytical and spectroscopic applications because the peak positions are often important measurement objectives).

The *Savitzky-Golay smooth* is based on the least-squares fitting of polynomials to segments of the data. Compared to the sliding-average smooths, the Savitzky-Golay smooth is less effective at reducing noise, but more effective at retaining the shape of the original signal. It is capable of differentiation as well as smoothing. The algorithm is more complex and the computational times are greater than the smooth types discussed above, but with modern computers the difference is not significant [62][65].

### 3.1.1.2.2 Noise reduction

Smoothing usually reduces the noise in a signal. If the noise is “white” (that is, evenly distributed over all frequencies) and its standard deviation is *s*, then the standard deviation of the noise remaining in the signal after one pass of a triangular smooth will be approximately *s*0.8/sqrt(m)*, where *m* is the smooth width. Smoothing operations can be applied more than once: that is, a previously-smoothed signal can be smoothed again. In some cases this can be useful if there is a great deal of high-frequency noise in the signal. However, the noise reduction for white noise is less in each successive smooth. For example, three passes of a rectangular smooth reduces white noise by a factor of approximately *s*0.7/sqrt(m)*, only a slight improvement over two passes.

The frequency distribution of noise, designated by noise color, substantially affects the ability of smoothing to reduce noise. Because smoothing is a low-pass filter process, it affects low frequency (pink) noise less, and high-frequency (blue) noise more, than white noise.

It should be clear that smoothing can never completely eliminate noise, because most noise is spread out over a wide range of frequencies, and smoothing simply reduces the noise in part of its frequency range. Only for some very specific types of noise (e.g. discrete frequency noise or single-point spikes) is there hope of anything close to complete noise elimination.

#### 3.1.1.2.3 End effects and the lost points problem

Note in the equations above that the 3-point rectangular smooth is defined only for j = 2 to n-1. There is not enough data in the signal to define a complete 3-point smooth for the first point in the signal (j = 1) or for the last point (j = n), because there are no data points before the first point or after the last point (similarly, a 5-point smooth is defined only for j = 3 to n-2, and therefore a smooth cannot be calculated for the first two points or for the last two points). In general, for an m-width smooth, there will be (*m*-1)/2 points at the beginning of the signal and (*m*-1)/2 points at the end of the signal for which a complete *m*-width smooth cannot be calculated. There are two approaches for this problem. One is to accept the loss of points and trim off those points or replace them with zeros in the smooth signal. The other approach is to use progressively smaller smooths at the ends of the signal, for example to use 2, 3, 5, 7... points smooths for signal points 1, 2, 3, and 4..., and for points n, n-1, n-2, n-3..., respectively. The later approach may be preferable if the edges of the signal contain critical information, but it increases execution time [65].

## 3.1.2. Contour detection

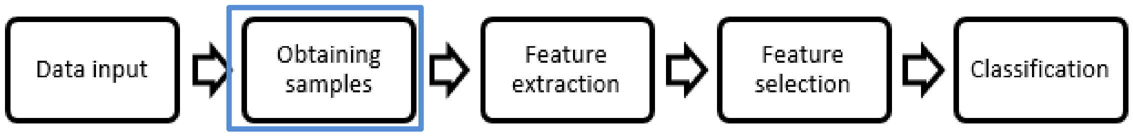


*Figure 3.4: Stages of hyperspectral data processing chain.*

This process is based on contour detection of tumor, healthy and necrosis samples. In fact, the contour of the samples is delimited by a function in order to highlight the samples ignoring the rest of the elements in the image (table, test plates and glow in the samples). This is possible due to the difference in density between different tissues and other elements in the image.

This function take a representative band which the image can be seen clearly, then it takes each pixel in the image and assigns a value of 1 or 0 depending on whether there is sample or not. To differentiate the samples from the rest of the image a contour function belonging to a MATLAB toolbox is used. This way, we obtain a binary image which has 1 in the samples and 0 in the rest of the image. This function has a condition with a decision threshold to decide if the pixel contains a sample or not. This threshold was chosen by trial and error as shown in the following images.

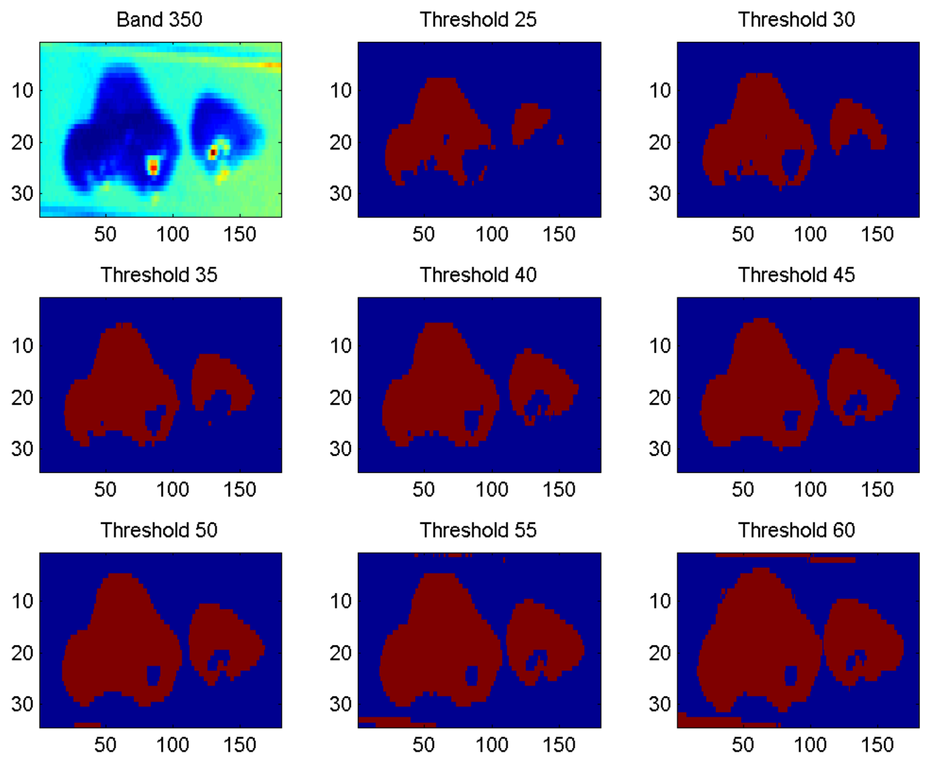
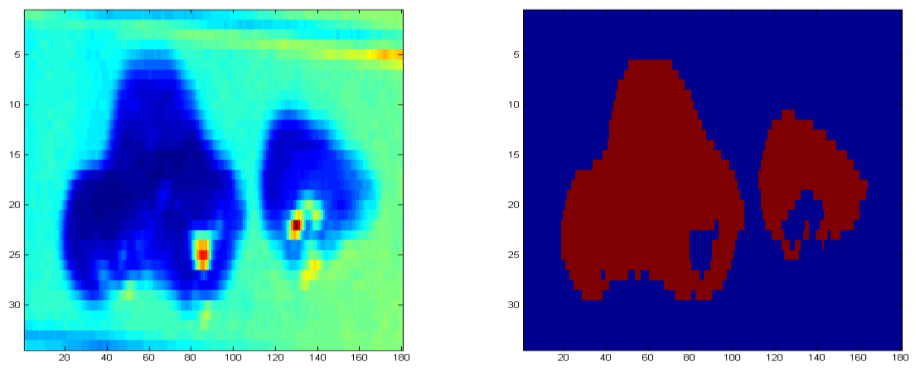


*Figure 3.5: Tumor sample 3 without inclination according to the threshold.*

In figure 3.5, the threshold is too low and we can see how some samples are lost. According as the threshold increases in order to compensate for sample loss, we can see that there are areas without samples which are marked as such.

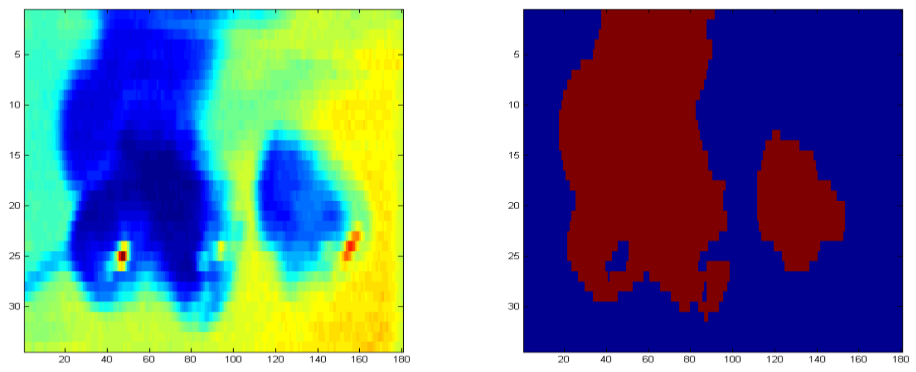
In the end, after several tests, the threshold was established in 40 because this value generally provides good average results.



(a) (b)

*Figure 3.6: Tumor 3 without inclination: (a) band 350 and (b) contour with threshold=40.*

However, this function is only valid for samples without inclination. Because in the samples with inclination, the samples generate a shadow which is confused with the sample and it produces a false samples extraction how is shown in figure 3.7.



(a) (b)

*Figure 3.7: Tumor 7 19.10 degrees of inclination: (a) band 350 and (b) contour with threshold=40.*

To solve this problem, it is necessary to show the image in other format which allow to differentiate clearly the sample from the shadow. For this purpose, firstly the image is changed to RGB format and then it is changed to HSV format. Due to these changes in the input data, it is necessary to modify the condition and the decision threshold in the contour function. This condition and its threshold were chosen by trial and error as shown in the images of Figure 3.8.

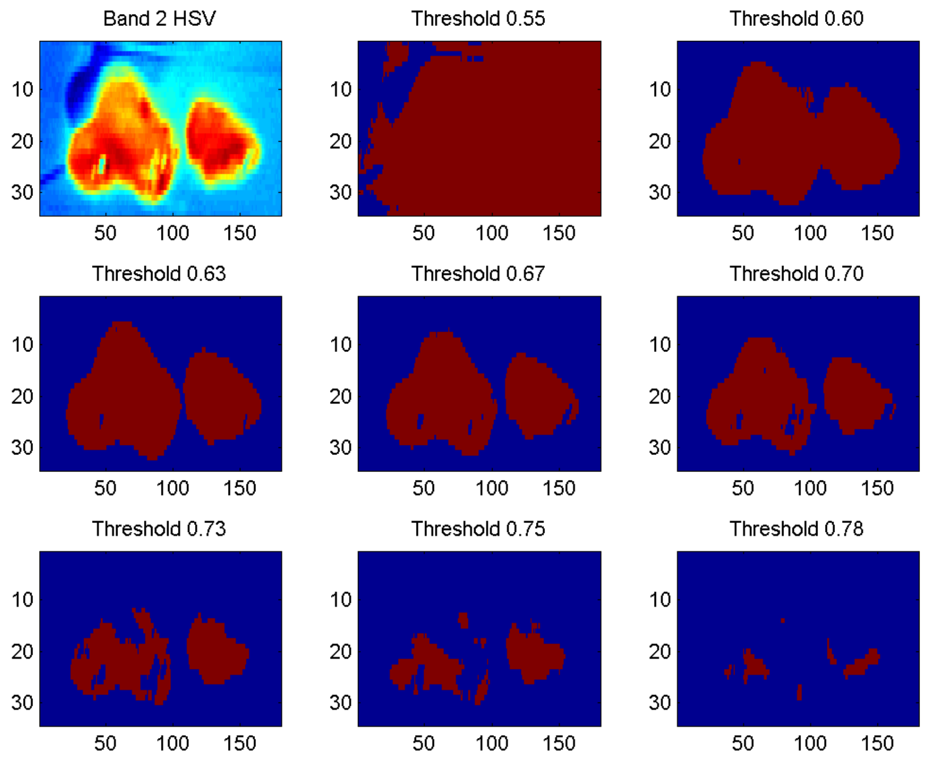
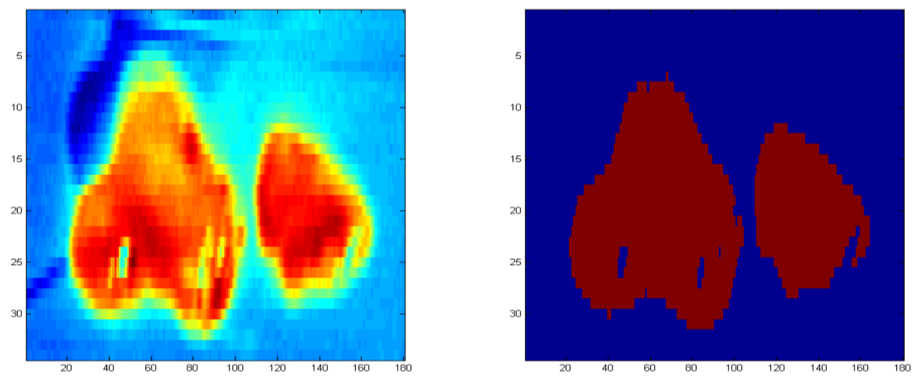


*Figure 3.8: Tumor sample 7, 19.10 degrees of inclination according to the threshold.*

In the first set of images, the threshold chosen is composed by a low value as we can see in the figure 3.8. Several studies were performed in order to find the correct value in which the experimental samples (tumor, necrosis and healthy) could be extracted. In this set of experiments, the threshold was established to 0.67.



*(a)* *(b)*

*Figure 3.9: Tumor 7 19.10 degrees of inclination: (a) band 350 HSV and (b) contour with threshold=0.67.*

Our study, conducted using three kind of inclination collected by different filters, provides several set of samples. In this regard, the nomenclatures used in this dissertation are: inclination 1, inclination 2 and inclination 3 to designate the inclinations of 0, 23.58, 19.10 degrees respectively, class 1, class 2 and class 3 to refer healthy, tumor and necrosis samples respectively, and filter 1 and filter 2 to denote HySime and smooth filter respectively.

The results of preprocessing stage are shown in the table 3.2, where we can observe the number of samples for each case:

| **Inclination** | **Filter** | **Samples** | | | |
|---|---|---|---|---|---|
| | | Class 1 | Class 2 | Class 3 | Total |
| **Inclination 1** | **No-Filter** | 2917 | 5486 | 1530 | 9933 |
| | **Filter 1** | 2971 | 5543 | 1555 | 10069 |
| | **Filter 2** | 2986 | 5258 | 1403 | 9647 |
| **Inclination 2** | **No-Filter** | 811 | 1787 | 441 | 3039 |
| | **Filter 1** | 807 | 1785 | 443 | 3035 |
| | **Filter 2** | 741 | 1789 | 423 | 2953 |
| **Inclination 3** | **No-Filter** | 802 | 3839 | 1464 | 6105 |
| | **Filter 1** | 803 | 3827 | 1469 | 6099 |
| | **Filter 2** | 742 | 3834 | 1404 | 5980 |

*Table 3.1: Samples obtained.*

Table 3.2 shows the number of samples corresponding to 3 %, 5% and 10% of the global set of samples. They are obtained by randomly selecting of the global pixels.

| Inclination | Filter | Training | | |
|---|---|---|---|---|
| | | 3% | 5% | 10% |
| **Inclination 1** | **No-Filter** | 300 | 498 | 996 |
| | **Filter 1** | 303 | 504 | 1008 |
| | **Filter 2** | 291 | 483 | 966 |
| **Inclination 2** | **No-Filter** | 93 | 153 | 306 |
| | **Filter 1** | 93 | 153 | 306 |
| | **Filter 2** | 90 | 150 | 297 |
| **Inclination 3** | **No-Filter** | 144 | 240 | 480 |
| | **Filter 1** | 183 | 306 | 612 |
| | **Filter 2** | 180 | 300 | 600 |

*Table 3.2: Trainings generated.*

In this set we have 3 possible classes: the number of samples per class for each training set is equal to the total number of samples in such training divided by 3. Moreover, each test is formed by removing the training samples from the total samples, so each test is composed by the number of samples resulting from subtracting the total number of samples for each case minus the number of samples of a particular training. This process is shown in figure 3.10.

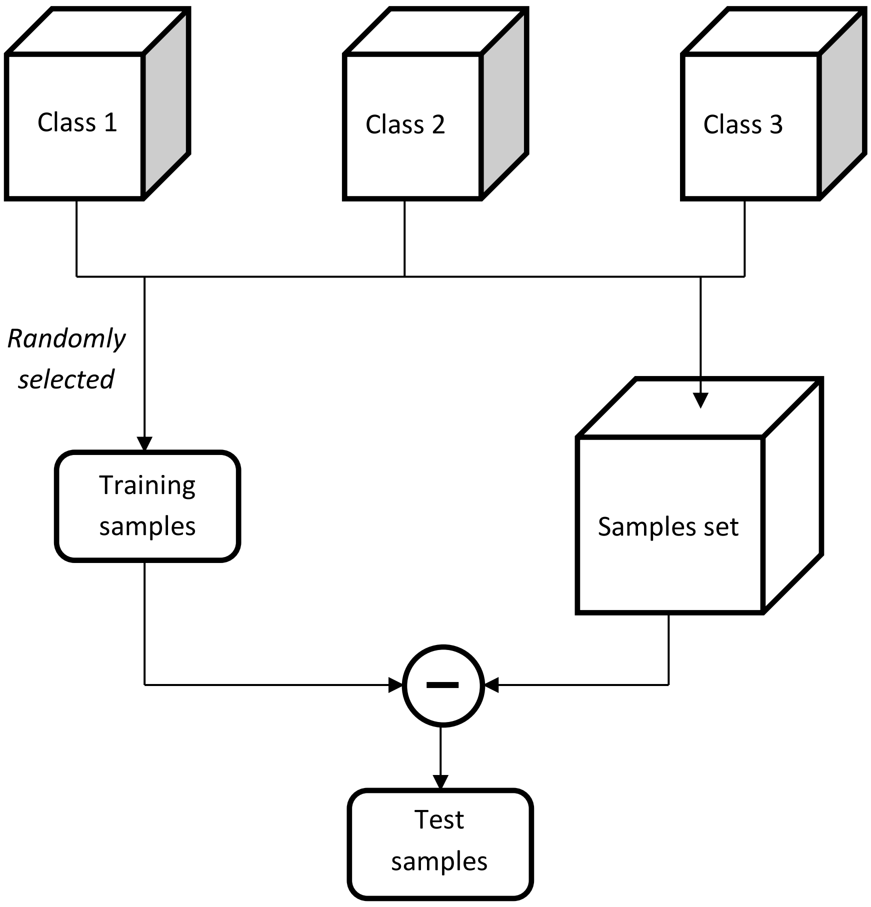


*Figure 3.10: Representation of the generation of samples set, training samples and test samples.*

In order to study all of the cases, the data with different angles are mixed together. In this case we remove the dependence over the angles.

| Filter | Samples |
|---|---|
| **No-Filter** | 19077 |
| **Filter 1** | 19203 |
| **Filter 2** | 18580 |

*Table 3.3: Total samples.*

Currently, the resulting trainings for these datasets are obtained.

| **Filter** | **Training** | | |
|---|---|---|---|
| | 3% | 5% | 10% |
| **No-Filter** | 573 | 954 | 1908 |
| **Filter 1** | 579 | 963 | 1923 |
| **Filter 2** | 558 | 930 | 1860 |

*Table 3.4: Training sets.*

Therefore, as a result of this block we obtain several set of samples and their corresponding training.

## 3.1.3. Feature extraction techniques

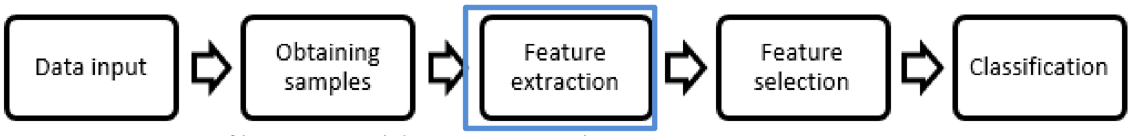


*Figure 3.11: Stages of hyperspectral data processing chain.*

Feature extraction transforms the data in a high-dimensional space to a space of fewer dimensions. It starts from an initial set of measured data and builds derived values (features) intended to be informative, non-redundant, facilitating the subsequent learning and generalization steps, in some cases leading to better human interpretations. The data transformation may be linear or nonlinear [66][67].

This process happens when the input data to an algorithm are too large to be processed and it is suspected to be redundant (e.g. the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features. The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

Feature extraction involves reducing the amount of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which overfits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

The spectral transformations data acting on the vectors to get new sets or components bands in the image are processing techniques for feature extraction. These new components represent an alternative description of the data, in which a pixel vector is related to its previous brightness value of the original image by a linear transformation of the spectral bands. These techniques seek to preserve the essential information of the original image by reducing the number of transformed dimensions and are used before the classification process in order to increase accuracy.

Regarding the hyperspectral image processing, reducing the information is very important. The best results are achieved when an expert constructs a set of application-dependent features. Nevertheless, if no such expert knowledge is available, general dimensionality reduction techniques may help. Thus in literature have been investigated various methods to solve the problem of the original repetitive information and perform more efficient characterization. In this report are included: Principal Component Analysis (PCA), Independent Component Analysis (ICA), Minimum Noise Fraction (MNF), Locality Preserving Projections (LPP) and Neighborhood Preserving Embedding (NPE) as a linear methods and Kernel Principal Component Analysis (KPCA) as a nonlinear method; which are known as 'reduction methods'.

Other applicable methodologies for feature extraction are endmembers extraction and unmixing algorithm, which there are investigations for hyperspectral image analysis. The hyperspectral unmixing is a source separation problem (scene materials) which is statistically dependent and must be combined in a nonlinear function. There are different strategies (spectral versus hybrid techniques) being compared looking for an efficient solution taking into account the high dimensionality of the data.

There are also other methods for feature extraction such as spatial contexts, which takes into account the neighborhood or space environment of the pixel considered, because they contain much more information than the pixel itself.

The fact of using pre-processing techniques of hyperspectral imaging aimed at reducing the dimensionality of the input data is given, among other reasons, by the known as Hughes phenomenon [68], described below. In a typical classification problem, the goal is to assign a class label to the input data. The minimum expected error that can be achieved by performing classification is known as the Bayes error [69]. The Bayes error is a function which decreases with the data dimensionality. A new feature adds information about the instance and then, one would expect that the classification would be as good as when this information had not been entered. However, in practice this is not thus, when adding a new feature to the data the Bayes error is decreased, but also the deviations of classification error increases. This increase is due to the fact that more parameters need be calculated based on the same number of examples. If the increase of the deviations in the classification error is greater than the decrease Bayes error, then the use of the additional characteristic degrades the decision rule. And this phenomenon is what is known as the Hughes effect [68]. Furthermore, when the data dimensionality and the complexity of the decision rule increase, the Hughes effect may become more serious [7].

In summary, the supervised classifier performance decreases with the data dimensionality unless the number of samples is infinite [68]. This dimensional reduction that arises is a step used to reduce the computational load of successive steps by removing noise and redundant information in the image. In spectral data, much of the information is repeated from image to image. This redundancy complicates analysis and classification unnecessarily. These methods perform a decrease in the number of bands. The goal is to obtain a minimum representation of the image that contains the necessary information to perform the analysis on a small subset of the original image [70]. Moreover, dimensional reduction techniques usually bring as a result an improvement in the SNR of data through the removing noise [71], which makes it attractive to use previous to classification step. The disadvantage of this alternative is the difficulty in interpreting the spectral data after the reduction step.

### 3.1.3.1. Dimensionality reduction techniques

In machine learning, dimensionality reduction is the process of reducing the number of random variables under consideration, [72] and can be divided into feature extraction and feature selection [73].

It is important to distinguish the dimension reduction techniques from compression techniques for hyperspectral imaging [74]. Contrary to the compression methods objective, the dimensional simplification process usually does not allow reconstructing the original image. By contrast, dimensional reduction aim is to obtain a minimum representation of the image that contains the necessary information to perform the analysis on a reduced subset of original image. Thus, dimensional reduction algorithms are usually designed so that minimizes errors when working with this subset, unconcerned about the possibility of recovering the original image [75].

#### 3.1.3.1.1. Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) method takes advantage of the high correlation between consecutive bands of a hyperspectral image [76]. The PCA transformation allows to obtain a reduced set of bands (called eigenvectors) poorly correlated to each other (orthogonal, in the ideal case) containing most of the information present in the original image. Thus the first eigenvector contains the highest percentage of the variance of the original image; the second contains higher percentage of variance than the third, and so on.

The last bands of decomposition usually come characterized by a low content in terms of relevant information and these are mostly composed of noise present in the original image. In this way, the PCA transformation allows to separate noise of useful information [77]. Importantly, the set of bands resulting of PCA transformation is obtained from linear combinations of the original image bands. The procedure used is based on the identification of a new system of orthogonal axes on which data are projected. These axes have their origin in the data average vector, and these are rotated successively in order to maximize the variance. The axes are identified from the decomposition of the covariance matrix of the image, $\mathit{\Gamma}$, according to the expression shown below:

$$\mathit{\Gamma} = \frac{1}{P}\sum_{i=1}^{P}(\boldsymbol{u}_i - \boldsymbol{\mu})(\boldsymbol{u}_i - \boldsymbol{\mu})^T, \tag{3.22}$$

where $\boldsymbol{u}_i$ reference image pixels, $\boldsymbol{\mu}$ is the mean vector of all the pixels, and $P$ is the number of the image pixels. The result of projecting the pixels of the hyperspectral image over the new axes obtained is a new hyperspectral image that is formed by $b_i, i = 1 \ldots N$ bands (where $N$ is the number of bands of the original image). These bands, also called principal components, can be obtained as projections of an eigenvectors set which indicate weighting applied to each of the original bands. The number of principal components is less than or equal to the number of original variables. In addition, a set of $\lambda_i, i = 1 \ldots N$ eigenvalues (scalars) associated is obtained, whose magnitude indicates the amount of information contained in corresponding eigenvector data [78]. Thereby, the covariance matrix can be expressed as follows:

$$\Gamma = V \Sigma V^T, \tag{3.23}$$

where $V$ is the unitary matrix of eigenvectors and $\Sigma$ is the diagonal matrix of eigenvalues of $\Gamma$. The PCA transformation is illustrated graphically in Figure 3.12. As shown in the figure, this transformation enables a new coordinate system on which data are projected.

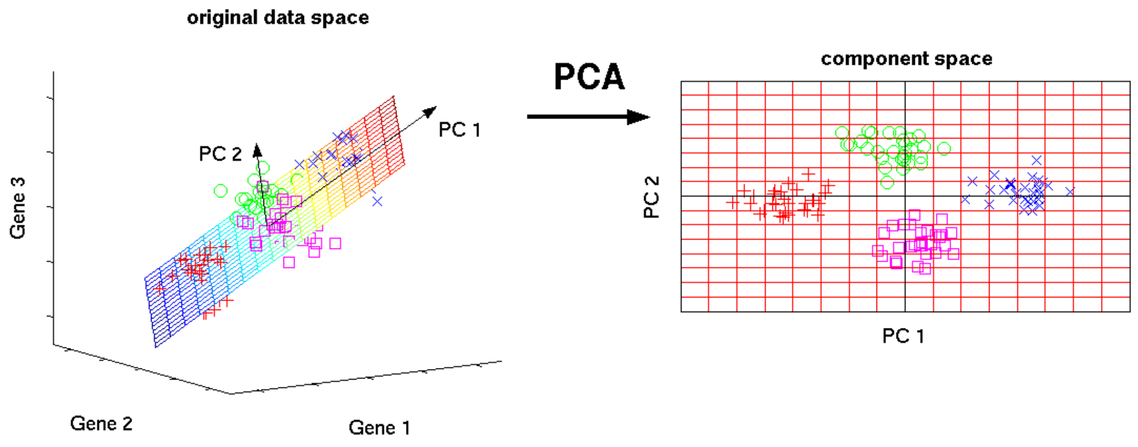


*Figure 3.12: Graphic illustration of the PCA transformation.*

An example of the application of the PCA transformation to real hyperspectral image is shown in Figure 3.13 [5]. The figure shows the first 20 bands obtained from the PCA transform. Visually, it can be seen that the presence of noise is much lower in the first bands, increasing considerably in the last bands.

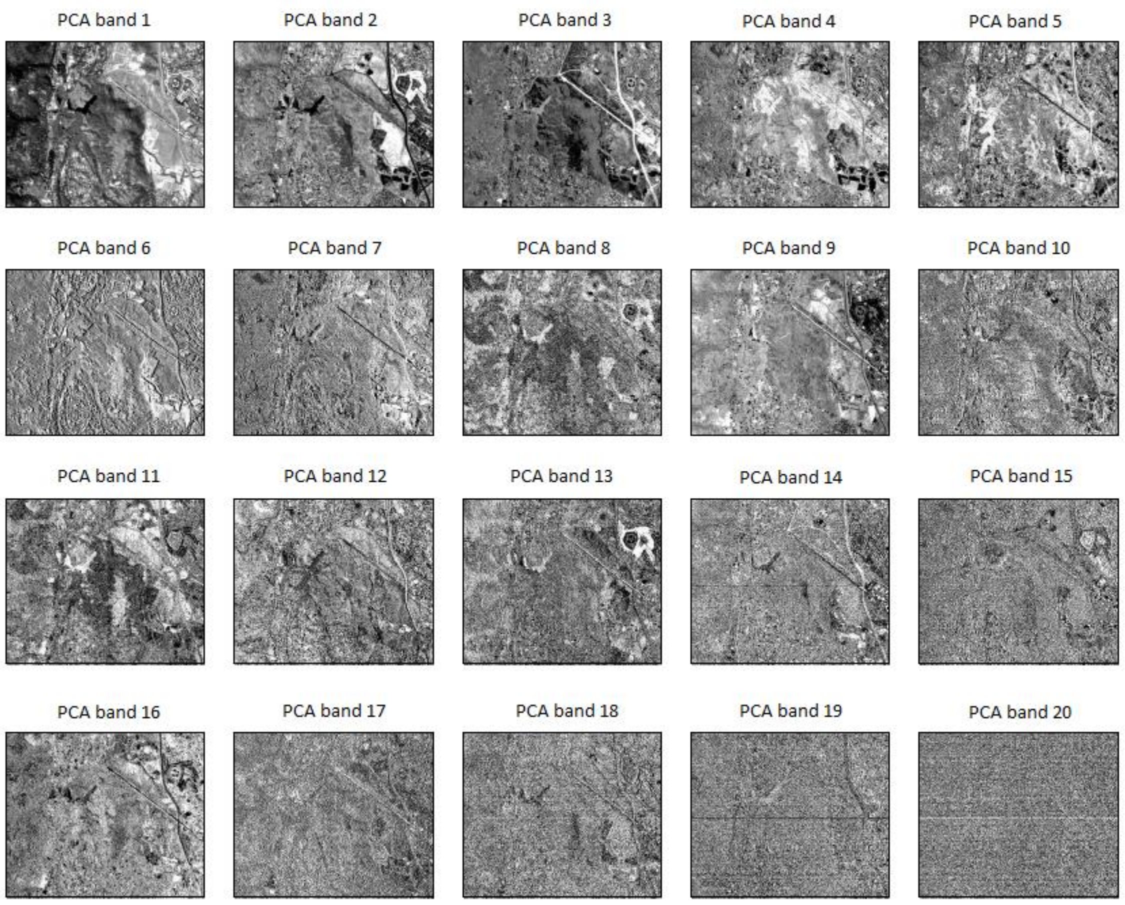


*Figure 3.13: Application example of the PCA transform on a hyperspectral image.*

Principal component analysis can be employed in a nonlinear way by means of the kernel function. The resulting technique is capable of constructing nonlinear mappings that maximize the variance in the data. The resulting technique is entitled kernel PCA, which will be explained in detail in the corresponding section.

### 3.1.3.1.2. Minimum Noise Fraction (MNF)

The Minimum Noise Fraction (MNF) transformation is a dimensional reduction method for hyperspectral image consisting in performing the steps described below [79].

1. First, PCA transformation is applied on the original image, through which the signal is separated from the noise, which is isolated in the last bands.
2. Then, the signal covariance matrix, $\Gamma_S$, and the noise covariance matrix, $\Gamma_R$, are estimated using the following expression.

$$\frac{\boldsymbol{v}^T \Gamma_S \boldsymbol{v}}{\boldsymbol{v}^T \Gamma_R \boldsymbol{v}} \tag{3.24}$$

3. Next, a set of components containing weighted information about the variance in the original data set is obtained. For this is used a MNF index that estimates the ratio between signal and noise present in the components provided by the PCA transform. The component having the minimum noise fraction is that whose associated eigenvector, $\boldsymbol{v}$, maximizes the expression (3.24).

The main difference between the PCA transformation and MNF transformation is the fact that in the second case a more detailed description of the relationship between the amount of signal and the amount of noise present in the image is performed [80]. Thus, the first band resulting from the MNF transformation is the one with highest SNR ratio. The second band has a better SNR than the third, and so on.

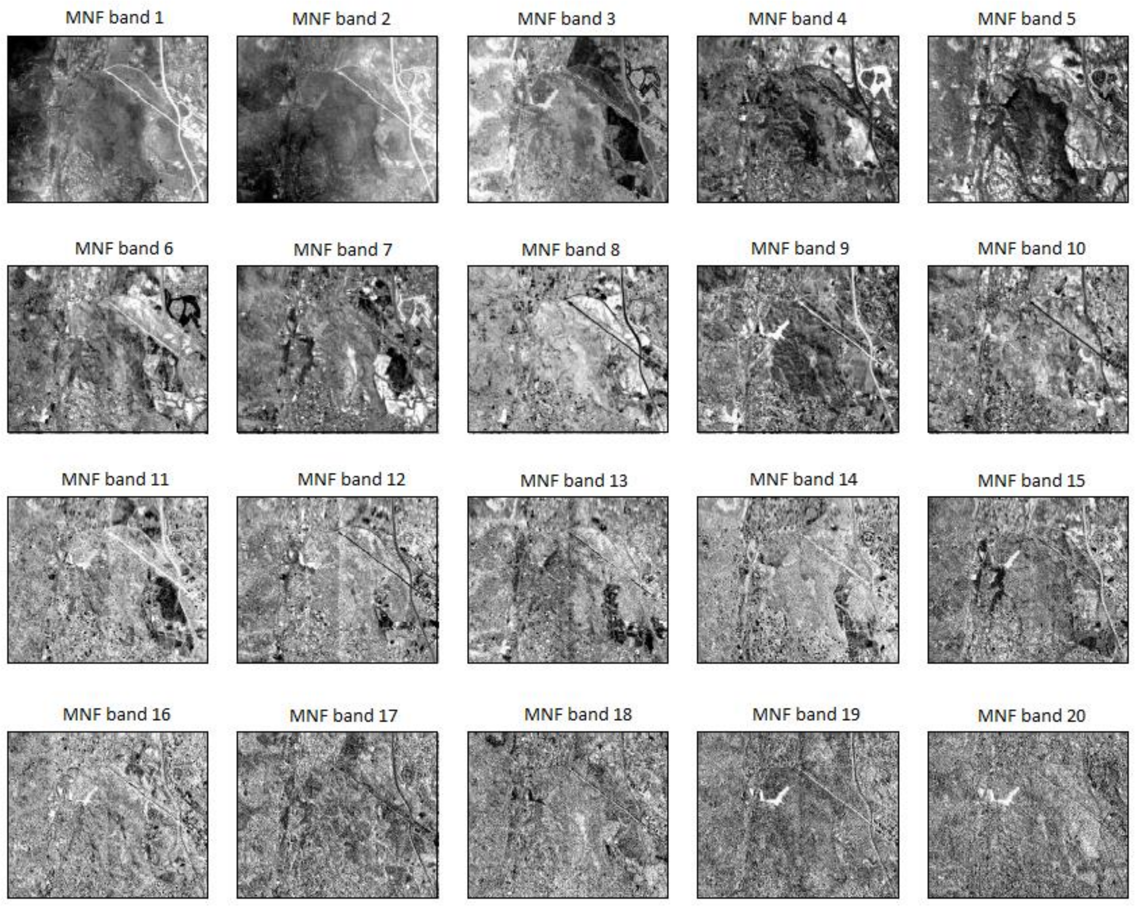


*Figure 3.14: Application example of the MNF transform on a hyperspectral image.*

As a consequence of the more accurate estimation of the noise conditions present in the image, in certain applications the MNF decomposition may provide more robust results than the PCA transform [81], because it is less sensitive to outliers and noisy pixels. The MNF transform performs a translation of the data, so that the coordinate origin is the centroid of the resulting point cloud. Sometimes, this feature allows obtaining a better description of the data. By comparison, Figure 3.14 [5] shows the first 20 bands obtained from the application of the MNF transform on the real hyperspectral image previously used.

#### 3.1.3.1.3. Independent Component Analysis (ICA)

In signal processing, Independent Component Analysis (ICA) is a statistical and computational method for separating a multivariate signal into additive subcomponents, revealing hidden factors that underlie sets of random variables, measurements, or signals. This is done by assuming that the subcomponents are non-Gaussian signals and that they are statistically independent from each other.

Independent Component Analysis attempts to decompose a multivariate signal into independent non-Gaussian signals. As an example, sound is usually a signal that is composed of the numerical addition, at each time t, of signals from several sources. The question then is whether it is possible to separate these contributing sources from the observed total signal. When the statistical independence assumption is correct, blind ICA separation of a mixed signal gives very good results. It is also used for signals that are not supposed to be generated by a mixing for analysis purposes. An important note to consider is that if N sources are present, at least N observations are needed to recover the original signals. This constitutes the square case (J = D, where D is the input dimension of the data and J is the dimension of the model). Other cases of underdetermined (J > D) and overdetermined (J < D) have been investigated.

The fact that the ICA separation of mixed signals provides very good results are based on two assumptions and three effects of mixing source signals. The two assumptions are:

1. The source signals are independent of each other.
2. The values in each source signal have non-Gaussian distributions.

The three effects of mixing source signals are:

1. **Independence:** As per assumption 1, the source signals are independent; however, their signal mixtures are not. This is because the signal mixtures share the same source signals.

2. **Normality:** According to the Central Limit Theorem, the distribution of a sum of independent random variables with finite variance tends towards a Gaussian distribution. Loosely speaking, a sum of two independent random variables usually has a distribution that is closer to Gaussian than any of the two original variables. Here we consider the value of each signal as the random variable.
3. **Complexity:** The temporal complexity of any signal mixture is greater than that of its simplest constituent source signal.

Those principles contribute to the basic establishment of ICA. If the signals we happen to extract from a set of mixtures are independent like sources signals, or have non-Gaussian histograms like source signals, or have low complexity like source signals, then they must be source signals. [82][83]

ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed to be non-Gaussian and mutually independent and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA.

ICA finds the independent components (also called factors, latent variables or sources) by maximizing the statistical independence of the estimated components. We may choose one of many ways to define independence, and this choice governs the form of the ICA algorithm. The two broadest definitions of independence for ICA are:

1. Minimization of mutual information (uses measures like Kullback-Leibler Divergence and maximum entropy).
2. Maximization of non-Gaussianity (motivated by the central limit theorem, uses kurtosis and negentropy).

Typical algorithms for ICA use centering (subtract the mean to create a zero mean signal), whitening (usually with the eigenvalue decomposition), and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening and dimension reduction can be achieved with Principal Component Analysis (PCA). Whitening ensures that all dimensions are treated equally a priori before the algorithm is run. Well-known algorithms for ICA include infomax, FastICA, and JADE, but there are many others. In this case, ICA JADE was used for this project.

Linear independent component analysis can be divided into noiseless and noisy cases, where noiseless ICA is a special case of noisy ICA. The data are represented by the random vector:

$$x = (x_1, \dots, x_m)^T \tag{3.25}$$

and the components as the random vector:

$$s = (s_1, \dots, s_n)^T \tag{3.26}$$

The task is to transform the observed data $x$, using a linear static transformation $W$ as:

$$s = Wx \tag{3.27}$$

into maximally independent components $s$ measured by some function:

$$F(s_1, \dots, s_n) \tag{3.28}$$

of independence.

In the linear noiseless ICA, the components $x_i$ of the observed random vector:

$$x = (x_1, \dots, x_m)^T \tag{3.29}$$

are generated as a sum of the independent components $s_k, k = 1, \dots, n$:

$$x_i = a_{i,1}s_1 + \dots + a_{i,k}s_k + \dots + a_{i,n}s_n \tag{3.30}$$

weighted by the mixing weights $a_{i,k}$.

The same generative model can be written in vectorial form as:

$$x = \sum_{k=1}^{n} s_k a_k \tag{3.31}$$

where the observed random vector $x$ is represented by the basis vectors:

$$a_k = (a_{1,k}, \dots, a_{m,k})^T \tag{3.32}$$

The basis vectors $a_k$ form the columns of the mixing matrix:

$$A = (a_1, \dots, a_n) \tag{3.33}$$

and the generative formula can be written as:

$$x = As \tag{3.34}$$

where

$$s = (s_1, \dots, s_n)^T \tag{3.35}$$

Given the model and realizations (samples) $x_1, \dots, x_N$ of the random vector $x$, the task is to estimate both the mixing matrix $A$ and the sources $s$. This is done by adaptively calculating the $w$ vectors and setting up a cost function which either maximizes the nongaussianity of the calculated:

$$s_k = (w^T * x) \tag{3.36}$$

or minimizes the mutual information. In some cases, a priori knowledge of the probability distributions of the sources can be used in the cost function.

The original sources $s$ can be recovered by multiplying the observed signals $x$ with the inverse of the mixing matrix:

$$W = A^{-1} \tag{3.37}$$

also known as the unmixing matrix. Here it is assumed that the mixing matrix is square $(n = m)$. If the number of basis vectors is greater than the dimensionality of the observed vectors, $n > m$, the task is overcomplete but is still solvable with the pseudo inverse.

In the case of linear noisy ICA with the added assumption of zero-mean and uncorrelated Gaussian noise:

$$n \sim N\big(0, diag(\Sigma)\big) \tag{3.38}$$

the ICA model takes the form:

$$x = As + n \tag{3.39}$$

The ICA transformation is illustrated graphically in Figure 3.15 [84].

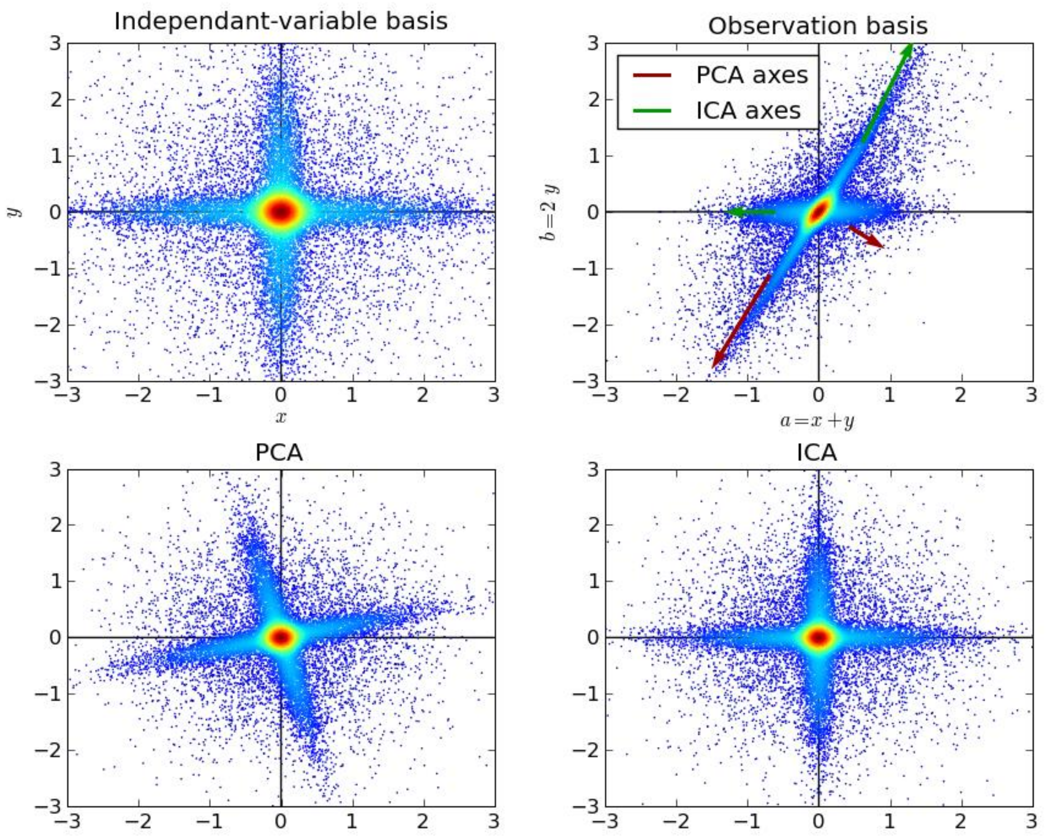


*Figure 3.15: Graphic illustration of the ICA transformation.*

An example of the application of the ICA transformation to real hyperspectral image is shown in Figure 3.16.

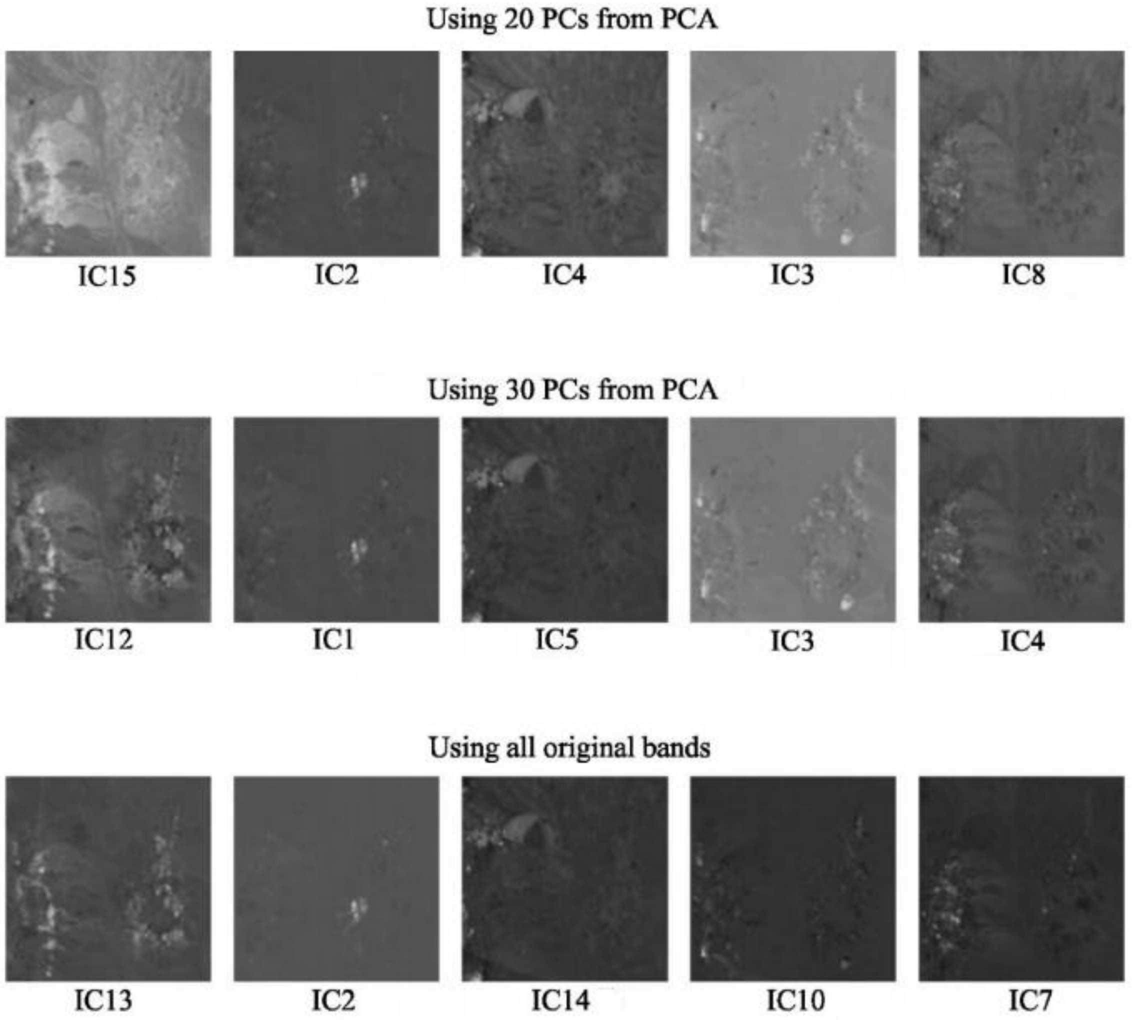


*Figure 3.16: Application example of the ICA transform on a hyperspectral image.*

ICA is somehow related to Principal Component Analysis (PCA). What distinguishes ICA from other methods is that it looks for components that are both statistically independent, and non-Gaussian. ICA is a much more powerful technique, however, capable of finding the underlying factors or sources when these classic methods fail completely [85].

#### 3.1.3.1.4. Locality Preserving Projections (LPP)

Locality Preserving Projections (LPP) are linear projective maps that arise by solving a variational problem that optimally preserves the neighborhood structure of the data set. LPP should be seen as an alternative to classical linear technique based on Principal Component Analysis (PCA). When the high dimensional data lies on a low dimensional manifold embedded in the ambient space, the Locality Preserving Projections are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. As a result, LPP shares many of the data representation properties of nonlinear techniques such as Laplacian Eigenmaps or Locally Linear Embedding. Yet LPP is linear and more crucially is defined everywhere in ambient space rather than just on the training data points [86].

Suppose we have a collection of data points of *n*-dimensional real vectors drawn from an unknown probability distribution. In increasingly many cases of interest in machine learning and data mining, one is confronted with the situation where *n* is *very large*. However, there might be reason to suspect that the "intrinsic dimensionality" of the data is much lower. This leads one to consider methods of dimensionality reduction that allow one to represent the data in a lower dimensional space.

LPP is a linear dimensionality reduction algorithm which builds a graph incorporating neighborhood information of the data set. Using the notion of the Laplacian of the graph, it computes a transformation matrix which maps the data points to a subspace. This linear transformation optimally preserves local neighborhood information in a certain sense. The representation map generated by the algorithm may be viewed as a linear discrete approximation to a continuous map that naturally arises from the geometry of the manifold [87].

The generic problem of linear dimensionality reduction is the following. Given a set $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m\ in\ \boldsymbol{R}^n$, find a transformation matrix *A* that maps these *m* points to a set of points $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_m\ in\ \boldsymbol{R}^l\ (l \ll n)$, such that $\boldsymbol{y}_i$ "represents" $\boldsymbol{x}_i$, where $\boldsymbol{y}_i = A^T \boldsymbol{x}_i$. This method is of particular applicability in the special case where $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m \in M$ and $M$ is a nonlinear manifold embedded in $\boldsymbol{R}^n$.

LPP is a linear approximation of the nonlinear Laplacian Eigenmap. The algorithmic procedure is formally stated below:

1. Constructing the adjacency graph: Let *G* denote a graph with *m* nodes. An edge is put between nodes *i* and *j* if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are "close". There are two variations:
    a. $\epsilon$-neighborhoods. [Parameter $\epsilon \in \boldsymbol{R}$]. Nodes *i* and *j* are connected by an edge if $\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 < \epsilon$ where the norm is the usual Euclidean norm in $\boldsymbol{R}^n$.
    b. $k$ nearest neighbors. [Parameter $k \in \boldsymbol{N}$]. Nodes *i* and *j* are connected by an edge if *i* is among *k* nearest neighbors of *j* or *j* is among *k* nearest neighbors of *i*.

2. Choosing the weights: Here, as well, we have two variations for weighting the edges. $W$ is a sparse symmetric $m \times m$ matrix with $W_{ij}$ having the weight of the edge joining vertices *i* and *j*, and 0 if there is no such edge.
    a. Heat kernel. [Parameter $t \in \boldsymbol{R}$]. If nodes *i* and *j* are connected, put
    $$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}} \tag{3.40}$$
    The justification for this choice of weights can be traced back to [87].
    b. Simple-minded. [No parameter]. $W_{ij} = 1$ if and only if vertices *i* and *j* are connected by an edge.
3. Eigenmaps: Compute the eigenvectors and eigenvalues for the generalized eigenvector problem:
    $$XLX^T\boldsymbol{a} = \lambda XDX^T\boldsymbol{a} \tag{3.41}$$
    where *D* is a diagonal matrix whose entries are column (or row, since *W* is symmetric) sums of *W*, $D_{ii} = \Sigma_j W_{ji}. L = D - W$ is the Laplacian matrix. The $i^{th}$ column of matrix *X* is $\boldsymbol{x}_i$.
    Let the column vectors $\boldsymbol{a}_0, \dots, \boldsymbol{a}_{l-1}$ be the solution of equation (3.41), ordered according to their eigenvalues, $\lambda_0 < \cdots < \lambda_{l-1}$. Thus, the embedding is as follows:
    $$\boldsymbol{x}_i \rightarrow \boldsymbol{y}_i = A^T\boldsymbol{x}_i, A = (\boldsymbol{a}_0, \boldsymbol{a}_1, \dots, \boldsymbol{a}_{l-1}) \tag{3.42}$$
    where $\boldsymbol{y}_i$ is a $l$-dimensional vector, and *A* is a $n \times l$ matrix.

An experiment was conducted with the *Multiple Features Database* [88]. This dataset consists of features of handwritten numbers ('0'-'9') extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. These data points are mapped to a 2-dimensional space using different dimensionality reduction algorithms, PCA, LPP, and Laplacian Eigenmaps. The experimental results are shown in Figure 3.17 [86].

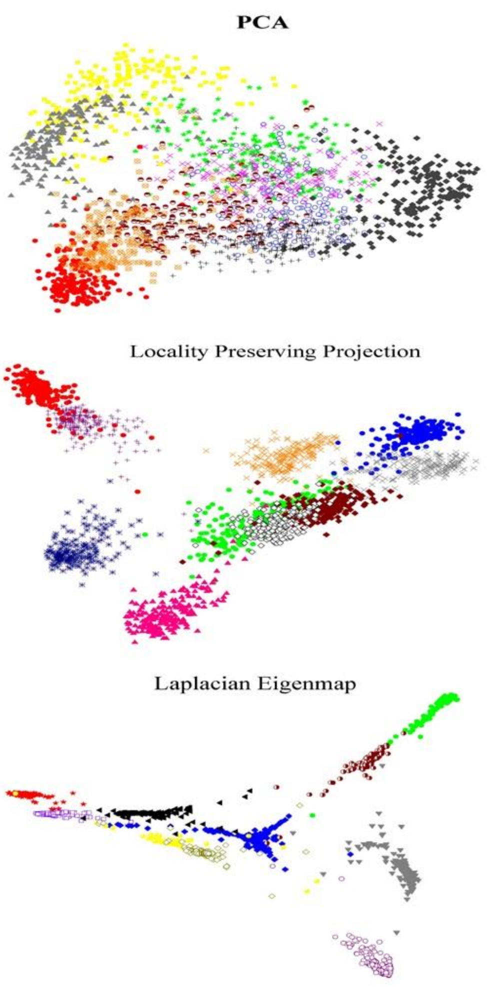


*Figure 3.17: The handwritten digits ('0'-'9') are mapped into a 2-dimensional space. Each color corresponds to a digit.*

LPPs are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. As a result, LPP shares many of the data representation properties of nonlinear techniques such as Laplacian Eigenmap. However, LPP is computationally much more tractable. Moreover, LPP is derived by preserving local information; hence it is less sensitive to outliers than PCA and it can has more discriminating power than PCA.

### 3.1.3.1.5. Neighborhood Preserving Embedding (NPE)

Neighborhood Preserving Embedding (NPE) is a subspace learning algorithm different from Principal Component Analysis (PCA) which aims at preserving the global Euclidean structure, NPE aims at preserving the local neighborhood structure on the data manifold. Therefore, NPE is less sensitive to outliers than PCA. Also, NPE is defined everywhere, rather than only on the training data points. Furthermore, NPE may be conducted in the original space or in the reproducing kernel Hilbert space into which data points are mapped [89].

NPE is a linear dimensionality reduction algorithm. This makes it fast and suitable for practical applications. Moreover, NPE can be performed in either supervised or unsupervised mode. Otherwise, NPE shares some similar properties with the Locality Preserving Projection (LPP) algorithm [86]. However, their objective functions are totally different.

Given a set of data points in the ambient space, we first build a weight matrix which describes the relationship between the data points. Specifically, for each data point, it is represented as a linear combination of the neighboring data points and the combination coefficients are specified in the weight matrix. We then find an optimal embedding such that the neighborhood structure can be preserved in the dimensionality reduced space.

The generic problem of linear dimensionality reduction is the following. Given a set $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m \ in\ \boldsymbol{R}^n$, find a transformation matrix $A$ that maps these $m$ points to a set of points $\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_m \ in\ \boldsymbol{R}^d\ (d \ll n)$, such that $\boldsymbol{y}_i$ "represents" $\boldsymbol{x}_i$, where $\boldsymbol{y}_i = A^T\boldsymbol{x}_i$. This method is of particular applicability in the special case where $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m \in M$ and $M$ is a nonlinear manifold embedded in $\boldsymbol{R}^n$.

NPE is a linear approximation to the Locally Linear Embedding (LLE) [90] algorithm. The algorithmic procedure is formally stated below:

1. Constructing the adjacency graph: Let *G* denote a graph with *m* nodes. The *i*-th node corresponds to the data point $\boldsymbol{x}_i$. There are two ways to construct the adjacency graph:
    a. *K nearest neighbors (KNN)*: Put a *directed* edge from node *i* to *j* if $\boldsymbol{x}_i$ is among the *K* nearest neighbors of $\boldsymbol{x}_i$.
    b. *$\epsilon$ neighborhoods*: Put an edge between nodes *i* and *j* if $\left\|\boldsymbol{x}_j - \boldsymbol{x}_i\right\| \le \epsilon$.

    The graph constructed by the first method is a directed graph, while the one constructed by the second method is an undirected graph. In many real world applications, it is difficult to choose a good $\epsilon$.

2. Computing the weights: In this step, the weights on the edges are computed. Let $W$ denote the weight matrix with $W_{ij}$ having the weight of the edge from node *i* to node *j*, and 0 if there is no such edge. The weights on the edges can be computed by minimizing the following objective function,

$$min \sum_i \left\| \boldsymbol{x}_i - \sum_j W_{ij} \boldsymbol{x}_j \right\|^2 \tag{3.43}$$

with constraints,

$$\sum_j W_{ij} = 1, j = 1, 2, \dots, m \tag{3.44}$$

Please see [90] for the details about how to solve the above minimization problem.

3. Computing the Projections: In this step, we compute the linear projections. Solve the following generalized eigenvector problem:

$$XMX^T\boldsymbol{a} = \lambda XX^T\boldsymbol{a} \tag{3.45}$$

where

$$X = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \tag{3.46}$$

$$M = (I - W)^T(I - W) \tag{3.47}$$

$$I = diag(1, \dots, 1) \tag{3.48}$$

It is easy to check that M is symmetric and semi-positive definite.

Let the column vectors $\boldsymbol{a}_0, \dots, \boldsymbol{a}_{d-1}$ be the solutions of equation (3.45), ordered according to their eigenvalues, $\lambda_0 < \cdots < \lambda_{d-1}$. Thus, the embedding is as follows:

$$\boldsymbol{x}_i \to \boldsymbol{y}_i = A^T\boldsymbol{x}_i \tag{3.49}$$

$$A = (\boldsymbol{a}_0, \boldsymbol{a}_1, \dots, \boldsymbol{a}_{d-1}) \tag{3.50}$$

where $\boldsymbol{y}_i$ is a $d$-dimensional vector, and *A* is a $n \times d$ matrix.

The NPE transformation is illustrated graphically in figure 3.18 [91]:

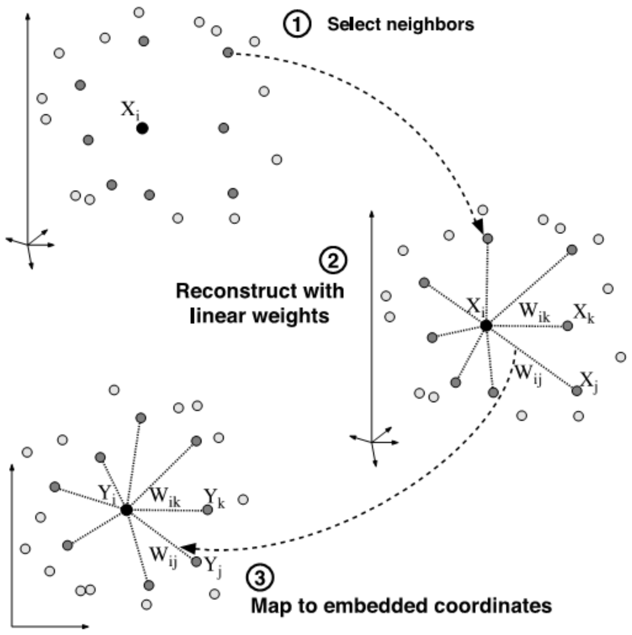

① Select neighbors
X_i
② Reconstruct with linear weights
X_i
W_ik
X_k
W_ij
X_j
Y_i
W_ik
Y_k
W_ij
Y_j
③ Map to embedded coordinates

*Figure 3.18: Application example of the NPE transform.*

### 3.1.3.1.6. Kernel Principal Component Analysis (KPCA)

In the field of multivariate statistics, Kernel Principal Component Analysis (KPCA) [92] is an extension of Principal Component Analysis (PCA) using techniques of kernel methods. Using a kernel, the originally linear operations of PCA are done in a reproducing kernel Hilbert space with a non-linear mapping.

Recall that conventional PCA operates on zero-centered data; that is:

$$\frac{1}{N}\sum_{i=1}^{N} x_i = 0 \tag{3.51}$$

It operates by diagonalizing the covariance matrix:

$$C = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T \tag{3.52}$$

in other words, it gives an eigendecomposition of the covariance matrix:

$$\lambda v = Cv \tag{3.53}$$

which can be rewritten as:

$$\lambda x_i^T v = x_i^T C v, \forall i \in [1, N] \tag{3.54}$$

To understand the utility of kernel PCA, particularly for clustering, observe that, while N points cannot in general be linearly separated in $d < N$ dimensions, they can almost always be linearly separated in $d \geq N$ dimensions [93]. That is, given $N$ points, $x_i$, if we map them to an *N*-dimensional space with:

$$\Phi(x_i) \text{ where } \Phi\colon \mathbb{R}^d \to \mathbb{R}^N \tag{3.55}$$

it is easy to construct a hyperplane that divides the points into arbitrary clusters. Of course, this $\Phi$ creates linearly independent vectors, so there is no covariance on which to perform eigendecomposition explicitly as we would in linear PCA.

Instead, in kernel PCA, a non-trivial, arbitrary $\Phi$ function is 'chosen' that is never calculated explicitly, allowing the possibility to use very-high-dimensional $\Phi$'s if we never have to actually evaluate the data in that space. Since we generally try to avoid working in the $\Phi$-space, which we will call the 'feature space', we can create the N-by-N kernel:

$$K = k(x, y) = \big(\Phi(x), \Phi(y)\big) = \Phi(x)^T \Phi(y) \tag{3.56}$$

which represents the inner product space of the otherwise intractable feature space. The dual form that arises in the creation of a kernel allows us to mathematically formulate a version of PCA in which we never actually solve the eigenvectors and eigenvalues of the covariance matrix in the $\Phi(x)$-space. The N-elements in each column of $K$ represent the dot product of one point of the transformed data with respect to all the transformed points (N points).

Because we are never working directly in the feature space, the kernel-formulation of PCA is restricted in that it computes not the principal components themselves, but the projections of our data onto those components. To evaluate the projection from a point in the feature space $\Phi(x)$ onto the $k^{th}$ principal component $V$ (where superscript k means the component k, not powers of k):

$${V^k}^T \Phi(x) = \left( \sum_{i=1}^{N} {a_i}^k \, \Phi(x_i) \right)^T \Phi(x) \tag{3.57}$$

We note that $\Phi(x_i)^T \Phi(x)$ denotes dot product, which is simply the elements of the kernel $K$. It seems all that's left is to calculate and normalize the ${a_i}^k$, which can be done by solving the eigenvector equation:

$$N\lambda a = Ka \qquad (3.58)$$

where $N$ is the number of data points in the set, and $\lambda$ and $a$ are the eigenvalues and eigenvectors of $K$. Then to normalize the eigenvectors $a^k$'s, we require that:

$$1 = (a^k)^T a^k \qquad (3.59)$$

Care must be taken regarding the fact that, whether or not $x$ has zero-mean in its original space, it is not guaranteed to be centered in the feature space (which we never compute explicitly). Since centered data is required to perform an effective principal component analysis, we 'centralize' $K$ to become $K'$:

$$K' = K - 1_N K - K 1_N + 1_N K 1_N \qquad (3.60)$$

where $1_N$ denotes a N-by-N matrix for which each element takes value $1/N$. We use $K'$ to perform the kernel PCA algorithm described above.

One caveat of kernel PCA should be illustrated here. In linear PCA, we can use the eigenvalues to rank the eigenvectors based on how much of the variation of the data is captured by each principal component. This is useful for data dimensionality reduction and it could also be applied to KPCA. However, in practice there are cases that all variations of the data are same. This is typically caused by a wrong choice of kernel scale [94][95].

In practice, a large data set leads to a large $K$, and storing $K$ may become a problem. One way to deal with this is to perform clustering on the dataset, and populate the kernel with the means of those clusters. Since even this method may yield a relatively large $K$, it is common to compute only the top $P$ eigenvalues and eigenvectors of $K$.

The Kernel PCA is illustrated graphically in the following example, where some well-known kernels are shown. Consider three concentric clouds of points shown in the following figure 3.19; we wish to use kernel PCA to identify these groups.

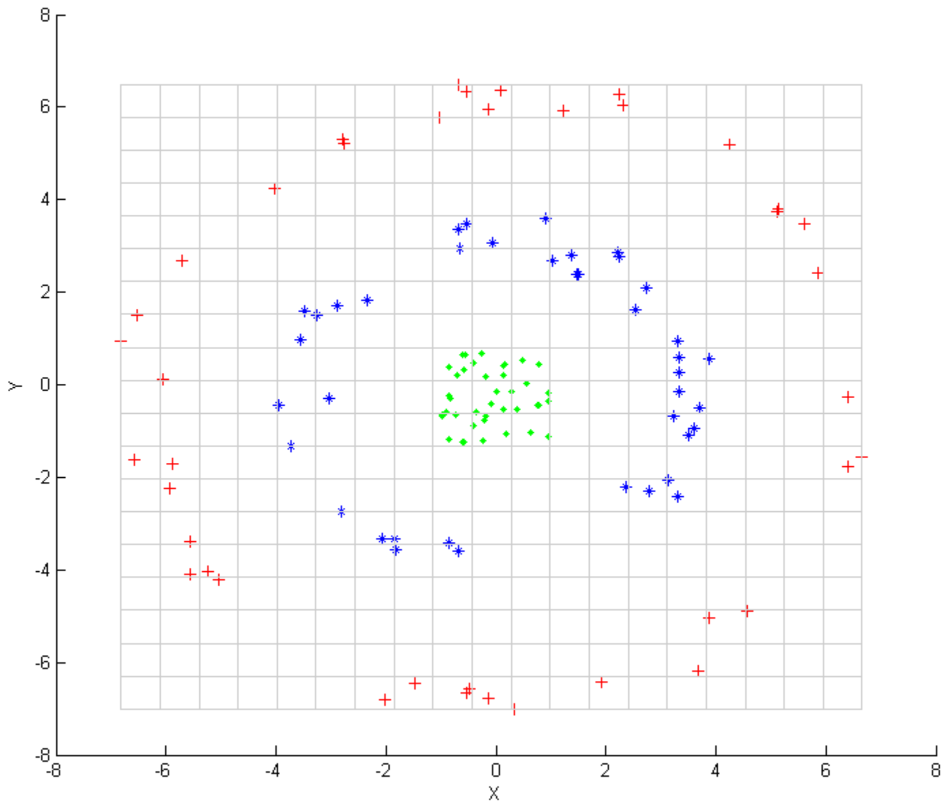


*Figure 3.19: Input points before kernel PCA.*

The color of the points is not part of the algorithm, but only there to show how the data groups together before and after the transformation.

First, consider the kernel:

$$k(x, y) = (x^T y + 1)^2 \tag{3.61}$$

Applying this to kernel PCA yields the next figure 3.20:

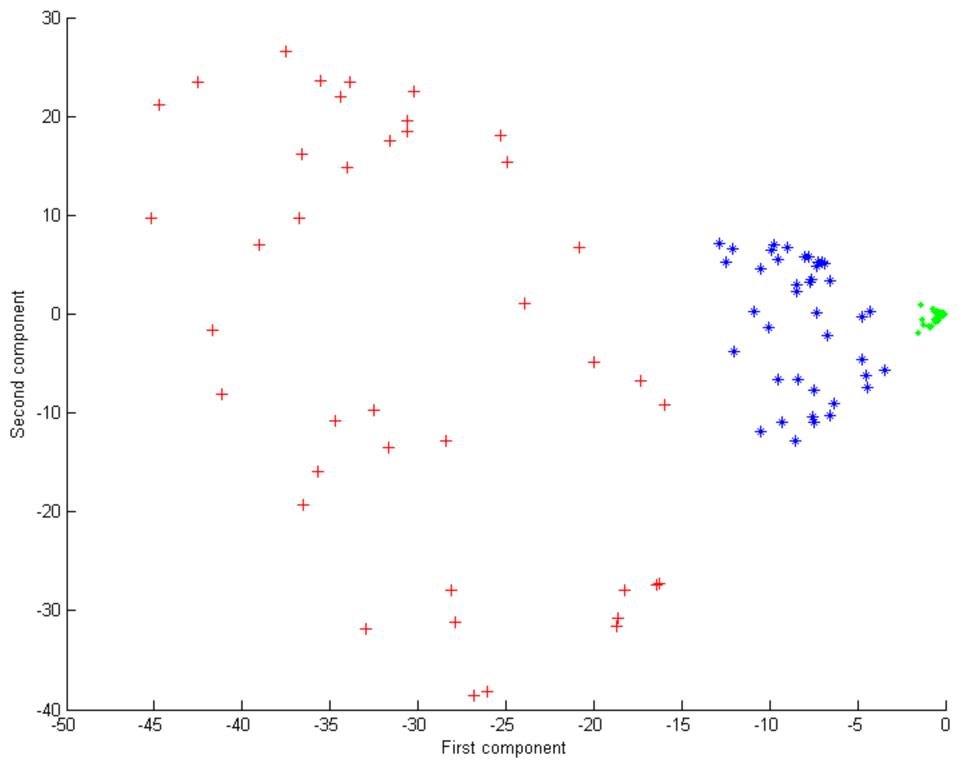


*Figure 3.20: Output after kernel PCA with* $k(x,y) = (x^T y + 1)^2$*. The three groups are distinguishable using the first component only.*

Now consider a Gaussian kernel:

$$k(x,y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}} \tag{3.62}$$

That is, this kernel is a measure of closeness, equal to 1 when the points coincide and equal to 0 at infinity.

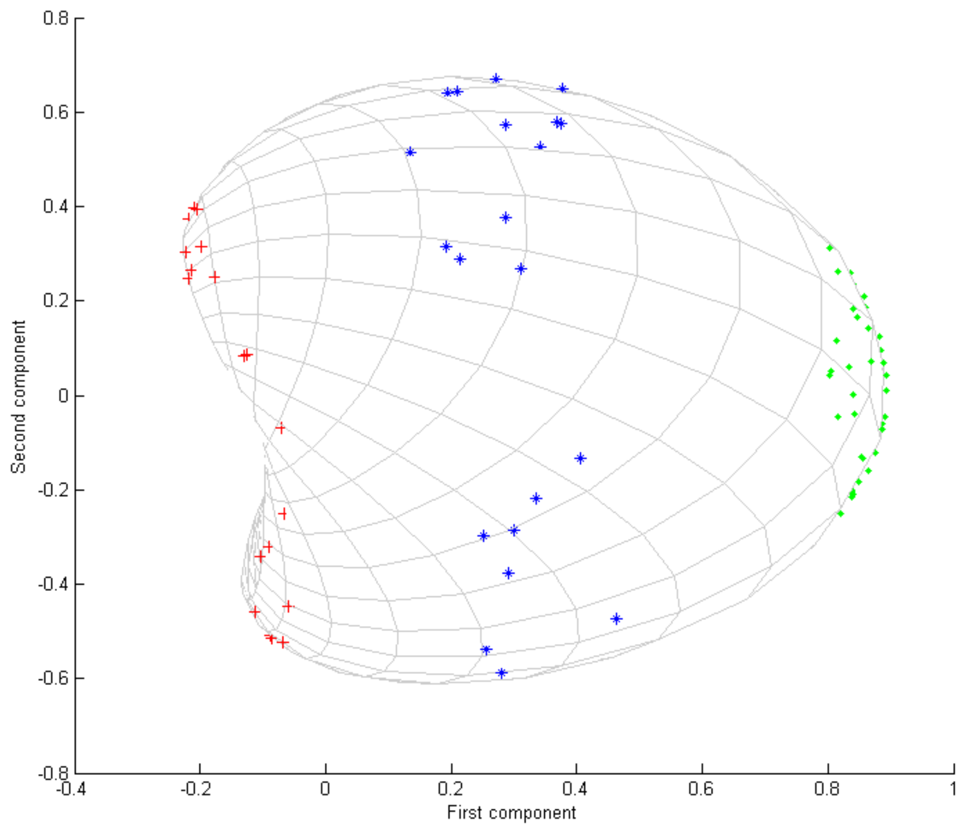


*Figure 3.21: Output after kernel PCA, with a Gaussian kernel.*

Note in particular that the first principal component is enough to distinguish the three different groups, which would be impossible using only linear PCA, because linear PCA operates only in the given (in this case two-dimensional) space, in which these concentric point clouds are not linearly separable.

### 3.1.3.2. Spectral unmixing Techniques

In many studies, hyperspectral analysis techniques are divided into full-pixel and mixed-pixel classification techniques [96][97][98], where each pixel vector defines a spectral signature or fingerprint that uniquely characterizes the underlying materials at each site in a scene. Full-pixel classification techniques assume that each spectral signature comprises the response of one single underlying material. Often, this is not a realistic assumption, because the spatial resolution of the sensor is not fine enough to separate different pure signature classes at a macroscopic level. In consequence, these can jointly occupy a single pixel, and the resulting spectral signature will be a composite of the individual pure spectral, often called endmembers in hyperspectral imaging terminology [99].

In this chapter, we explore the use of spectral unmixing for feature extraction prior to supervised classification of hyperspectral data using SVM. Moreover, differently from most feature extraction techniques available in literature, the features obtained using linear spectral unmixing are potentially easier to interpret due to their physical meaning [100].

Hyperspectral data are often used to determine what materials are present in a scene. In our case, materials of interest could include some tissues as tumor, healthy, necrosis in addition to blood, veins and surgical elements, etc. In fact, each pixel of a hyperspectral image could be compared to a material database to determine the type of material making up the pixel.

Spectral unmixing aims at the decomposition of the mixed pixel spectrum into its constituent spectra, also called endmembers [101]. Each pixel in the hyperspectral image can be considered as being composed of linear combination of ground spectra or endmembers with each endmember contributing to the pixel spectra. Thus the spectral signature at each pixel in a *L*-dimensional hyperspectral image, $Y \in R^L$, when *p* is the number of endmembers, can be expressed as,

$$y = x + n \tag{3.63}$$

where, *y* - *L*-dimensional pixel vector.

*x* and *n* - *L*-dimensional signal and noise vectors respectively.

Since the signal vectors lie in an unknown *p*-dimensional subspace, each signal vector is given as,

$$x = Ms = \sum_{i=1}^{p} m_i s_i \tag{3.64}$$

where, $M$ is a *L*×*p* matrix, whose columns are *L*×1 endmembers.

$s$ is the abundance fraction of each endmember in a pixel.

In essence, spectral unmixing can be defined as the process of determination of the number of image endmembers and their pure signatures and the amount in which they appear in the given mixes pixel [58]. Namely, Spectral unmixing consists of estimating the fraction of the pixel area covered by each material present in the scene [4].

Pixel unmixing algorithms can be separate into two main areas: Endmember Determination and Abundance Estimation algorithms as shown in figure 3.22.

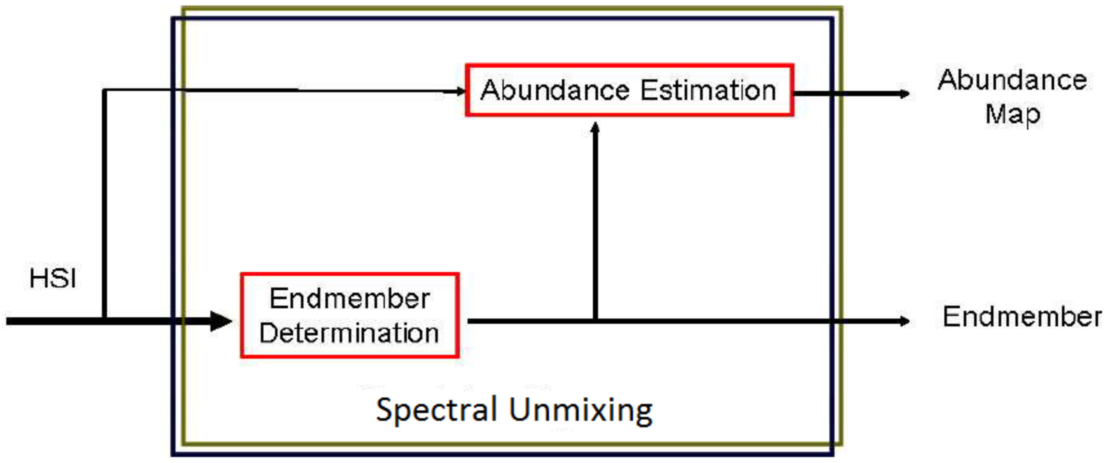


*Figure 3.22: Spectral unmixing Diagram Process.*

Before describing the algorithms considered, it should emphasize the context in which these algorithms will be applied. In this regard, figure 3.23 [103] describes the classic method for analyzing hyperspectral images using linear mixed model. As shown in figure 3.23 [103], the methodology starts from a hyperspectral image and performs the following steps:

1. **Dimensionality reduction:** This step is optionally used by certain algorithms to reduce the computational load of successive steps by removing noise and redundant information in the image.
2. **Endmembers extraction:** In this step the pure spectral signatures that combine to result in mixed pixel in the image are identified.
3. **Abundances estimation:** The abundance of pure spectral signatures or endmembers is estimated at each pixel of the image.

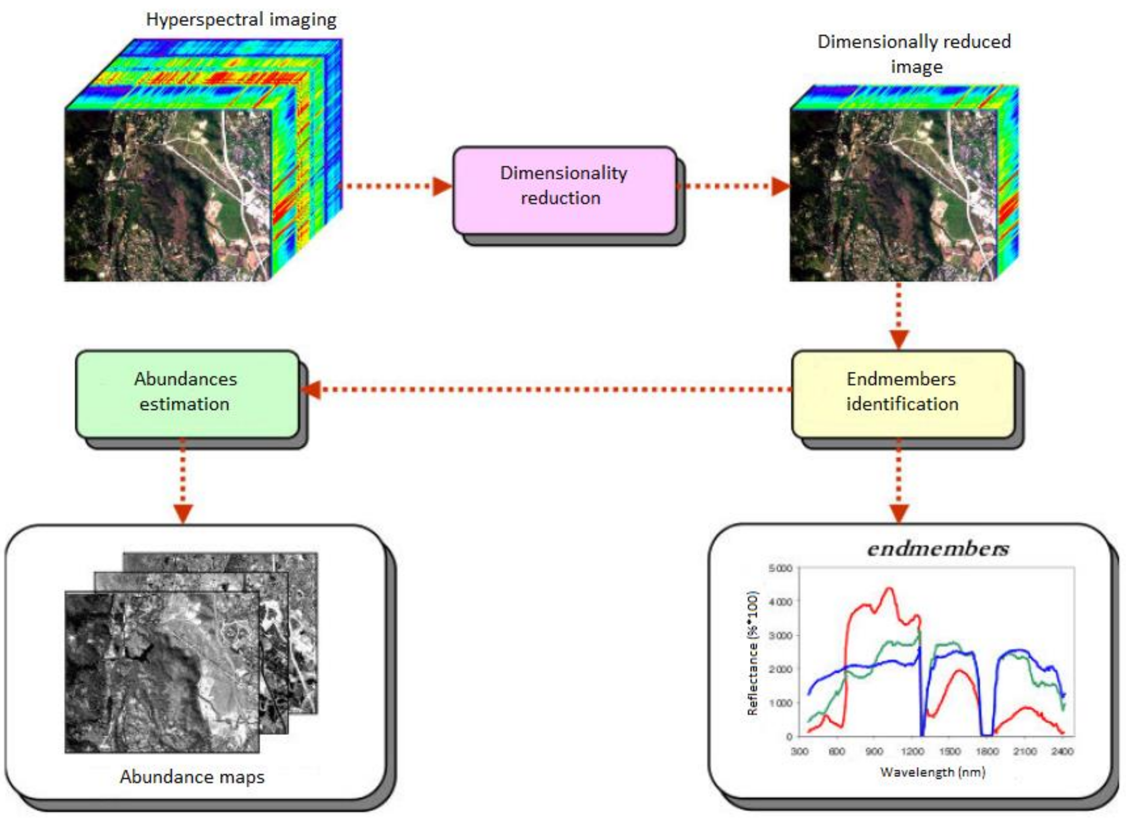


*Figure 3.23: classic method for analyzing hyperspectral images using linear mixed model.*

The spectral unmixing chain can be divided in three steps: number of endmember estimation, endmember extraction and abundances estimation.

### 3.1.3.2.1. Number of endmember estimation

First, the number of endmembers $p$ is estimated directly from the original $n$-dimensional hyperspectral image. For this purpose, two standard techniques widely used in the literature such as the HySime method [59] and the VD concept [104] are recommended. These techniques are described in the next section 3.1.4 corresponding to feature selection.

In our case, the estimation of number of endmembers is not necessary because this number is known to us a priori and associated with the number of classes.

### 3.1.3.2.2. Endmember extraction algorithms

Once the number of endmembers $p$ has been estimated, an automatic algorithm to extract a set of endmembers from the original hyperspectral image is applied [105]. There are many algorithms to unmix hyperspectral data each with their own strengths and weaknesses. Many algorithms assume that pure pixels (pixels which contain only one material) are present in a scene.

Endmember extraction algorithms can be divided into two groups: first those based only in image spectral information and secondly those that uses both spectral and spatial information. In this dissertation we have considered only the algorithms in the first group, where the used endmember extraction techniques have been: OSP [106], VCA [107] and N-FINDR [108] as representative automatic algorithms of the trend based on use only spectral information. Some examples of algorithms belonging to the second group are: Automatic Morphological Endmember Extraction (AMEE) [109], Spatial Spectral Endmember Extraction (SSEE) [110] y Spatial Pre-Processing (SPP) [111] as representative automatic algorithms of the trend based using both spatial and spectral information.

The endmember extraction algorithms based on spectral information depends on the ability to discriminate the pixels based solely on its spectral features. The following sections describe, the OSP, N-FINDR and VCA algorithms as three of the most representative techniques within this category. Other classic techniques such as PPI algorithm have been discarded for this study because of its semi-supervised nature, making it difficult to obtain consistent results in several automated executions.

#### 3.1.3.2.2.1. Orthogonal Subspace Projection (OSP)

The OSP algorithm was initially developed to find spectral signatures using the concept of orthogonal projections [112][113]. The algorithm uses an orthogonal projection operator that is given by the expression [114]:

$$P_U^{\perp} = I - U(U^T U)^{-1} U^T \tag{3.65}$$

where $U$ is a matrix of spectral signatures, $U^T$ is the transpose of this matrix, and $I$ is the identity matrix. The algorithm uses the operator shown in equation (3.65) repetitively until finding a set of $p$ orthogonal pixels from an initial pixel. The iterative process carried out by this algorithm can be summarized as follows:

1. Calculate $t_0$, the brightest pixel in the hyperspectral imaging, using the following expression: $t_0 = arg\{max_{(x,y)} F(x,y)^T \cdot F(x,y)\}$, where $F(x,y)$ is the pixel in the $(x,y)$ coordinates of the image. As can be seen, the brightest pixel is the one with the greatest value when performing the cross product between the vector associated with that pixel and its transposed $F(x,y)^T$, or what is the same, the first norm of the pixel.
2. Applying an orthogonal projection operator denoted as $P_U^\perp$, based on the expression (3.65), with $U = t_0$. This operator is applied to all pixels in the hyperspectral imaging.
3. Then the algorithm finds a new endmember called $t_0$ with the highest value in the complementary space $\langle t_0 \rangle^\perp$, orthogonal to $t_0$, as follows: $t_1 = arg\{max_{(x,y)} [P_U^\perp \cdot F(x,y)]^T \cdot [P_U^\perp \cdot F(x,y)]\}$. In other words, the algorithm searches for the pixel with the highest orthogonality with respect to $t_0$.
4. The next step is to modify the $U$ matrix by adding the new endmember found, i.e., $U = [t_0 t_1]$.
5. Next, the algorithm finds a new endmember called $t_2$ with the highest value in the complementary space $\langle t_0, t_1 \rangle^\perp$, orthogonal to $t_0$ and $t_1$, as follows: $t_2 = arg\{max_{(x,y)} [P_U^\perp \cdot F(x,y)]^T \cdot [P_U^\perp \cdot F(x,y)]\}$. It should be noted that, unlike step 3) in which $U = t_0$, at this point the orthogonal projector is based on a matrix $U = [t_0 t_1]$.
6. The process is repeated iteratively until the desired number of endmembers is found.

This algorithm is effective in identifying a set of spectrally differentiated endmembers by the orthogonality condition imposed in the search process. As a negative feature, the algorithm can be sensitive to outliers and anomalous pixels [103][106].

#### 3.1.3.2.2.2. Vertex Component Analysis (VCA)

This algorithm also makes use of the concept of orthogonal projections. However, unlike the previously described algorithm OSP, VCA algorithm exploits the fact that the endmember are the vertices of a simplex and that the affine transformation of a simplex is also a simplex. As a result, VCA models the data using a positive cone whose projection in an appropriately chosen hyperplane is another simplex whose vertices are the final endmembers. After projecting the data in the selected hyperplane, the VCA algorithm projects all the image pixels in a random direction and uses the pixel with the greatest projection as the initial endmember. The others endmembers are sequentially identified by projecting iteratively the data in a direction orthogonal to the subspace spanned by the endmembers currently selected. The new endmember is then selected as the corresponding pixel to the most extreme projection, and the procedure is repeated until p endmembers are selected [103][107].

#### 3.1.3.2.2.3. N-FINDR

N-FINDR algorithm [115][116] uses a technique based on identifying endmembers as the vertices of the simplex with largest volume that can form on the set of points. N-FINDR does not work with the whole data hypercube but with a simplification of it with as many bands as endmembers are desired to find. For this type of dimensionality reduction are often used techniques as explained above, such as Principal Component Analysis (PCA) [117], Minimum Noise Fraction (MNF) [117] or Independent Component Analysis (ICA) [82]. The only parameter that has this algorithm is the number of endmembers to identify. The operation of the algorithm is described in the following steps:

1. Make a reduction of the image to a number of bands equal to the number of endmembers to be extracted, using one of the dimensionality reduction algorithms discussed above.
2. Select a random number of pixels that are labeled as endmembers. This initial selection will be refined iteratively.
3. The third step is to select a pixel of the original image. This pixel will successively exchange with each endmembers initially selected.
4. As the pixel will exchange with the initial endmembers, the volume of the hyperpolygon formed with the new point considered is calculated.
5. If the volume obtained after the exchange is greater than it had before the exchange, the new point results in a replacement in the endmembers set and the new pixel becomes part of endmembers set. Otherwise, the exchange is reversed.
6. Steps 3-5 are iteratively repeated to check all pixels of the image. Of such form that at the end of process we have a set of endmembers such that its volume is the largest possible.

It should be noted that in the second step of the algorithm, an initial set of endmembers is randomly established. If the initial estimate is suitable, the algorithm will reach the optimal solution. On the contrary, an erroneous initial estimate may result in failure to reach the optimal solution but we stay in a local maximum of the growth function of the hyperpolygon [116]. The algorithm assumes that an increase in the volume of the defined hyperpolygon by incorporating a new pixel in the endmembers set carries a higher quality thereof. However, figure 3.24 shows that the fact of using a polygon of greater volume does not ensure a better description of the set of points [103]. A more reliable parameter is the increase in the number of pixels that can be described using the new set of endmembers.

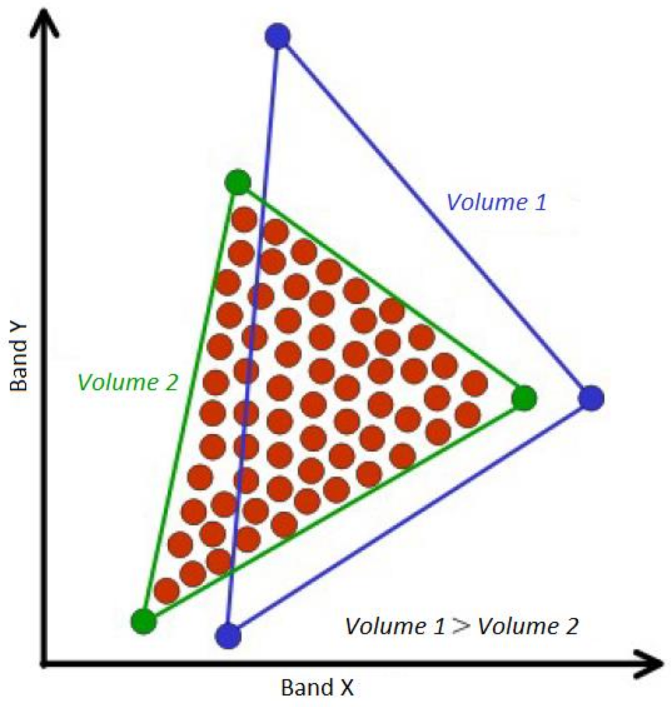


*Figure 3.24: N-FINDR algorithm operation.*

To conclude the description of this method, it is important that the endmembers identified by the algorithm N-FINDR correspond to pixels belonging to the original dataset. Using this algorithm, it is not possible to generate artificial endmembers because the replacements are always performed using existing points in the set of available samples. Thus, it may happen that the selected endmembers are not the most pure. Furthermore, the method is sensitive to outliers [103][108].

## 3.1.3.2.3. Abundance estimation algorithms

Finally, linear spectral unmixing (either unconstrained or constrained) can be used to estimate the abundance of each endmember in each pixel of the scene, providing a set of p abundance maps [100].

### 3.1.3.2.3.1. Linear Spectral Unmixing

In hyperspectral images, spectral mixing is the result of mixing of two or more spectrally distinct substances. Spectral unmixing is the process by which we can identify the constituents of the mixed pixel and their proportions. Generally, two models of mixing are assumed: linear and nonlinear [58]. As opposed to nonlinear unmixing, which generally require detailed information about physical properties that may not be always available, linear spectral unmixing consists of identifying the pure spectral components or endmembers. When the pure spectral signatures are identified, the proportion of each material in each pixel can be estimated. Abundances provide additional information about the composition of each pixel; if this information is used in a correct way, it may complement the results provided by traditional “hard” classification techniques. Moreover, non-linear mixing results from multiple scattering often due to non-flat surface [100].

The simplest and the most commonly assumed model for a mixed spectrum is a linear model [96]. A single pixel can be portrayed as a checkerboard mixture, assuming that there is no multiple scattering between components, then the spectral response of the pixel is a linear combination of the fractional abundances (area covered by each endmember in the pixel) of the individual substances [101], hence the term Linear Mixture Model (LMM). If there are $p$ endmembers, then the linear mixture model can be expressed as

$$x = \sum_{i=1}^{p} m_i s_{ij} + w_j = Ms + w \tag{3.66}$$

$$j = 1,2, \dots \dots, N$$

where, $x$ is the $L \times 1$ received pixel spectra.

$M$ is a $L \times p$ matrix, whose columns are $L$×1 endmembers.

$s$ is the abundance fraction of each endmember in a pixel.

$w$ is the $L \times 1$ additive noise.

$N$ is the number of pixels in the image.

In linear mixing models each pixel is modeled as a linear sum of all the radiated energy curves of materials making up the pixel. Therefore, each material contributes to the sensor's observation in a linear fashion.

Additionally, a conservation of energy constraint is often observed thereby forcing the weights of the linear mixture to sum to one in addition to being positive. Namely, to be physically meaningful, the linear mixture model can be subjected to following two constraints; the first is the Abundance Non-negativity Constraint (ANC) [118],

$$s_{ij} \geq 0 \tag{3.67}$$

and the second is the full additivity Abundance Sum-to-one Constraint (ASC) [118],

$$\sum_{i=1}^{p} s_{ij} = 1 \tag{3.68}$$

#### 3.1.3.2.4 Abundance Maps

Abundance estimation is the problem of estimating the set of corresponding fractions that indicate the proportion of each endmember present in the pixel of a hyperspectral image [119][120].

Once the fundamental materials of a scene are determined, it is often useful to construct an abundance map of each material which displays the fractional amount of material present at each pixel.

## 3.1.4. Feature selection

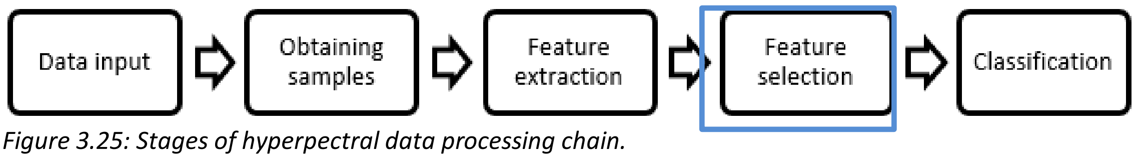


*Figure 3.25: Stages of hyperpectral data processing chain.*

The fact of choosing a set of features whose dimensionality is the most appropriate and reasonable is also important in reducing the high dimensionality of the data. In this regard, feature selection approaches try to find a subset of the original variables (features). Namely, given a set of training patterns, feature selection aims to choose from the set of initial spectral bands of a hyperspectral image, those that allow obtaining more information in order to make a more efficient classification. Because in some cases, data analysis such as classification can be done in the reduced space more accurately than in the original space [5].

In machine learning, feature selection is the process of selecting a subset of relevant features for use in model construction and these techniques are used for three reasons:

- Simplification of models to make them easier to interpret by users. [121]
- Shorter training times.
- Enhanced generalization by reducing overfitting [122] (formally, reduction of variance [121]).

The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information [122]. Redundant or irrelevant features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated [123].

Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points).

A feature selection algorithm can be seen as the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets. The simplest algorithm is to test each possible subset of features finding the one which minimizes the error rate. This is an exhaustive search of the space and is often expensive from a computational point of view.

This way, the features selection step allows an efficient and selective process for choosing the best subset based on the large data, namely this block provides the best combination of bands and/or features for a given problem.

However, to find the best combination of bands and/or features for a given problem fast algorithms are needed. The combination of these features is complex and an optimal solution through a single data transformation cannot be guaranteed.

To address the problem of choosing this subset *d* of characteristics from an initial set of measures *D*, with *d* < *D*, two techniques are used in this project: HySime and Virtual Dimensionality.

#### 3.1.4.1. Hyperspectral Subspace Identification by Minimum Error

Hyperspectral Subspace Identification by Minimum Error (HySime) is a denoising method which detects the regions of the spectrum that are more affected by noise. Most noise reduction methods use the spatial distribution of the image pixels. However, HySime does not use this kind of information. This strategy allows solving those cases where only information about a few (not necessarily spatially related) numbers of pixels in the image is available [124].

In [125], it was proposed that an unsupervised approach that can be utilized to select the most distinctive and informative bands. It employs a way for selecting the initial feature based on the orthogonal distance from the Prototype space (PS) diagonal and determines the optimal feature size by employing the HySime algorithm in the PS [58]. In order to define the band correlations a criterion based on the tangent angles between the band vectors in prototype space is used to select the initial feature from the hyperspectral image [59].

HySime concentrates on reduction or signal subspace identification (SSI). It is a minimum mean square error based approach to infer the subspace by minimizing the power of the signal projection error and the power of the noise projection. This algorithm is eigen-decomposition based and adaptive (i.e., it does not depend on any tuning parameters) method which means that it decomposes or reduces the original matrix into subsets of eigen vectors. The subspace obtained by HySime algorithm optimally represents the original input matrix with minimum error [58][126].

HySime starts by estimating the signal and the noise correlation matrices using multiple regressions and then it selects the subset of eigenvectors that best represents the signal subspace in the minimum mean square error sense. A subset of eigenvectors of the signal correlation matrix is then used to represent the signal subspace of reduced dataset. The signal subspace is inferred by minimizing the sum of the projection error power with the noise power, which are, respectively, decreasing and increasing functions of the subspace dimension. Therefore, if the subspace dimension is overestimated the noise power term is dominant, whereas if the subspace dimension is underestimated the projection error power term is the dominant. The overall scheme is adaptive in the sense that it does not depend on any tuning parameters [59].

HySime algorithm comprises of two parts, namely Noise estimation and sub-space identification. A multiple regression based approach for noise estimation and input covariance matrices from high dimensional data is used. In Sub-space identification the noise and input correlation matrices are computed. For more information, HySime algorithm has been explained in its corresponding section 3.1.1.1.

Some experimental study about the denoising effect on the selection capability of different band selection techniques for regression tasks applied to real hyperspectral datasets have shown good results offered by HySime. However if the number of bands is relatively small, the noise reduction using the HySime method does not significantly improve the regression error [124].

### 3.1.4.2. Virtual Dimensionality (VD)

When very high-dimensional data are well structured, data tend to be distributed in a low-dimensional space. Virtual Dimensionality (VD) is defined as the minimum number of spectrally distinct signal sources that characterize the hyperspectral data from the perspective view of target detection and classification rather than the image endmembers defined in [127], which are idealized pure signatures. It is different from the commonly used intrinsic dimensionality (ID) in the sense that the signal sources are determined by the proposed VD based only on their distinct spectral properties. These signal sources may include known and unknown image endmembers, natural signatures, anomalies, and unknown interfering sources, which cannot be identified by prior knowledge [128]. With this definition, a Neyman–Pearson detection theory-based thresholding method is developed to determine the VD of hyperspectral imagery, where eigenvalues are used to measure signal energies in a detection model. This eigenthresholding based method is Harsanyi–Farrand–Chang (HFC) [104].

Harsanyi, Farrand, and Chang [129] developed a Neyman-Pearson [130] detection theory-based thresholding method (HFC) to determine the number of spectral endmembers in hyperspectral data, referred to in [104] as virtual dimensionality (VD). The HFC method is based on a detector built on the eigenvalues of the sample correlation and covariance matrices [59]. Let the eigenvalues generated by the sample correlation matrix and the sample covariance matrix be denoted by correlation eigenvaules and covariance eigenvalues, respectively. Since the component dimensionality is equal to the total number of eigenvalues, each eigenvalue specifies a component dimension and provides an indication of the significance of that particular component in terms of energy or variance. If there is no signal source contained in a particular component, the corresponding correlation eigenvalue and covariance eigenvalue in this component should reflect only the noise energy, in which case, correlation eigenvalue and covariance eigenvalue are equal. This fact provides a base from which to formulate the difference between the correlation eigenvalue and its corresponding covariance eigenvalue as a binary composite hypothesis testing problem. The null hypothesis represents the case of the zero difference, while the alternative hypothesis indicates the case that the difference is greater than zero. When the Neyman–Pearson test is applied to each pair of correlation eigenvalue and its corresponding covariance eigenvalue, the number of times the test fails indicates how many signal sources are present in the image. In other words, a failure of the Neyman–Pearson test in a component indicates a truth of the alternative hypothesis, which implies that there is a signal source in this particular component. Using this approach, it is possible to estimate the VD with the receiver operating characteristic (ROC) analysis for evaluating the effectiveness of the decision [104].

HFC method model the correlation eigenvalue and the corresponding covariance eigenvalue as random parameters in [131], the sample size must be sufficiently large to ensure that the covariance between these two types of eigenvalues is asymptotically zero. However, this may not be valid for a small sample size.

## 3.2. Classification algorithms

These sections are based in the classification process, particularly in support vector machine classifier.

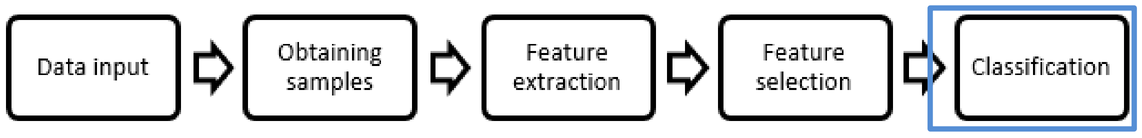


*Figure 3.26: Stages of hyperspectral data processing chain.*

For our set of experiments, we use the library LIBSVM [Online: https://www.csie.ntu.edu.tw/~cjlin/libsvm/].

The purpose of a supervised classifier is to use a set of observations called training set to find a decision function. This function classifies every new object in a pre-defined class. This is achieved by a learning process while the training objects are classified. With recent technological advances and the large amount of hyperspectral data, there are the necessary means for efficient classification to discriminate classes according to the spectral resolution for each pixel of an image obtained. However, the large number of bands is the feature that produces greatest complexity in analysis techniques [5].

In this regard, conventional methods of classification as the machine learning (ML) algorithm can be applied to hyperspectral data, but require complex processing due to the high dimensionality. The difficulty which gets many methods based on conventional statistical approaches is that these employ a specified covariance matrix of each class [132]. Another disadvantage of this type of functions which works with covariance matrices is that the classifications made with little or limited number of training sets when working with high dimensionality data is that result in poor generalization processes (classification) [68].

### 3.2.1. Support Vector Machine Classifier (SVM)

At the beginning of XXI century it was tested the great effectiveness of methods based on statistical learning theory to work with both problems: high dimensional and sparse training set. The training of the classifiers, both statistical and neural networks make use of the principle of Empirical Risk Minimization (ERM), which consists of allowing the minimization of the error rate for a given training set. The problem arises when it is necessary to extend or to generalize that classification to the rest of objects, then a good performance is not achieved, i.e., the resulting error rate is higher than for the training set.

The Support Vector Machine (SVM) is a pattern recognition supervised method recently introduced in the framework of statistical learning theory of Vapnik Vladimir and his work team at AT&T Labs [133]. It combines following ideas: the optimal hyperplanes search technique as a solution, the idea of convolution scalar product, the extension of the linear functions to nonlinear and the notion of soft margin to allow for errors in the training patterns. An important advantage is that it works on the principle of Structural Risk Minimization (SRM), it is better than ERM which use others many techniques. SVM then enables better generalization rather than a better classification of the training set (at the level of errors).

There are two other reasons that have increased the interest in this newfangled classifier. SVM can be reduced to a problem of convex quadratic programming (QP), which is easier to solve compared to traditional methods and that seem to have a better performance (more robust) with high amounts of data. SVM has been used hitherto in many fields, such as: Text categorization, recognition of hand-written texts, image classification, bioinformatics, remote sensing, and now also in medical hyperspectral imaging, where it seems to have a higher performance than other classical techniques used [134].

#### 3.2.1.1 Theoretical foundations of SVM classifier

This classifier belongs to the family of linear classifiers that induce linear separators or hyperplanes in spaces of high dimensionality characteristics, though may be easily adapted to act as non-linear classifiers by applying a function or not linear kernel in the input data. Its main objective is to get one surface (or hyperplane) capable of separating the different classes that can be grouped in a data distribution of a N-dimensional space, using an optimization process based on obtaining vectors which defining the class boundaries. These vectors are usually referred support vectors [135]. If we see the input data as two sets of vectors in a N-dimensional space, the SVM algorithm objective is simply to build a separating hyperplane in that space, which maximizes the margin of distance to the two data sets [136].

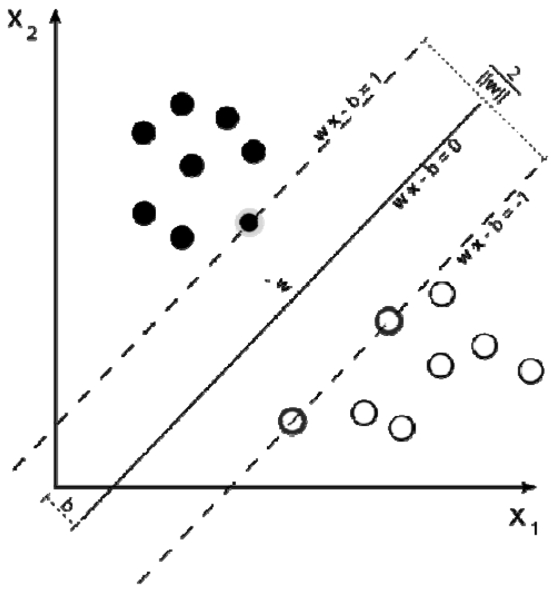


*Figure 3.27: Functional diagram of the SVM classifier.*

In figure 3.27 [5] we can see how to calculate this separating hyperplane, constructing two parallel hyperplanes, one on each side of the first. The two parallel hyperplanes are pushed or widened to as close as possible to the datasets. Intuitively, a good separation is achieved when the separating hyperplane is farthest from both classes. The greater the separation distance, the lower will generally be the classifier error [135]. In mathematical terms, given a training set of the equation:

$$D = \{(x_i, c_i) | x_i \in R^p, c_i \in \{-1,1\}\}_{i=n}^{n} \tag{3.69}$$

Where $c_i$ is 1 or -1, indicating the class to which $x_i$ belongs. Each $x_i$ is a real vector p-dimensional, we want to obtain a maximum distance hyperplane to the training sets which divide those belonging to $c_i = 1$ of those with the value $c_i = -1$. Any hyperplane can be written as a set of points $x$ that satisfies the equation:

$$w \cdot x - b = 0 \tag{3.70}$$

The $w$ vector is a normal vector perpendicular to the hyperplane. The $\frac{b}{\|w\|}$ parameter determines the displacement of the hyperplane over the origin. We want to choose the $w$ and the $b$ which maximize the distance between the two parallel hyperplanes, which are as far apart as possible depending on the data. These hyperplanes can be described with the formulas described below:

$$w \cdot x - b = 1 \tag{3.71}$$

and,

$$w \cdot x - b = -1 \tag{3.72}$$

Note that if the training set is linearly separable, we can choose two hyperplanes on the edge of the sets such that there are no points between them and then try to maximize their distance. Using geometry, we find that the distance between them is $\frac{2}{\|w\|}$, so it is intended to minimize $\|w\|$. As we have to prevent the points are located in the boundary area, we add the constraint of equation

$$w \cdot x_i - b \geq 1 \tag{3.73}$$

for $x_i$ belonging to the first class and the restriction of equation

$$w \cdot x_i - b \leq -1 \tag{3.74}$$

for $x_i$ in the second class.

This can be written as:

$$c_i(w \cdot x_i - b) \geq 1, \forall\, 1 \leq i \leq n \tag{3.75}$$

We can compact the expression to reach the optimization problem: Choose $w$, $b$ to minimize $\|w\|$:

$$\text{Subject to } c_i(w \cdot x_i - b) \geq 1, \forall\, 1 \leq i \leq n \tag{3.76}$$

The optimization problem presented above is difficult because only depends on a value $|w|$. The reason is that it is a non-convex optimization problem, which is known to be much more difficult to solve than the convex optimization problem. Fortunately, is possible to replace $\|w\|$ by $\frac{1}{2}\|w\|^2$ without changing the solution. This is an optimization problem of quadratic programming. More clearly, the optimization problem can be reformulated as follows:

$$\text{Minimize } \frac{1}{2}\|w\|^2, \text{ subject to } c_i(w \cdot x_i - b) \geq 1, \forall\, 1 \leq i \leq n \tag{3.77}$$

The 1/2 factor is used as a mathematical convenience. Now, the problem presented to us can be resolved through programs and standard quadratic programming techniques. Writing the classification rule dual in its extended form, this reveals that the maximum distance to the hyperplane, and therefore the task of classification, it is only a function of support vectors, ie, data that are on the borderline. The second form of SVM can be derived as the following expression:

$$max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \, c_i c_j x_i^T x_j \tag{3.78}$$

$$\text{Subject to } \alpha_i \geq 0, \text{ and } \textstyle\sum_{i=1}^{n} \alpha_i c_i = 0 \tag{3.79}$$

Where α terms are another representation of the weight vector in terms of the training set:

$$w = \sum_{i} \alpha_i c_i \, x_i \tag{3.80}$$

The original algorithm specifies a linear classifier, however, it may be modified to solve nonlinear classification problems, replacing the scalar product by a nonlinear kernel function. This allows the algorithm to determine the maximum distance to the hyperplane in a transformed features space. The transformation could be a nonlinear transformation and the transformed space could be a high dimensional space; this way, even though the classifier is a hyperplane in the space of high-dimensional features, it may not be linear in the original input space [137]. If the kernel used is Gaussian radial basis function type, the characteristics space is a Hilbert space of infinite dimension. The maximum distance is regulated, so the infinite dimension will not spoil the results [138]. Some of the commonly used kernels in SVM classifiers are listed below:

- Polynomial (homogeneous): $k(x, x') = (x \cdot x')^d$
- Polynomial (heterogeneous): $k(x, x') = (x \cdot x' + 1)^d$
- Radial Basis Function: $k(x, x') = exp(-\gamma \|x - x'\|^2), for\, \gamma > 0$
- Gaussian Radial Basis Function: $k(x, x') = exp\left(\frac{\|x - x'\|^2}{2\sigma^2}\right)$
- Sigmoide: $k(x, x') = tan(kx \cdot x' + c), for\ some\ k > 0\ and\ c < 0$

In literature, we can find examples of kernels based on spectral metrics commonly used in hyperspectral analysis [139]. As stated at the beginning, processing used by the SVM does not require a large number of training patterns, as long as the chosen patterns are truly representative [140].

### 3.2.1.2 Kernel functions available

Our classifier has the following kernel functions that allow us to project information to a higher dimensional space which increases the computational capacity of the classifier:

- **Linear:**

$$u' * v \tag{3.81}$$

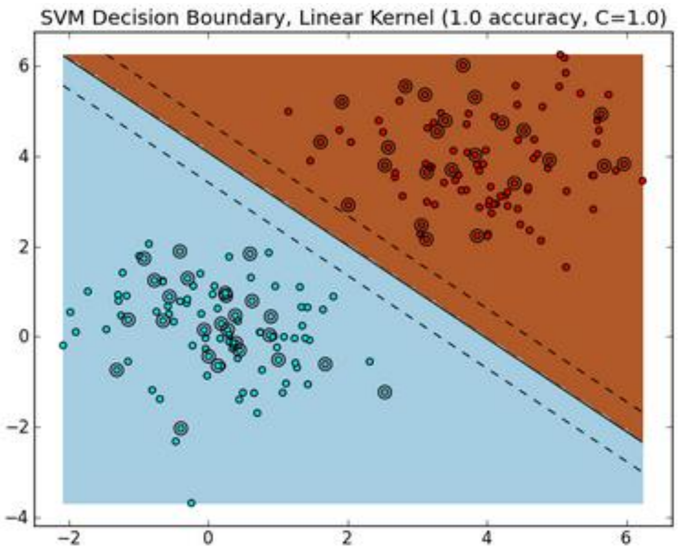


*Figure 3.28: The decision boundary of a Linear SVM.*

- **Polynomial:**

$$(\gamma * u' * v + coef0)^n \tag{3.82}$$

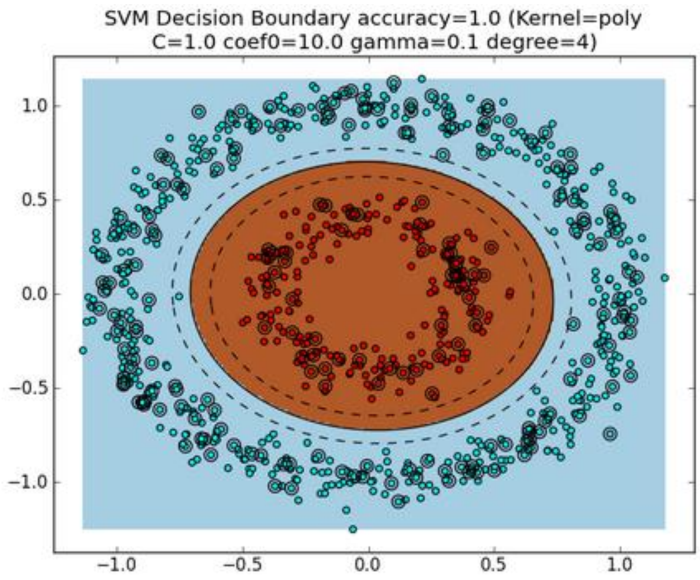


*Figure 3.29: The decision boundary with a Polynomial kernel.*

- **Radial basis function:**

$$e^{(-\gamma * |u-v|^2)} \tag{3.83}$$

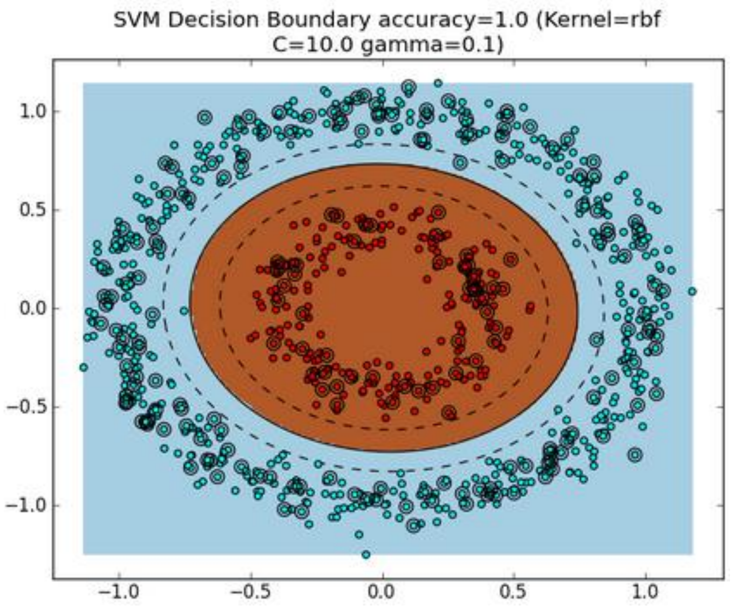


*Figure 3.30: The decision boundary with a Radial Basis Function (RBF) kernel.*

- **Sigmoid:**

$$\tanh(\gamma * u' * v + coef0) \tag{3.84}$$

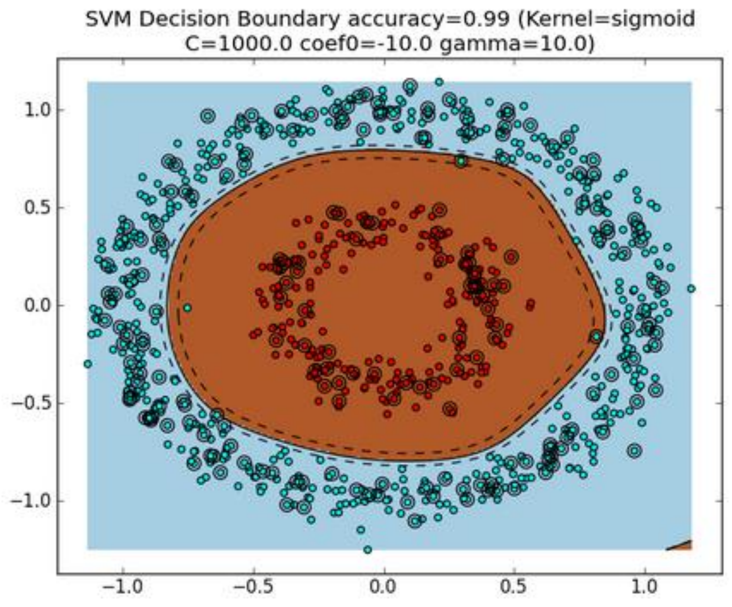


*Figure 3.31: The decision boundary with a Sigmoid kernel.*

- **Precomputed kernel:** Allows us to introduce our own kernel using a matrix.

  By default, the classifier works with the radial basis function kernel.

## 3.3. Summary

To take advantage of the large amount of information offered by hyperspectral imaging is necessary to design a robust preprocessing chain. This is because the high dimensionality that provides hyperspectral data leads to the presence of noise and redundant information which complicate its direct processing.

To address this issue, preprocessing chains usually consist of two main parts: feature extraction and feature selection, in which it is possible to apply many different techniques.

Firstly, feature extraction is responsible for extracting the main features of the hyperspectral data using transformations theory. Then feature selection is responsible of choosing the most important characteristics of this new set of transformed data. The result is an alternative representation of the original data set but with lower dimensionality and therefore much easier to process.

The ultimate goal of this process is to classify the data in the most efficient way possible. For this purpose a Support Vector Machine (SVM) will be used. SVM is a very robust classifier that provides very good results working with this type of data.

# Chapter 4

# Proposed process for cancer detection using hyperspectral images

In recent years the hyperspectral sensors technology has rapidly advanced providing data with very high spectral fidelity compared to multispectral systems. While these sensors facilitate the identification and classification, the high dimensionality and volume of data increases the bandwidth and computational complexity of the analysis. In addition, we have a large amount of redundancy in the hyperspectral data due to the high correlation between adjacent bands. In order to make an optimum classification and minimize the computational time, it is necessary to find a method to reduce the dimensionality of data while keeping the necessary spectral characteristics to classify data.

The most general processing chain can be described with a small group of elaborate block chains. In fact, after obtaining a sample set (forming the hypercube, filtering the data, extracting samples from its contour, labeling samples, etc.) the relevant information can be extracted from the hyperspectral data. Starting from the sample set, three main steps can be described in the processing chain: transformation and feature extraction, feature selection and classification.

In this dissertation, feature extraction is implemented in two different ways to reduce the dimensions of the data: dimensionality reduction and unmixing techniques. The feature selection process enables efficient and selective process for choosing the best subset, based on the high-dimensional data. The combination of these features is complex, so fast and efficient algorithms are needed for brain cancer detection. Finally, the classification step produces a high level result ready for user interpretation. In this regard, each process contains a classification stage which is performed by the supervised pattern recognition algorithm called Support Vector Machine (SVM).

In the beginning we only had *ex vivo* data available and in consequence the whole processing chain was used with this dataset while we waited to get the *in vivo* data. In this regard, we tested extensively all the processing chains with *ex vivo* data in order to select the processing chains that provided best results. Thus, when we had *in vivo* data available we could test the best combinations of processing with these new data.

In this chapter, we present all the possible combinations of block chains selected for experimentation. The reason for the choice of the processing block chains that we will present below is based on previous studies in other research frameworks (some outside the medical field) with hyperspectral imaging, in which the techniques presented in this dissertation obtained good results. These chains showed good performance in terms of accuracy percentage in classification with respect to others. However, there is a world of possibilities where is possible to replace, add, or remove a block chain, or research on new techniques in order to obtain good results in characterizing the image. Based on previous experiments with different types of images or datasets and using different percentages of training sets, we have considered the chains that are described below.

## 4.1. Process #1: Support vector machine classifier used in brain cancer detection

In this process the original data set is classified using SVM classifiers. In order to obtain the best possible results, the parameters of SVM classifier are studied. For this, all kernels available for this classifier are used to check the experimental results. The optimal parameters were selected using 10-fold cross-validation (selected after testing different configurations).

Furthermore, it is shown the process of obtaining the set of samples from the files generated by the camera. Along this processing, the hyperspectral data cubes are formed from the files produced by the camera. These cubes have areas with samples of interest (healthy, tumor and necrosis tissues) and other areas (table, test plates and glow in the samples) which must be removed from the image. The hypercubes are spatially reduced in order to process it more easily and quickly, and for extracting samples removing the other elements in the image a contour functions are used. In addition, the original data are noisy due to the circumstances in which they were taken. For this reason, in this chain is intended to apply the types of filtering discussed above, in order to further improve the results of the classification. Finally all samples obtained from each hypercube are united in a single set of samples whose samples are properly labeled.

Once obtained the sample set, it proceeds to the classification of the pixels using the SVM classification algorithm from the set of samples obtained and the training set. This stage produces a classified image with each pixel assigned to a particular class. This last step is common in almost all chains, so in the following sections where other processing chains are explained, it is possible to obviate the description of this block.

In conclusion, taking as a decision criterion the best results obtained, this chain is intended to get two objectives: to establish the best possible configuration of SVM algorithm that will be used throughout this dissertation as well as to establish, depending on the type of filter used, the set of samples that will be used in other processing chains, while showing the process of obtaining the samples to classify. For this reason, the rest of processing chains begin directly from the set of samples obtained.

The diagram used in this processing chain is shown in figure 4.1:

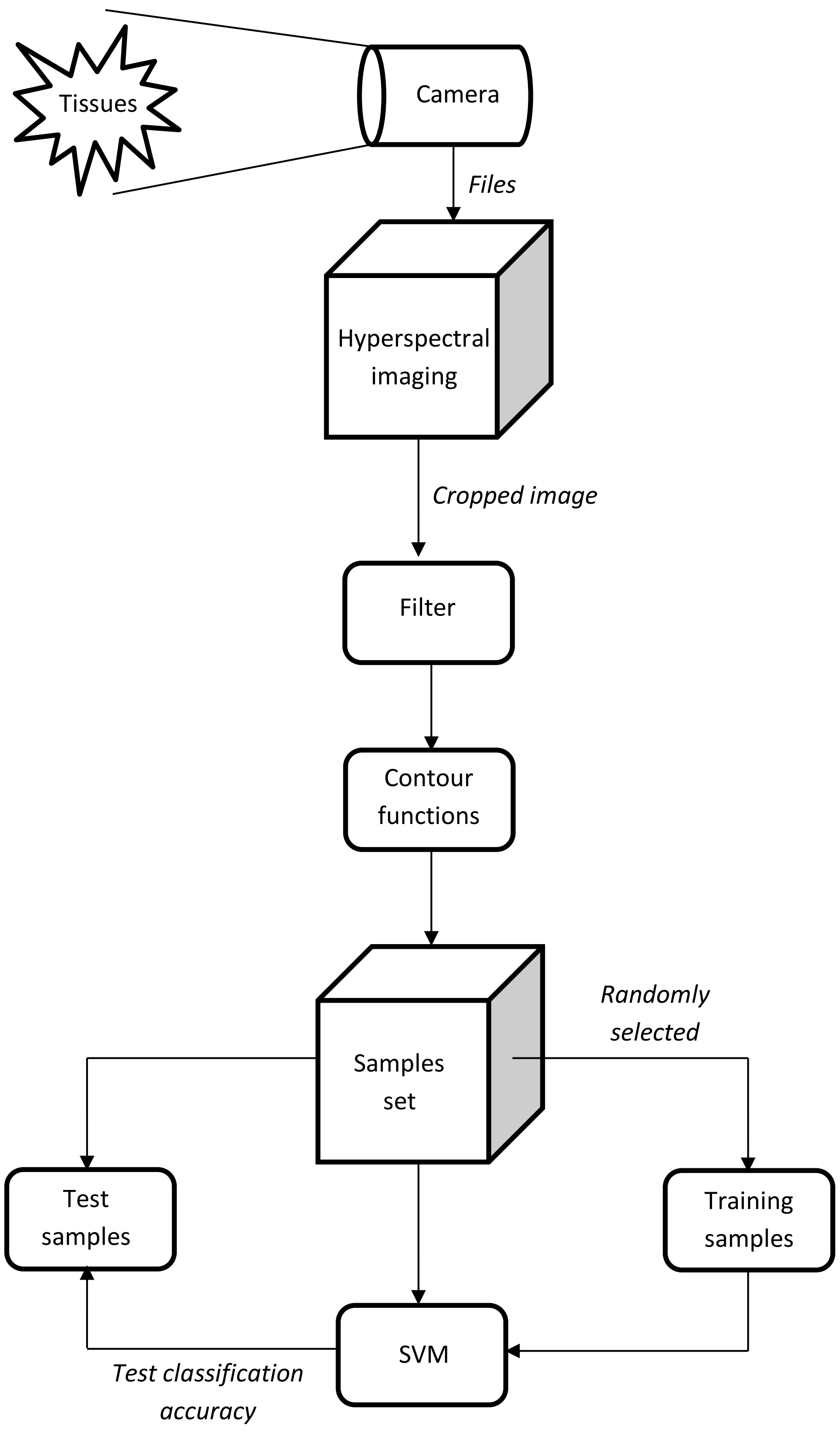


*Figure 4.1: General flowchart of the processing chain #1.*

## 4.2. Process #2: Dimensionality reduction techniques used in brain cancer detection

The chain chosen in this process is based on dimensionality reduction from the original input data as a preliminary step to classification. We have to keep in mind that that this block is a process of transformation and/or reducing the dimensionality of the original image. The input data are pre-processed where it seeks to obtain the information with less correlation and redundancy of the image and grouping it by linear combinations in a subset of bands. The dimensionality reduction stage is formed by two sub stages: feature extraction and feature selection, in which several combinations of different techniques will be tested.

The process can be summarized in figure 4.2:

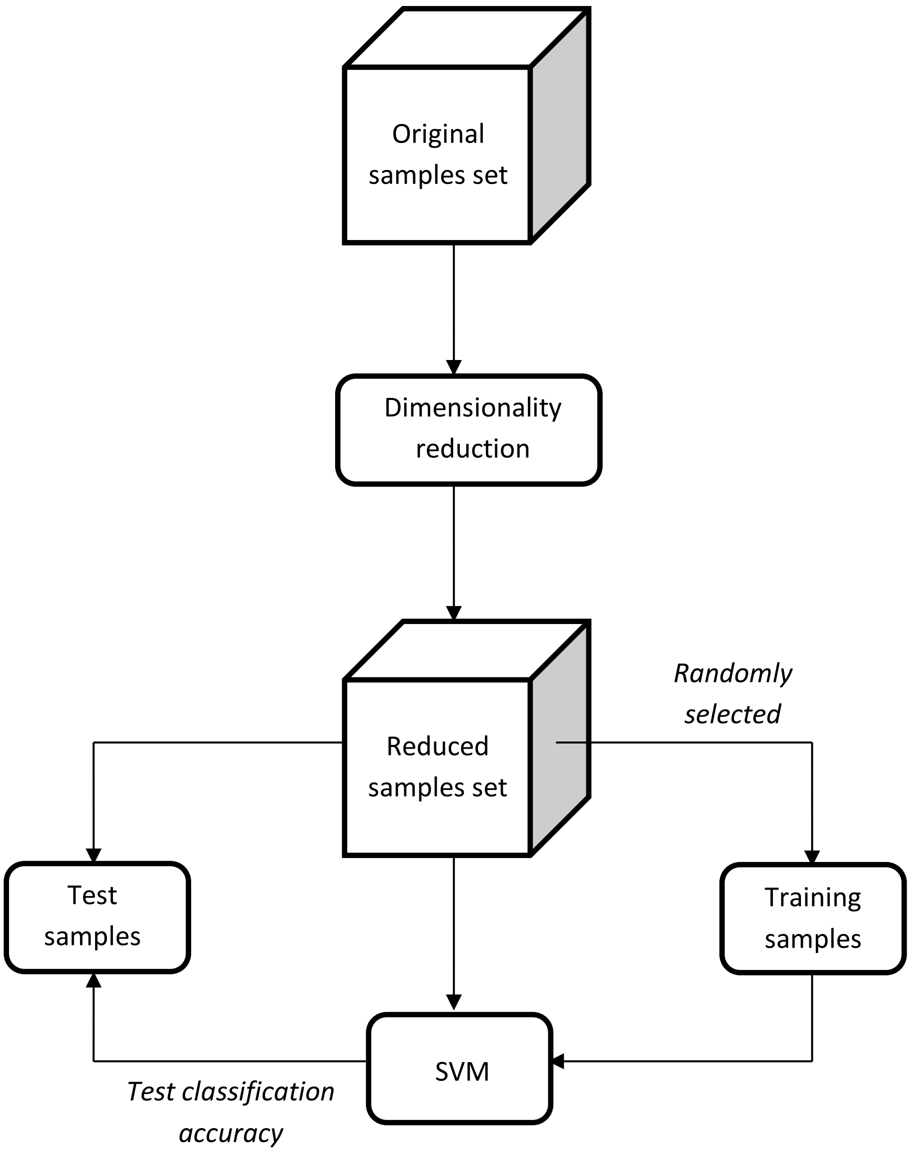


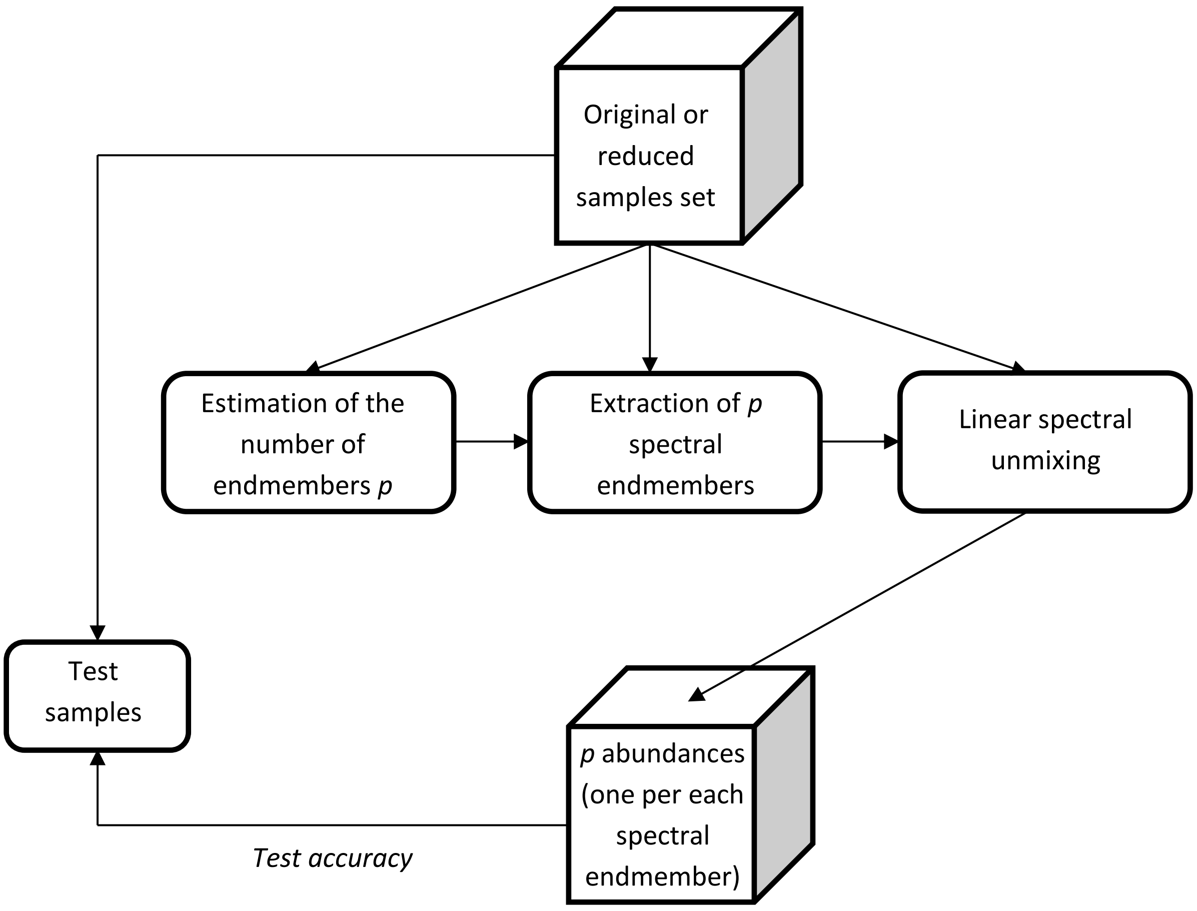
*Figure 4.2: General flowchart of the processing chain #2.*

## 4.3. Process #3: Spectral unmixing concepts used in brain cancer detection

In this process spectral unmixing techniques are applied. It is composed by 3 steps: estimation of the number of endmembers, extraction of endmembers and abundances estimation. In this dissertation the number of endmember is always known a priori (number of classes). Regarding to the extraction of endmembers, several techniques will be applied and even it will be manually extracted in several different ways. Finally for abundances estimation linear spectral unmixing is used. In addition to the combination between the different sub stages, we used two dataset as input data: the original samples set and the reduced samples set. The reduced samples set chosen was the one that best results offered in the process chain #2.

The unmixing results is an alternative representation of the data in which each pixel is shown as a set of abundances (one per each spectral endmember) representing the proportion of each class present in each pixel. These dataset can be interpreted by the same data based on the abundances obtained, so this block will not have a classification stage.

The diagram used in this processing chain is shown in figure 4.3:



Original or reduced samples set
Estimation of the number of endmembers *p*
Extraction of *p* spectral endmembers
Linear spectral unmixing
Test samples
*p* abundances (one per each spectral endmember)
*Test accuracy*

*Figure 4.3: General flowchart of the processing chain #3.*

## 4.4. Process #4: Mixed techniques Unmixing-SVM used in brain cancer detection

This process is very similar to the previous one. The procedure and combination tested are the same as in the previous chain, but in this case, the resulting abundances from applying unmixing concepts are introduced as input data to the SVM classifier, as it is able to correctly interpret these data. Then, the main objective for this processing chain is to combine the power and advantages of both techniques: unmixing and classification.

The diagram used in this processing chain is shown in figure 4.4:

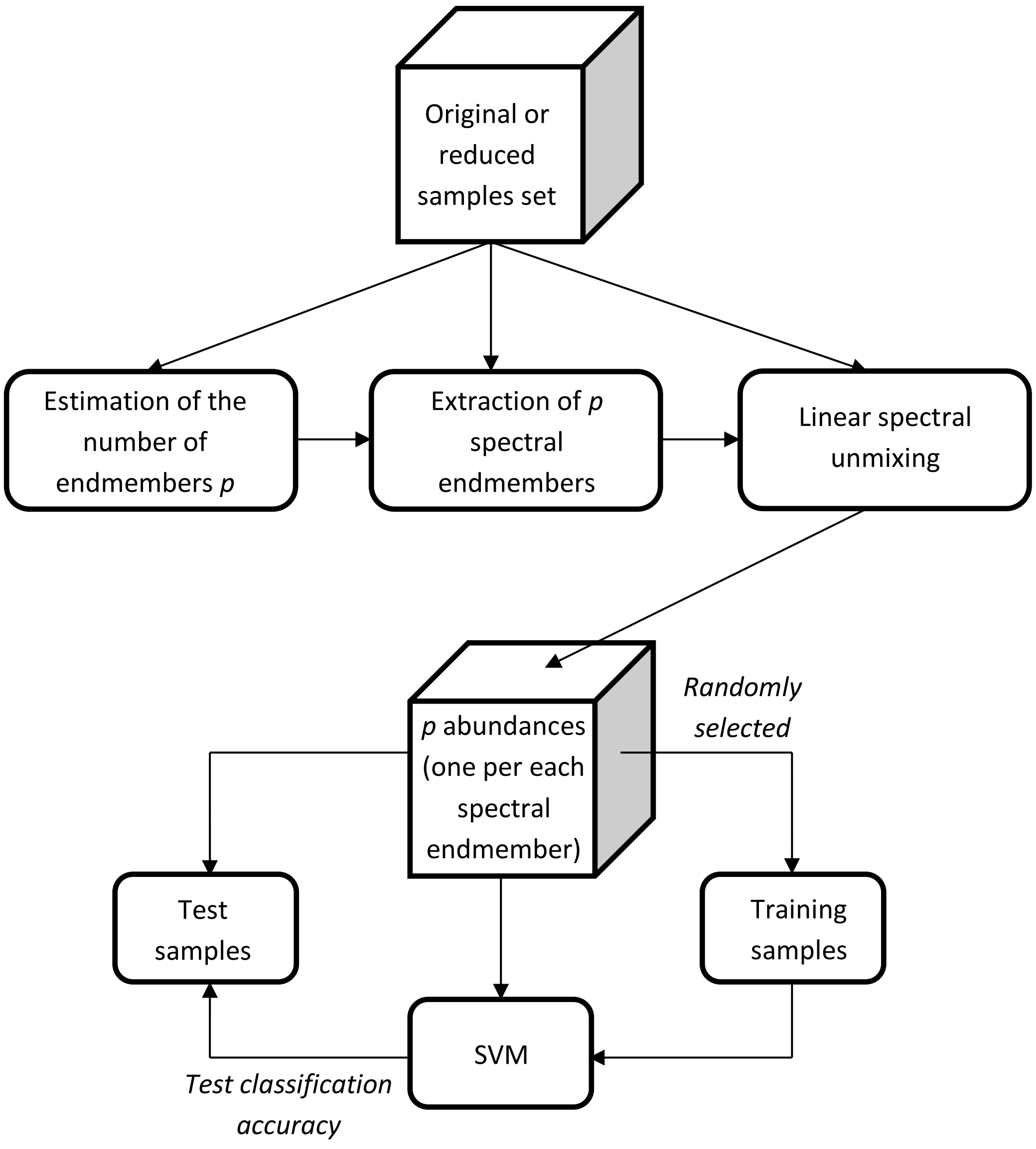

Original or reduced samples set
Estimation of the number of endmembers *p*
Extraction of *p* spectral endmembers
Linear spectral unmixing
*p* abundances (one per each spectral endmember)
*Randomly selected*
Test samples
Training samples
SVM
*Test classification accuracy*

*Figure 4.4: General flowchart of the processing chain #4.*

## 4.5. Process #5: Different methods for extracting training set using SVM in brain cancer detection

This process is focused on determining the best way to extract the set of training samples for classification. In this regard, three methods were tested in order to improve the results obtained:

- **Random training:** The training set is randomly selected.
- **Guided training:** The training set is chosen following several selection criteria.
- **Mixed random-guided training:** Meant to be a commitment to the training selection criteria between the two previous selection processes.

As input data we use the best possible combination of the different techniques used so far, i.e., data sets generated by processing chains that best results offered. SVM classifier is used at the end of the processing chain in order to measure the accuracy of training extraction techniques proposals.

Moreover, it is intended to use this processing chain to evaluate how training size affects the classification results. For this, the size of the training set will be progressively increased.

The diagram used in this processing chain is shown below in figure 4.5:

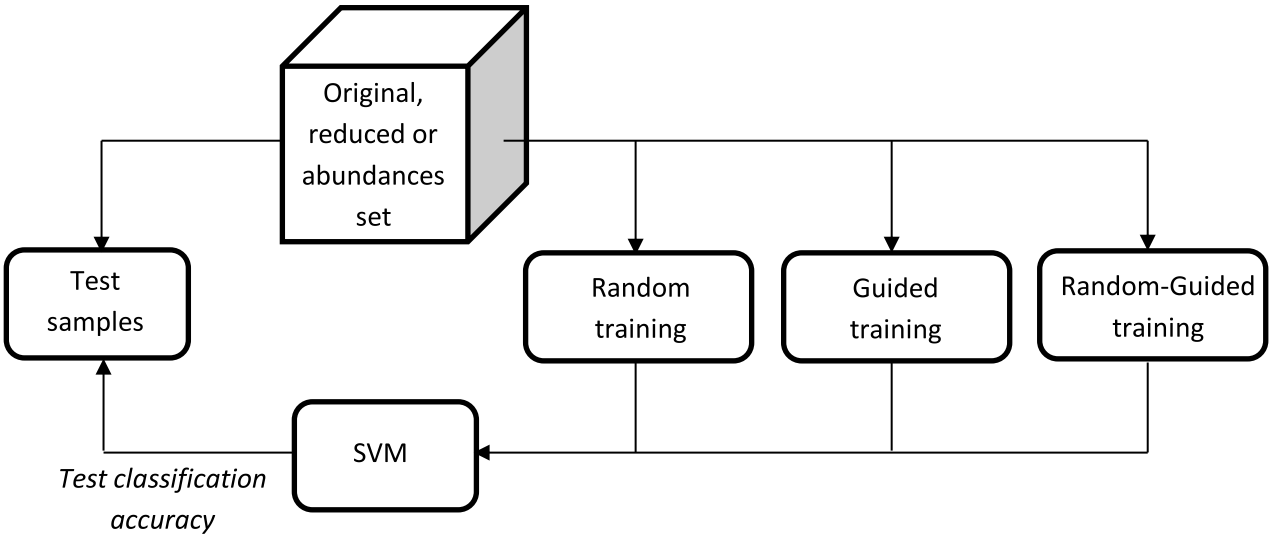


*Figure 4.5: General flowchart of the processing chain #5.*

## 4.6. Process #6: Patient simulation using SVM in brain cancer detection

This process is based on dividing into two subsets of samples the *ex vivo* dataset as a final step before starting to work with *in vivo* samples, in order to simulate a real surgery. This is because in surgeries it is proposed to work on a database generated from generic spectral signatures obtained from different patients, so that a patient is classified using as training other patients from the database. Therefore by dividing the *ex vivo* dataset in two subsets is intended to simulate this situation, a subset of samples is classified obtaining the training set from the other subset of samples.

The SVM classifier works extracting the training set from the data set that is to be classified. The SVM algorithm must be modified to work classifying a dataset and extracting the training of a different dataset. This modification is carried out in this chain to perform these tests and it will also be required to work with *in vivo* samples in which a patient is classified and the training is obtained from other patients.

The subsets are totally independent, i.e., once separated, these are not joined at any time and these are divided into different sizes to perform several tests. Once again, in this processing chain the data sets from previous processing chains which obtained the best results are used.

The diagram used in this processing chain is shown below in figure 4.6:

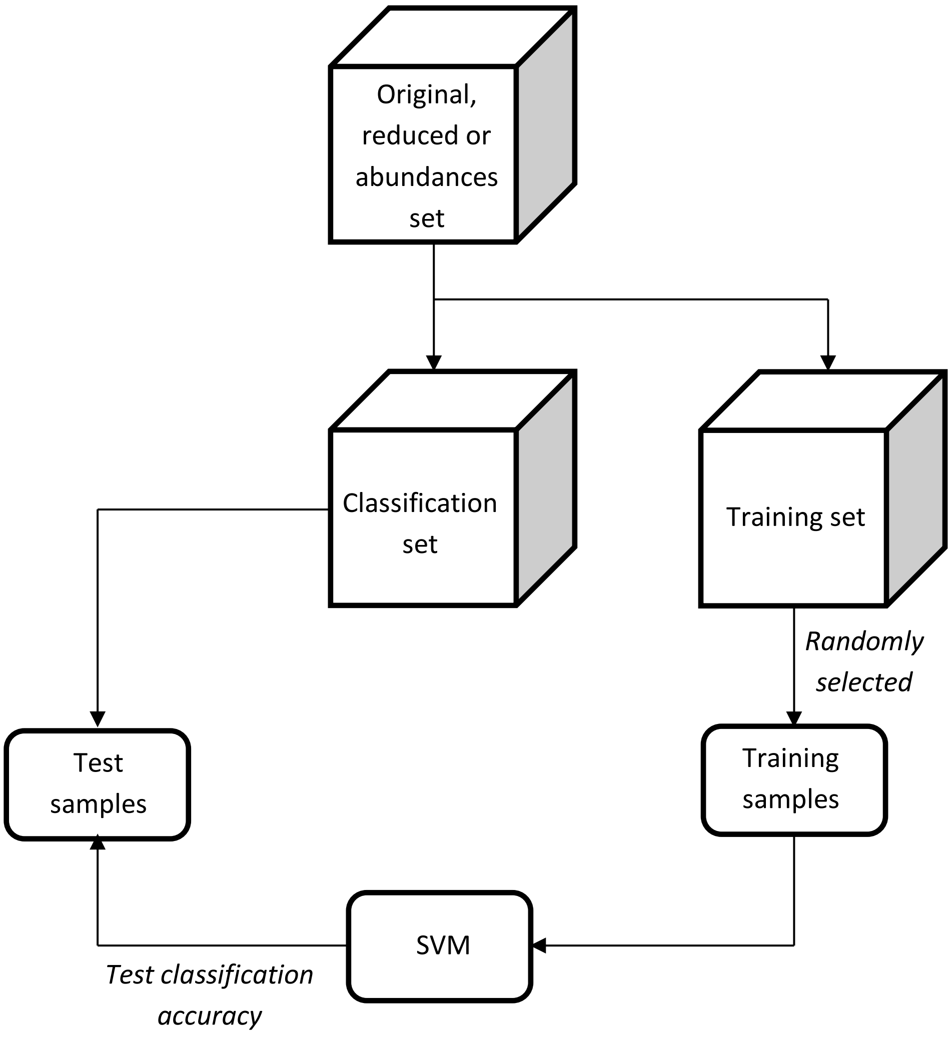


*Figure 4.6: General flowchart of the processing chain #6.*

## 4.7. Process #7: Consolidation of processing chains tested using *in vivo* data in brain cancer detection

In this latest proposed process is intended to test the best combinations of techniques used in previous processing chains for *ex vivo* data now using *in vivo* data, checking how the change of data will affect the results.

For working on this chain we have 5 different patients listed as 4, 5, 7, 8 and 10 referring to the number of surgical intervention. First, each patient is individually classified using the best combinations of techniques obtained in the previous processing chains.

After that, we perform various combinations of patients, so we have patients to classify and patients to extract the trainings. In this regard several different tests will be performed in order to obtain the best procedure when establishing the training set facing real situations and always using the best processing chains obtained so far.

In the case in which we use 2 groups of patients, one for classifying and one for training, these two datasets are independent and never are joined. For this reason, it is not possible to apply dimensionality reduction, because the transformation depends on the data set which is applied, then to apply the same technique of dimensionality reduction to two independent datasets generates that the data are transformed to different domains and these cannot be compared among them. For applying dimensional reduction techniques to two independent data sets would be necessary to transform one of the data sets and use the eigenvectors and eigenvalues obtained for transforming the second data set, so that both sets of data are transformed to the same domain.

The diagram used in this processing chain is shown in figure 4.7:

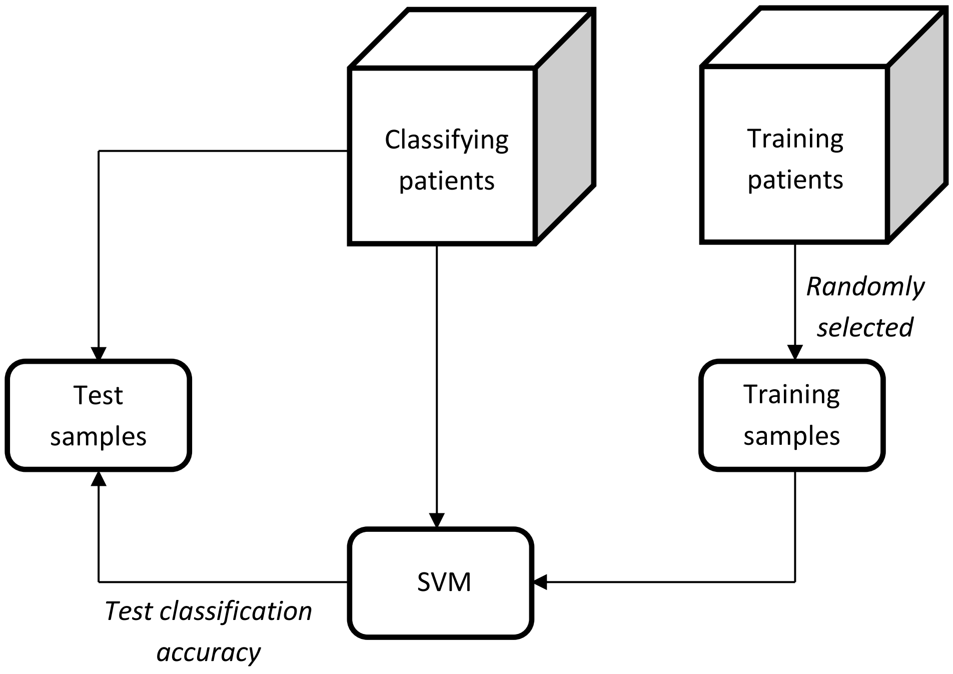


*Figure 4.7: General flowchart of the processing chain #7.*

## 4.8. Summary

In this section the different process developed in this dissertation are presented. All the processing chains are exhaustively tested using *ex vivo* samples.

In summary, based on the results we should make the following decisions:

1. To decide whether to filter the data using the proposed filters or not to filter the data.
2. To decide whether to process the data using the dimensionality reduction techniques, unmixing techniques or a combination of both, or not to process the data.
3. To decide how to be extracted the training sets: randomly, guided or a combination of both.

Once the best options are set, the resulting chains are tested with *in vivo* samples.

# Chapter 5
# Experimental results

In this chapter we conducted an experimental evaluation of the different process of hyperspectral data described in the previous sections. Processing chains are evaluated according to different criteria, such as accuracy in the classification results obtained or its computational time.

The main difficulty associated with thematic classification techniques is that usually there are several possible classes associated with different targets. The goal is ultimately to determine the presence or absence of each of the targets considered in each pixel, situation that can be expressed as a binary classification problem which can subsequently be extended to any number of classes. Several techniques are developed in order to evaluate the classification process in hyperspectral images. The general process is illustrated by a simple diagram in figure 5.1. As shown in this figure, the training process consists in using a subset of this information to train the supervised classifier (training patterns) and evaluate performance of the classifier with the rest of patterns labeled (test patterns), as it is shown in figure 5.1.

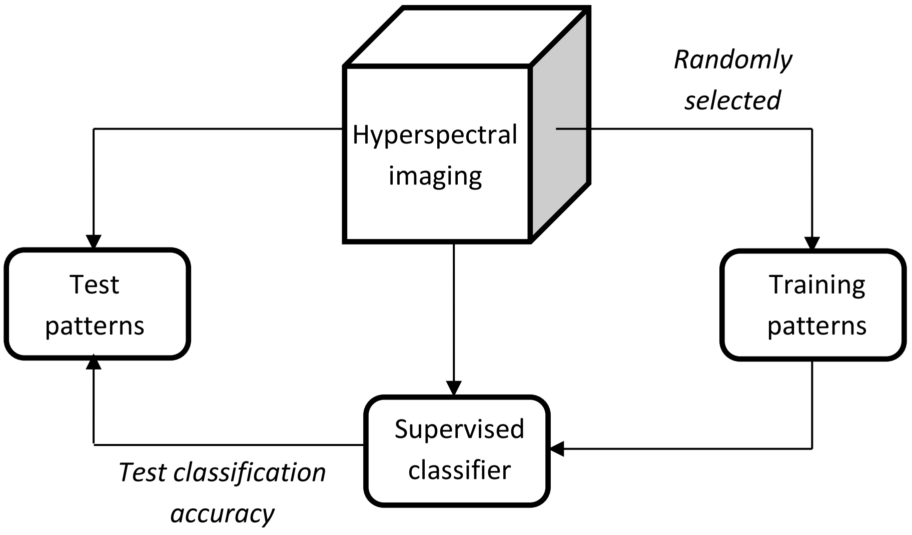


*Figure 5.1: General process of  a supervised system.*

Although in this project are used guided trainings, the distinction between training patterns and test patterns is generally performed randomly, trying to minimize the maximum number of training patterns necessary to achieve a satisfactory result of classification with the rest of samples, mainly because it is often difficult to obtain training patterns in this kind of applications. Due to the high dimensionality of the original data is attempted to compensate the need for a large number of samples using (optionally) data dimensionality reduction techniques. Moreover, due to the presence of mixed pixels in the hyperspectral imaging is attempted to solve the problem by applying (optionally) unmixing techniques. Therefore, throughout this chapter is intended to validate the combination of a set of techniques of dimensionality reduction and unmixing with a supervised classifier SVM.

As for the sets of hyperspectral imaging selected to perform the experiments, these have been fully described in the corresponding section 2.2.3. However, it is worth noting that in this dissertation we have used two sets of samples: *ex vivo* and *in vivo*. *Ex vivo* samples are extracted from a real surgery after being removed from the patient, these are used to test all possible processing chains. When the best possible processing chains are established, *in vivo* samples obtained directly during the surgeries are used to check the performance in real situations of selected processing chains.

## 5.1. Results of processing chains

In this section we describe the results obtained by applying the different process presented in the previous chapter on the different hyperspectral imaging considered (*ex vivo* and *in vivo*), using ground-truth (this term refers to the accuracy of the training set's classification for supervised learning techniques) information to statistically validate the classification results obtained by applying the analysis methodology and evaluation metrics described above. Once the results obtained are presented separately for each processing chain, we proceed to discuss the results globally previously to the end of this chapter.

It should be recalled that in all processing chains, except in which spectral unmixing techniques are applied, a supervised classifier SVM is used, so that the information that varies from one processing chain to another is mainly based on the preprocessing stage used.

The classification is done by assigning a specific class label to each pixel once the learning phase of SVM classifier is completed, in which involves a number of training patterns selected from labeled pixels in the image from ground-truth information.

Followed we present a detailed quantitative and comparative study analyzing the results obtained by different processing chains of hyperspectral imaging used.

### 5.1.1. Results of support vector machine classifier used in brain cancer detection

In this experiment, we use the *ex vivo* samples data sets to analyze the possible kernels available in the SVM and the need to filter the datasets to remove noise from the samples prior to classification. For this processing chain we generate ten training sets by randomly selecting 3%, 5% and 10% of the ground-truth pixels.

Then, the three considered types of input samples (original, HySime filter and smooth filter) are built for the selected training samples and used to train an SVM classifier in which two types of kernels: linear and radial basis function (RBF) are used. The SVM was trained with each of these training subsets and then evaluated with the remaining test set. Each experiment was repeated ten times, and the mean and standard deviation of the different evaluation metrics values were reported. Kernel parameters were automatically optimized and selected by the internal function of SVM. The rest of kernels offered by SVM were tested also, but linear and RBF kernels were selected for being the most representative.

The followings tables summarizes the evaluation metrics obtained after applying the considered SVM classification system (with linear and radial basis function (RBF) kernels) to the set of samples obtained after applying the filters considered in the process #1 (see 4.1) to the *ex vivo* samples.

To refer to the filters used and the classes present, the nomenclature used is the same as that set forth previously in the section 3.1.2.

| **AA** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 88.08% (0.0101) | 88.76% (0.0079) | 89.35% (0.0065) | 94.90% (0.0047) | 95.61% (0.0034) | 96.65% (0.0011) | 96.57% (0.0037) | 97.14% (0.0030) | **98.27% (0.0018)** |
| **RBF** | 77.27% (0.0259) | 83.84% (0.0196) | 90.78% (0.0119) | 79.31% (0.0370) | 87.65% (0.0177) | 92.56% (0.0187) | 82.28% (0.0268) | 89.73% (0.0199) | 95.71% (0.0127) |
| **OA** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 85.76% (0.0119) | 86.59% (0.0082) | 87.56% (0.0089) | 93.53% (0.0066) | 94.65% (0.0050) | 95.91% (0.0014) | 95.41% (0.0048) | 96.23% (0.0039) | **97.55% (0.0028)** |
| **RBF** | 74.21% (0.0272) | 80.90% (0.0212) | 88.93% (0.0143) | 77.10% (0.0405) | 85.15% (0.0216) | 91.08% (0.0199) | 79.16% (0.0263) | 87.75% (0.0220) | 94.50% (0.0152) |

*Table 5.1: Classification accuracies (percentage) and standard deviation obtained after applying the considered SVM classification system (with linear and RBF kernels) to three different types of input datasets (original, HySime filter and smooth filter) extracted from the ex vivo samples (ten randomly chosen training sets).*

| **Sensitivity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 92.02% (0.0110) | 92.99% (0.0168) | 94.72% (0.0086) | 96.92% (0.0119) | 97.30% (0.0077) | 98.19% (0.0045) | 98.75% (0.0044) | 98.82% (0.0057) | **99.73% (0.0015)** |
| **RBF** | 75.95% (0.0585) | 83.17% (0.0226) | 90.80% (0.0238) | 78.83% (0.0651) | 87.21% (0.0351) | 92.30% (0.0284) | 81.25% (0.0404) | 90.25% (0.0335) | 96.53% (0.0116) |
| **Specificity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 98.91% (0.0025) | 99.07% (0.0021) | 99.18% (0.0013) | 99.64% (0.0011) | 99.79% (0) | 99.85% (0) | 99.87% (0.0012) | 99.98% (0) | **99.98% (0)** |
| **RBF** | 93.78% (0.0107) | 96.40% (0.0113) | 98.50% (0.0029) | 94.43% (0.0183) | 97.80% (0.0052) | 98.85% (0.0045) | 95.96% (0.0133) | 98.37% (0.0050) | 99.57% (0.0022) |

*Table 5.2: Sensitivity and specificity for class 1 (percentage) and standard deviation obtained after applying the considered SVM classification system (with linear and RBF kernels) to three different types of input datasets (original, HySime filter and smooth filter) extracted from the ex vivo samples (ten randomly chosen training sets).*

| **Sensitivity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 94.22% (0.0078) | 94.67% (0.0074) | 95.10% (0.0035) | 97.97% (0.0021) | 98.18% (0.0046) | 98.69% (0.0011) | 98.86% (0.0028) | 99.08% (0.0029) | **99.57% (0.0013)** |
| **RBF** | 88.10% (0.0124) | 92.51% (0.0123) | 96.11% (0.0059) | 88.95% (0.0219) | 94.58% (0.0098) | 97.00% (0.0100) | 91.58% (0.0185) | 95.56% (0.0093) | 98.60% (0.0052) |
| **Specificity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 77.35% (0.0169) | 78.19% (0.0134) | 78.95% (0.0145) | 88.32% (0.0125) | 90.27% (0.0139) | 92.08% (0.0034) | 91.03% (0.0094) | 92.45% (0.0094) | **94.59% (0.0070)** |
| **RBF** | 64.66% (0.0287) | 71.40% (0.0250) | 81.04% (0.0214) | 68.37% (0.0459) | 76.59% (0.0283) | 84.39% (0.0292) | 69.45% (0.0288) | 79.86% (0.0305) | 89.27% (0.0255) |

*Table 5.3: Sensitivity and specificity for class 2 (percentage) and standard deviation obtained after applying the considered SVM classification system (with linear and RBF kernels) to three different types of input datasets (original, HySime filter and smooth filter) extracted from the ex vivo samples (ten randomly chosen training sets).*

| **Sensitivity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 61.86% (0.0257) | 62.60% (0.0259) | 62.23% (0.0223) | 78.37% (0.0216) | 81.74% (0.0259) | 84.38% (0.0093) | 81.77% (0.0188) | 84.47% (0.0191) | **87.85% (0.0155)** |
| **RBF** | 49.81% (0.0291) | 56.86% (0.0304) | 68.79% (0.0246) | 53.69% (0.0407) | 63.29% (0.0287) | 73.78% (0.0372) | 54.42% (0.0380) | 66.63% (0.0371) | 79.61% (0.0445) |
| **Specificity** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 96.64% (0.0048) | 96.86% (0.0054) | 97.05% (0.0026) | 98.82% (0.0017) | 98.85% (0.0033) | 99.16% (0) | 99.31% (0.0015) | 99.37% (0.0020) | **99.71% (0)** |
| **RBF** | 94.07% (0.0113) | 96.29% (0.0059) | 98.03% (0.0036) | 94.45% (0.0102) | 97.17% (0.0048) | 98.41% (0.0054) | 95.84% (0.0082) | 97.67% (0.0067) | 99.20% (0.0030) |

*Table 5.4*: *Sensitivity and specificity for class 3 (percentage) and standard deviation obtained after applying the considered SVM classification system (with linear and RBF kernels) to three different types of input datasets (original, HySime filter and smooth filter) extracted from the ex vivo samples (ten randomly chosen training sets).*

| **Kappa** | **No filter** | | | **Filter 1** | | | **Filter 2** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Kernel** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** | **TR. 3%** | **TR. 5%** | **TR. 10%** |
| **Linear** | 0.7623 (0.0185) | 0.7742 (0.0127) | 0.7858 (0.0141) | 0.8893 (0.0109) | 0.9073 (0.0081) | 0.9273 (0.0024) | 0.9204 (0.0080) | 0.9339 (0.0066) | 0.9558 (0.0049) |
| **RBF** | 0.5829 (0.0399) | 0.6849 (0.0321) | 0.8086 (0.0231) | 0.6250 (0.0618) | 0.7521 (0.0336) | 0.8448 (0.0335) | 0.6587 (0.0410) | 0.7920 (0.0355) | 0.9024 (0.0260) |

*Table 5.5*: *Kappa coefficient and standard deviation obtained after applying the considered SVM classification system (with linear and RBF kernels) to three different types of input datasets (original, HySime filter and smooth filter) extracted from the ex vivo samples (ten randomly chosen training sets).*

As shown by the tables, the classification results are correlated with the training set size (the larger the training set, the better the classification results and the computing times). The good generalization ability exhibited by SVM is demonstrated by the classification results reported for the original spectral information, even with very limited training sets. These results can be improved in this chain by two different ways: kernels used and filtering samples. On the one hand, regarding the kernels used we show a comparison between linear kernel and RBF kernel, from which we can see that the linear kernel performs better than RBF for this particular application. In addition to the better results offered by linear kernel, this is much faster than RBF kernel in terms of computing time. On the other hand, about whether or not filter the samples we can discern the need for a filtering stage. This is because the samples are very noisy due to the conditions in which these were obtained. More specifically, it has been found that the smooth filter offers better results than the HySime filter for this type of data.

As a conclusion of this processing chain, we have that the best kernel which we can use is the linear kernel because it offers the best results. Moreover, we see that the data are necessary to be filtered in order to further improve the results obtained. Regarding the training set, we have chosen the training formed by 10% of total samples because this provides optimal performance in a compromise between results obtained, computing time and size. For this reason, in the rest of processing chains we have used linear kernel in SVM classifier, smooth filter to filtering samples and the training formed by 10% of the ground-truth pixels because this combination offers the best results for this chain. In future processing chains this combination of parameters may be not mentioned as its use is obviated.

### 5.1.2. Results of dimensionality reduction techniques used in brain cancer detection

In this experiment we apply the process #2 for dimensionality reduction prior to classification. Due to the results obtained by the process #1, in this process chain the smooth filter, linear kernel and training formed by 10% of total of samples are used.

First step in dimensionality reduction stage is the feature selection. In this regard we have used two techniques to estimate the optimal number of bands or features depending on the dataset used, in this case *ex vivo* samples filtered by smooth. These techniques are: Virtual dimensionality and HySime, which are described in its relevant sections. The results obtained regarding the optimal number of features are shown in the table 5.6:

| **Technique** | **Virtual Dimensionality** | | | | | **V.D. mean** | **HySime** |
|---|---|---|---|---|---|---|---|
| **Features** | 21 | 18 | 18 | 17 | 17 | 18.20 | 87 |

*Table 5.6: Optimal number of features to ex vivo samples filtered by smooth.*

As we can see from the table 5.6, five similar results were obtained with virtual dimensionality, whose average values are also shown in this table. Conversely, the HySime technique provides as results a very remote value with respect to the other five obtained by virtual dimensionality. For this reason, HySime value has been discarded and we have selected 18 and 21 as optimum values for the number of features for being respectively the average value and the largest value obtained by virtual dimensionality.

In table 5.7 to table 5.12 the results of applying the feature extraction step of dimensionality reduction stage are shown. In the tables, the name of the techniques used is accompanied by the number of features selected, namely 18 or 21. In addition we can see a first column of results called *original* which refers to the results obtained in the best case of the process chain #1, i.e., without additional preprocessing stages. This nomenclature will be used in future sections. Moreover, the computing time is also show in order to compare and find the quickest way to perform the classification.

Due to its size, the results obtained have been divided into three different tables in which the main values are highlighted. In the first table we show the three techniques based on principal components analysis: PCA, MNF and ICA.

| **Dimensionality reduction** | **Original** | **PCA 18** | **PCA 21** | **MNF 18 (ENVI)** | **MNF 21 (ENVI)** | **ICA 18** | **ICA 21** |
|---|---|---|---|---|---|---|---|
| **AA** | **98.27% (0.0018)** | 94.16% (0.0059) | 94.48% (0.0057) | 96.61% (0.0017) | **96.87% (0.0016)** | 94.36% (0.0060) | **94.78% (0.0054)** |
| **OA** | **97.55% (0.0028)** | 92.52% (0.0086) | 92.84% (0.0108) | 95.84% (0.0027) | **96.06% (0.0024)** | 92.93% (0.0093) | **93.34% (0.0080)** |
| **Kappa** | **0.9558 (0.0049)** | 0.8687 (0.0143) | 0.8744 (0.0176) | 0.9255 (0.0046) | **0.9295 (0.0040)** | 0.8755 (0.0154) | **0.8826 (0.0134)** |
| **Time (sec)** | 203.6258 | 75.3976 | 76.6932 | - | **-** | **8.2881** | 9.5911 |

*Table 5.7: Results of applying PCA, MNF and ICA techniques for dimensionality reduction prior to classification.*

| Sensitivity | Original | PCA 18 | PCA 21 | MNF 18 (ENVI) | MNF 21 (ENVI) | ICA 18 | ICA 21 |
|---|---|---|---|---|---|---|---|
| **Class 1** | **99.73% (0.0015)** | 98.83 % (0.0043) | 99.14% (0.0053) | 98.95% (0.0040) | 99.14% (0.0054) | 99.05% (0.0032) | **99.38% (0.0016)** |
| **Class 2** | **99.57% (0.0013)** | 97.98% (0.0033) | 98.15% (0.0034) | 98.76% (0.0021) | **98.94% (0.0023)** | 97.97% (0.0022) | **98.19% (0.0022)** |
| **Class 3** | **87.85% (0.0155)** | 70.64% (0.0282) | 71.44% (0.0385) | 82.45% (0.0152) | **82.87% (0.0154)** | 71.91% (0.0313) | **72.79% (0.0266)** |
| **Specificity** | **Original** | **PCA 18** | **PCA 21** | **MNF 18 (ENVI)** | **MNF 21 (ENVI)** | **ICA 18** | **ICA 21** |
| **Class 1** | **99.98% (0)** | 99.87% (0) | 99.91% (0) | 99.93% (0) | **99.95% (0)** | 99.92% (0) | **99.93% (0)** |
| **Class 2** | **94.59% (0.0070)** | 85.58% (0.0164) | 86.09% (0.0220) | 91.73% (0.0079) | **92.02% (0.0077)** | 86.39% (0.0178) | **86.99% (0.0150)** |
| **Class 3** | **99.71% (0)** | 98.66% (0.0022) | 98.75% (0.0022) | 99.16% (0.0016) | **99.27% (0.0016)** | 98.62% (0.0016) | **98.77% (0.0015)** |

*Table 5.8: Results of applying PCA, MNF and ICA techniques for dimensionality reduction prior to classification.*

In the second table we show the two techniques based on nearest neighbor: LPP and NPE.

| Dimensionality reduction | Original | LPP 18 | LPP 21 | NPE 18 | NPE 21 |
|---|---|---|---|---|---|
| **AA** | **98.27% (0.0018)** | 67.26% (0.0206) | 71.06% (0.0216) | 80.14% (0.0265) | **85.12% (0.0210)** |
| **OA** | **97.55% (0.0028)** | 75.83% (0.0201) | 79.24% (0.0211) | 76.75% (0.0208) | **87.11% (0.0159)** |
| **Kappa** | **0.9558 (0.0049)** | 0.5257 (0.0374) | 0.5928 (0.0410) | 0.6080 (0.0354) | **0.7648 (0.0300)** |
| **Time (sec)** | 203.6258 | **60.3011** | 62.6412 | 245.2735 | 241.4797 |

*Table 5.9: Results of applying LPP and NPE techniques for dimensionality reduction prior to classification.*

| Sensitivity | Original | LPP 18 | LPP 21 | NPE 18 | NPE 21 |
|---|---|---|---|---|---|
| **Class 1** | 99.73% (0.0015) | 94.91% (0.0451) | 95.87% (0.0208) | 99.59% (0.0032) | **99.81% (0.0024)** |
| **Class 2** | **99.57% (0.0013)** | 76.47% (0.0171) | 78.98% (0.0171) | 87.31% (0.0283) | 90.30% (0.0203) |
| **Class 3** | **87.85% (0.0155)** | 59.31% (0.0525) | 65.84% (0.0396) | 47.13% (0.0248) | 66.88% (0.0313) |
| **Specificity** | **Original** | **LPP 18** | **LPP 21** | **NPE 18** | **NPE 21** |
| **Class 1** | **99.98% (0)** | 85.38% (0.0198) | 87.33% (0.0211) | 90.61% (0.0248) | 92.97% (0.0184) |
| **Class 2** | **94.59% (0.0070)** | 75.56% (0.0504) | 81.02% (0.0398) | 66.29% (0.0240) | 84.77% (0.0227) |
| **Class 3** | **99.71% (0)** | 91.62% (0.0082) | 92.48% (0.0078) | 98.46% (0.0049) | 97.42% (0.0060) |

*Table 5.10: Results of applying LPP and NPE techniques for dimensionality reduction prior to classification.*

The techniques shown in the previous tables are linear techniques. In this third table 5.11 the results for non-linear technique KPCA using linear and polynomial kernels are shown.

| Dimensionality reduction | Original | KPCA (linear) 18 | KPCA (linear) 21 | KPCA (polynomial) 18 | KPCA (polynomial) 21 |
|---|---|---|---|---|---|
| **AA** | **98.27% (0.0018)** | 94.15% (0.0057) | **94.52% (0.0058)** | 74.06% (0.0673) | 76.72% (0.0577) |
| **OA** | **97.55% (0.0028)** | 92.51% (0.0082) | **92.88% (0.0107)** | 75.28% (0.0991) | 72.25% (0.1273) |
| **Kappa** | **0.9558 (0.0049)** | 0.8686 (0.0137) | **0.8750 (0.0175)** | 0.5806 (0.1385) | 0.5714 (0.1521) |
| **Time (sec)** | **203.6258** | 1.7454e+004 | 1.7450e+004 | 1.9825e+004 | 2.3759e+004 |

*Table 5.11: Results of applying KPCA technique for dimensionality reduction prior to classification.*

| Sensitivity | Original | KPCA (linear) 18 | KPCA (linear) 21 | KPCA (polynomial) 18 | KPCA (polynomial) 21 |
|---|---|---|---|---|---|
| **Class 1** | 99.73% (0.0015) | 98.83% (0.0043) | 99.14% (0.0053) | 89.53% (0.1) | 98.65% (0.0058) |
| **Class 2** | **99.57% (0.0013)** | 97.97% (0.0032) | 98.17% (0.0032) | 83.51% (0.0506) | 88.64% (0.0436) |
| **Class 3** | **87.85% (0.0155)** | 70.61% (0.0270) | 71.52% (0.0381) | 42.53% (0.1349) | 37.45% (0.1555) |
| **Specificity** | **Original** | **KPCA (linear) 18** | **KPCA (linear) 21** | **KPCA (polynomial) 18** | **KPCA (polynomial) 21** |
| **Class 1** | **99.98% (0)** | 99.87% (0) | 99.91% (0) | 96.60% (0.0393) | 99.66% (0.0017) |
| **Class 2** | **94.59% (0.0070)** | 85.56% (0.0157) | 86.13% (0.0216) | 66.89% (0.1174) | 63.44% (0.1473) |
| **Class 3** | **99.71% (0)** | 98.65% (0.0021) | **98.77% (0.0021)** | 90.69% (0.0251) | 92.82% (0.0316) |

*Table 5.12: Results of applying KPCA technique for dimensionality reduction prior to classification.*

To summarize, the results obtained are reported in the bar graphs of figure 5.2 to figure 5.9:

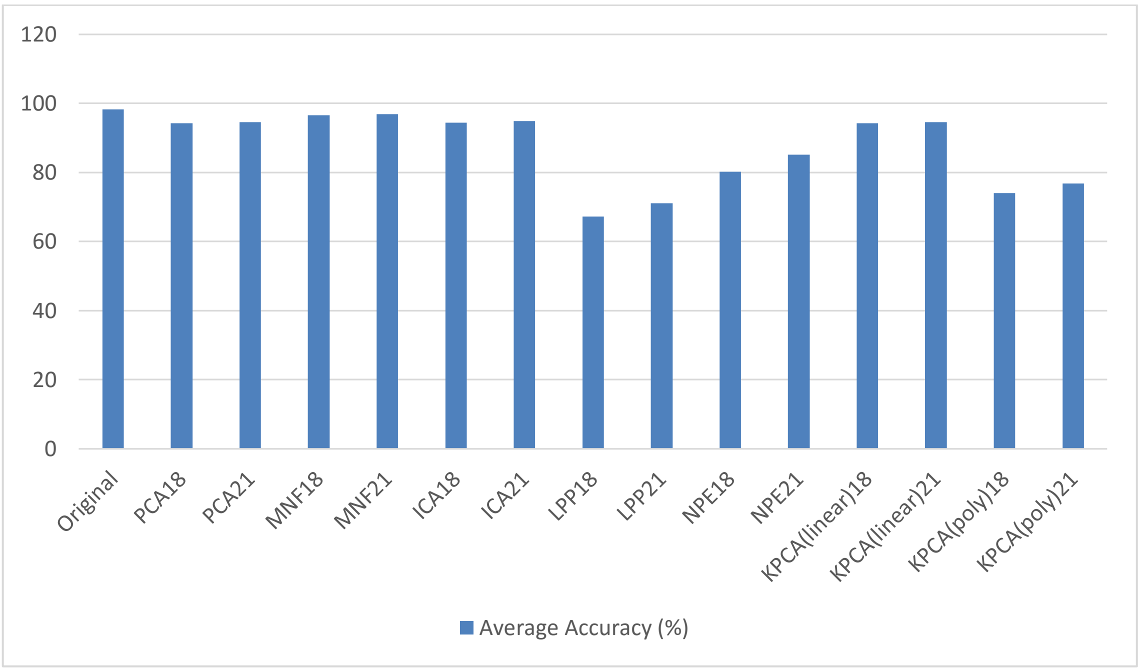


*Figure 5.2: Average Accuracy for the different dimensionality reduction techniques applied.*

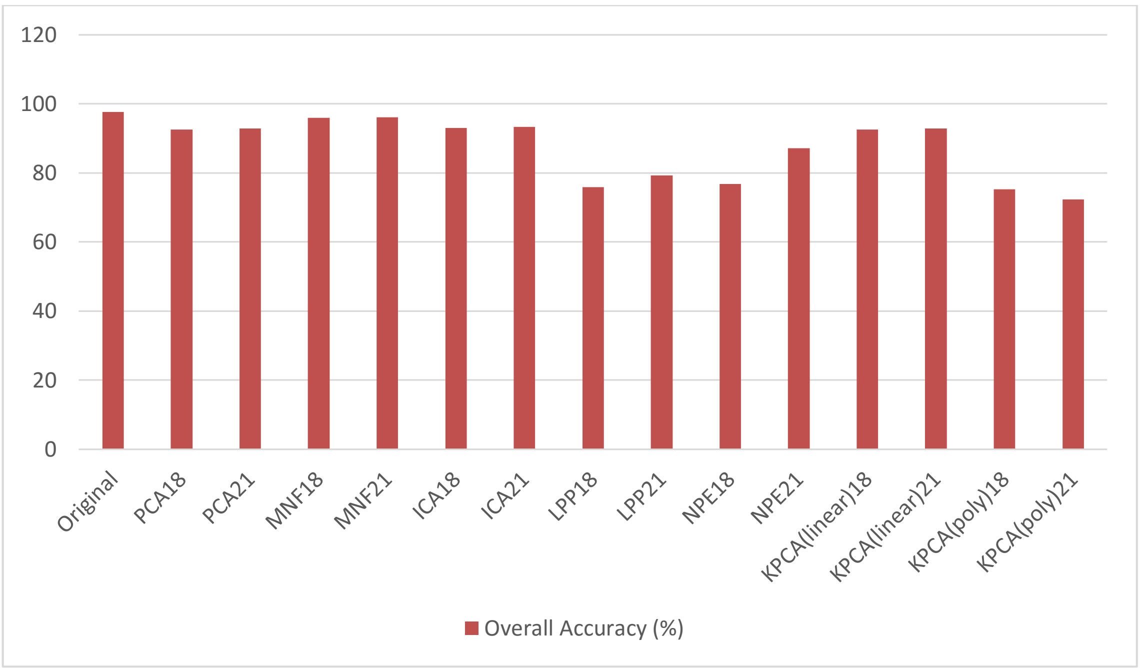


*Figure 5.3: Overall Accuracy for the different dimensionality reduction techniques applied.*

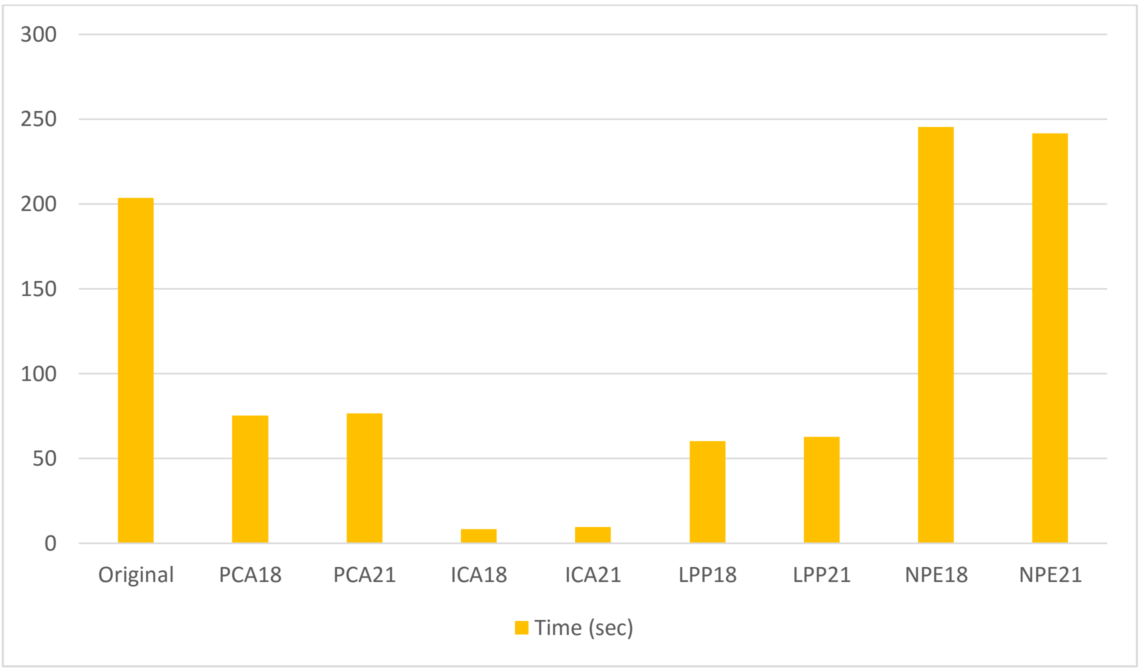


*Figure 5.4: Computing time for the different dimensionality reduction techniques applied.*

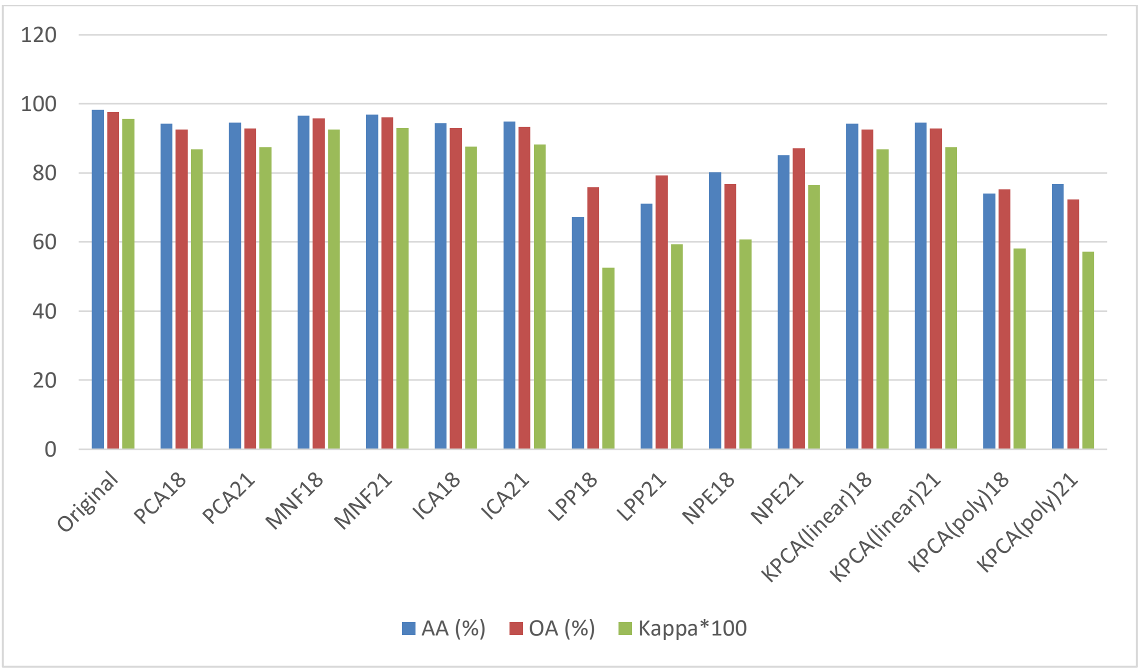


*Figure 5.5: AA, OA and Kappa for the different dimensionality reduction techniques applied.*

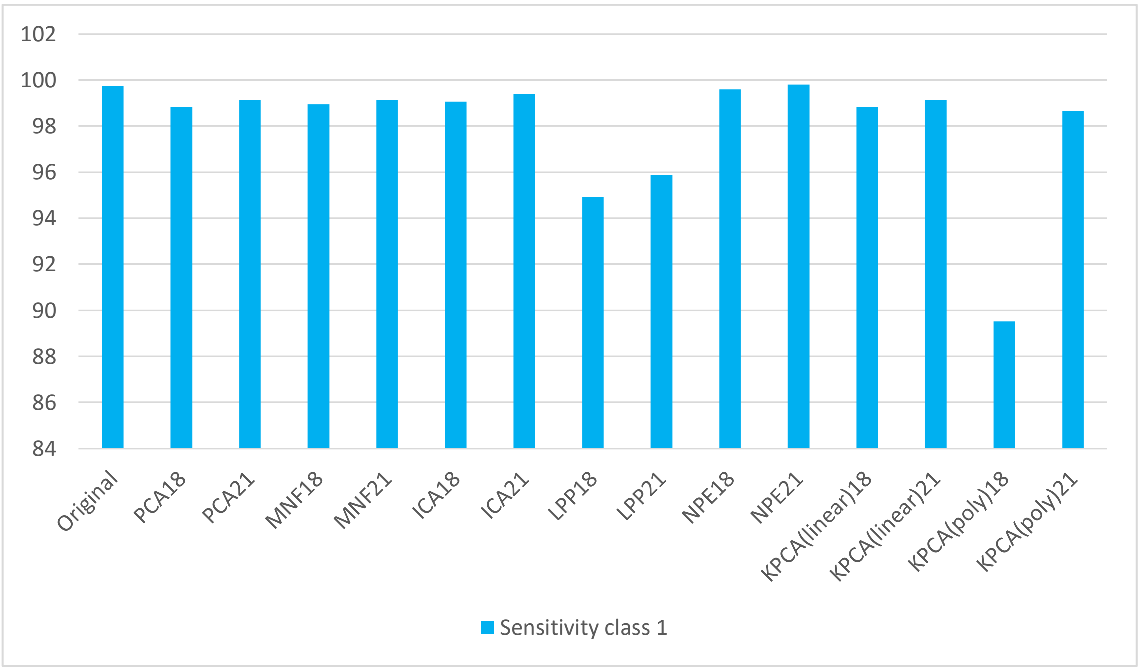


*Figure 5.6: Sensitivity of class 1 for the different dimensionality reduction techniques applied.*

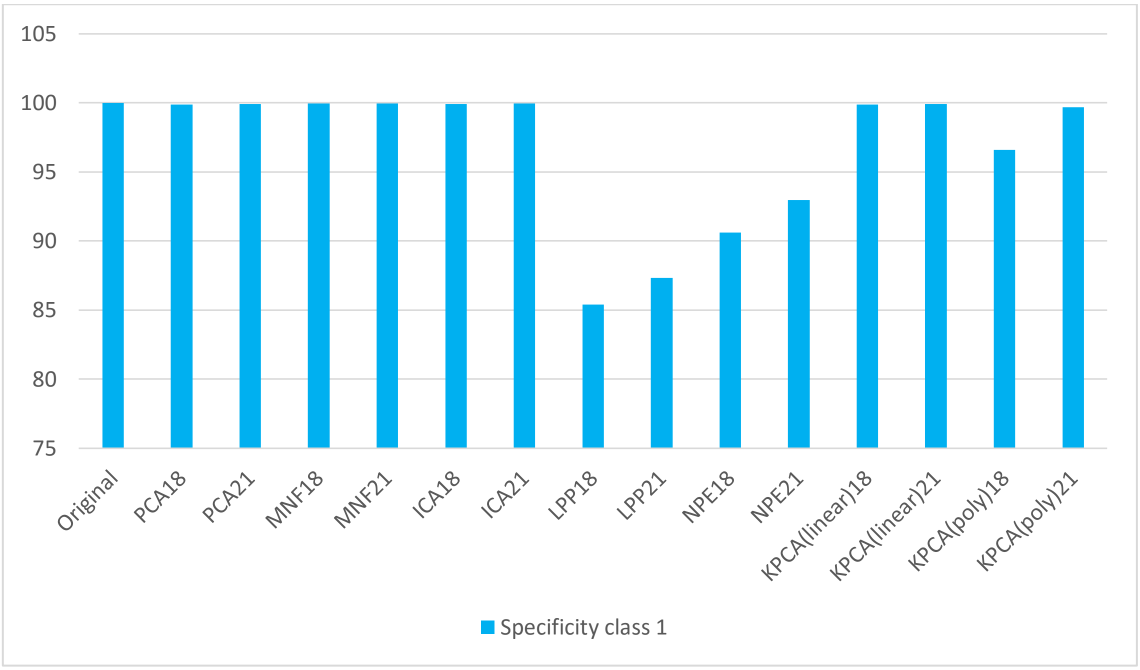


*Figure 5.7: Specificity of class 1 for the different dimensionality reduction techniques applied.*

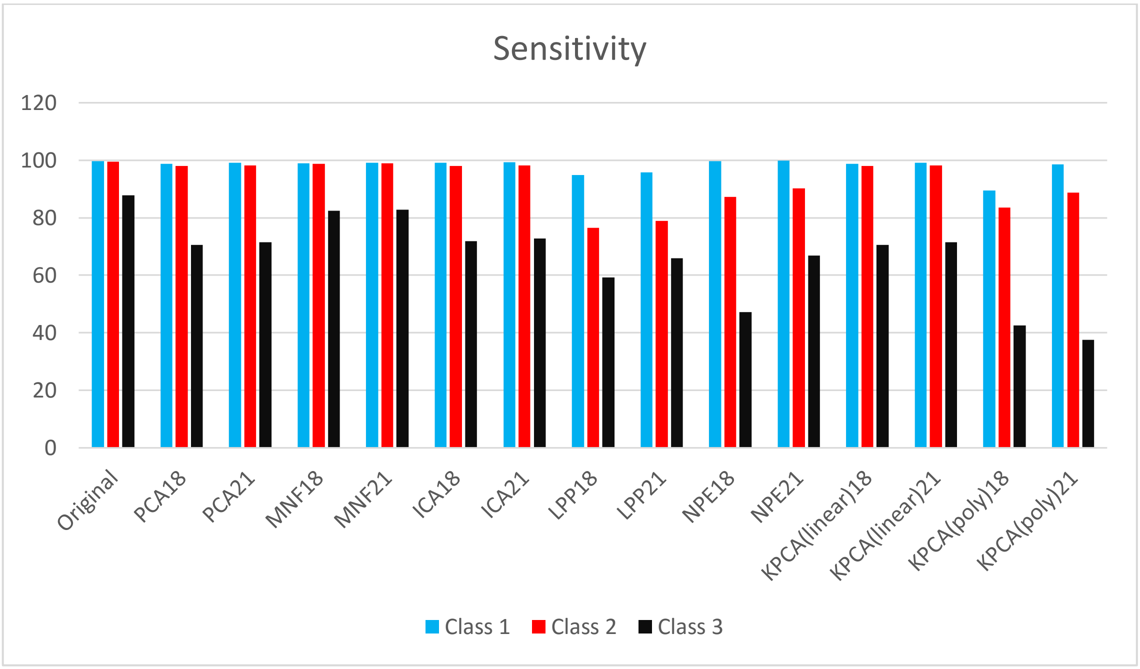


*Figure 5.8: Sensitivity for the different dimensionality reduction techniques applied.*

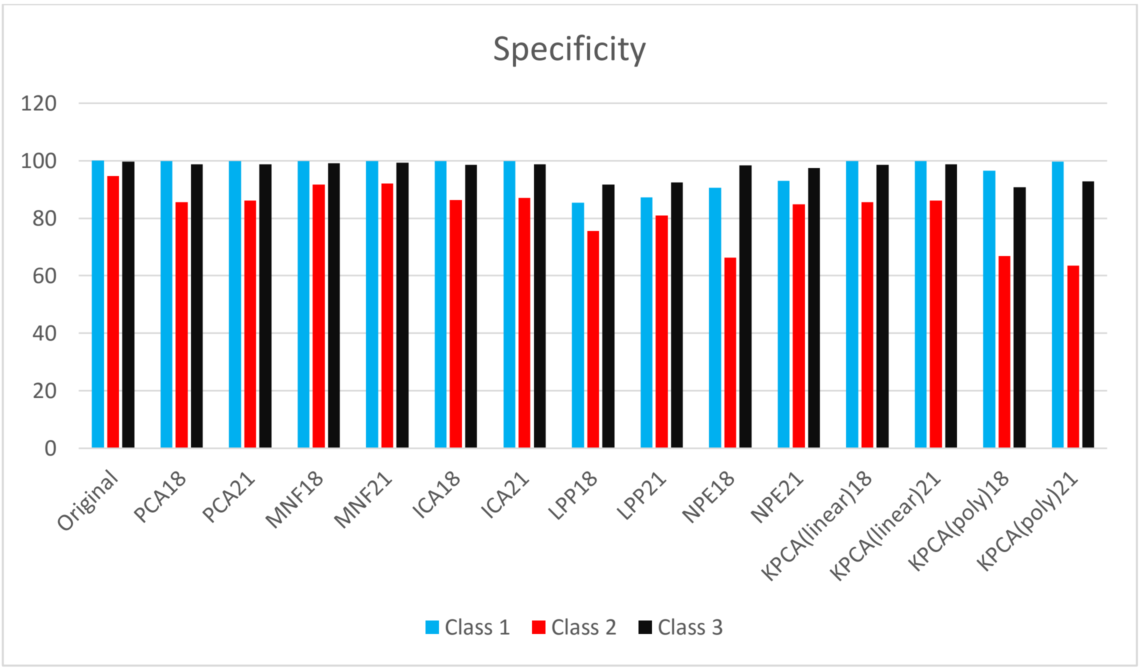


*Figure 5.9: Sensitivity for the different dimensionality reduction techniques applied.*

As shown in the above tables and graphs, the best results applying dimensionality reduction is obtained by MNF. Nevertheless, MNF technique was applied using the ENVI software. This is proprietary software and it cannot be included in this dissertation because the ULPGC has not purchase a license.

Therefore, the ICA technique is the best option in terms of dimensionality reduction because this offers very competitive results and reduces the computational cost.

However, the original information provides the better accuracy results in comparison to the rest of the chain but the computational cost increases significantly.

An alternative way of applying dimensionality reduction is a special version from smooth, which is called smooth reduction filter in this dissertation. This version was developed by a coworker in the department and allows us to reduce the number of bands of the original dataset similar to dimensionality reduction. In this regards, we apply the following process chain recommended by smooth reduction filter developer:

1. Denoising the samples using HySime filter.
2. Apply smooth reduction filter to reduce to 115 bands.
3. Implement a dimensionality reduction stage.
4. Classify the dataset.

Once the dataset is obtained after applying the smooth reduction filter, the feature selection step is applied, obtaining the results shown in the table 5.13:

| **Technique** | **Virtual Dimensionality** | | | | | **V.D. mean** | **HySime** |
|---|---|---|---|---|---|---|---|
| **Features** | 22 | 21 | 20 | 20 | 18 | 20.20 | 26 |

*Table 5.13: Optimal number of features to samples filtered by smooth reduction.*

In this case, the values obtained from different techniques are similar, so we have chosen the average value obtained by virtual dimensionality technique and the value obtained by HySime, namely 20 and 26.

In the table 5.14 the results obtained of applying smooth reduction filter prior to dimensionality reduction stage are shown. By way of comparison, in the firs column of results in this table, the results obtained of classifying directly the dataset without dimensionality reduction stage are shown.

| **Smooth reduction filter** | **Smooth reduction 115** | **PCA 20** | **PCA 26** | **ICA 20** | **ICA 26** |
|---|---|---|---|---|---|
| **AA** | **95.39% (0.0033)** | 92.46% (0.0039) | 93.50% (0.0036) | 92.54% (0.0045) | 93.55% (0.0043) |
| **OA** | **94.33% (0.0044)** | 90.74% (0.0046) | 92.18% (0.0049) | 90.92% (0.0043) | 92.34% (0.0050) |
| **Kappa** | **0.8999 (0.0075)** | 0.8393 (0.0074) | 0.8633 (0.0080) | 0.8422 (0.0072) | 0.8658 (0.0083) |
| **Time (sec)** | 57.7579 | 51.7009 | **51.3960** | 89.4440 | 63.0582 |

*Table 5.14*: *Results of applying smooth reduction filter prior dimensionality reduction and classification.*

To summarize, the results obtained are reported in the following bar graphs, from figure 5.10 to figure 5.12:

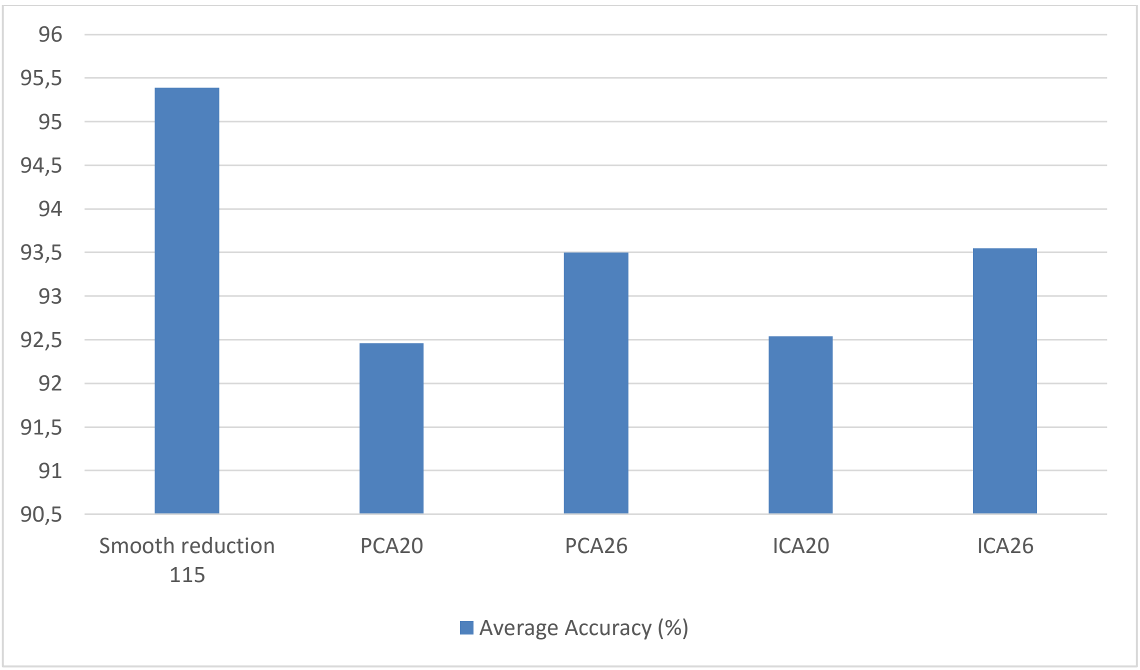


*Figure 5.10: Average Accuracy for the application of smooth reduction filter prior dimensionality reduction stage.*

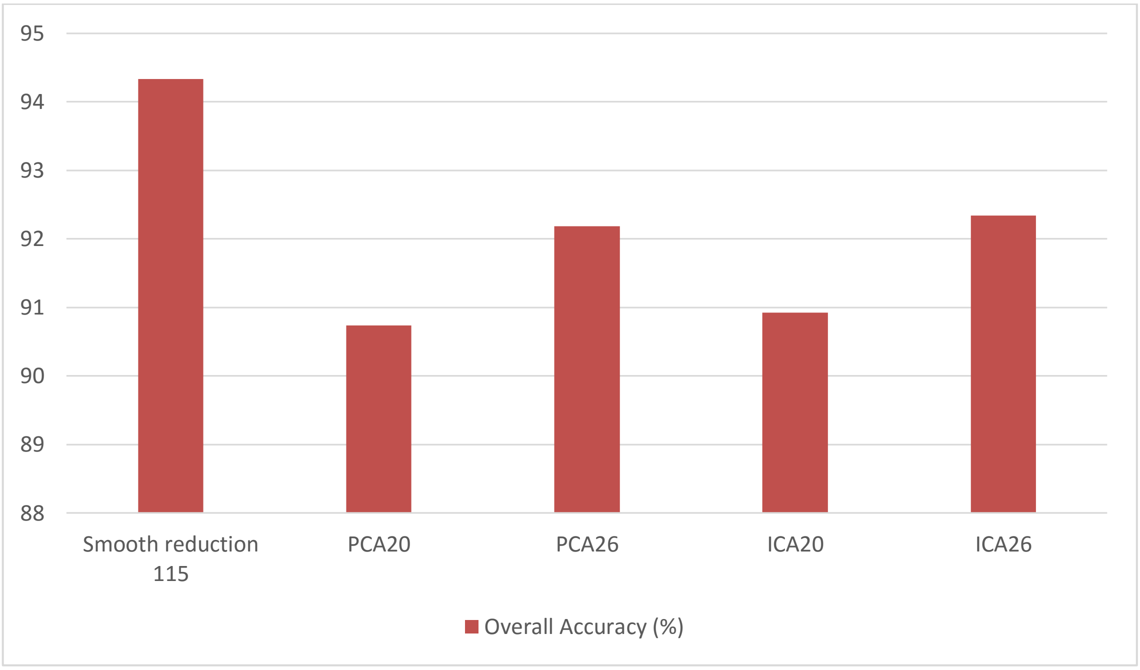


*Figure 5.11: Overall Accuracy for the application of smooth reduction filter prior dimensionality reduction stage.*

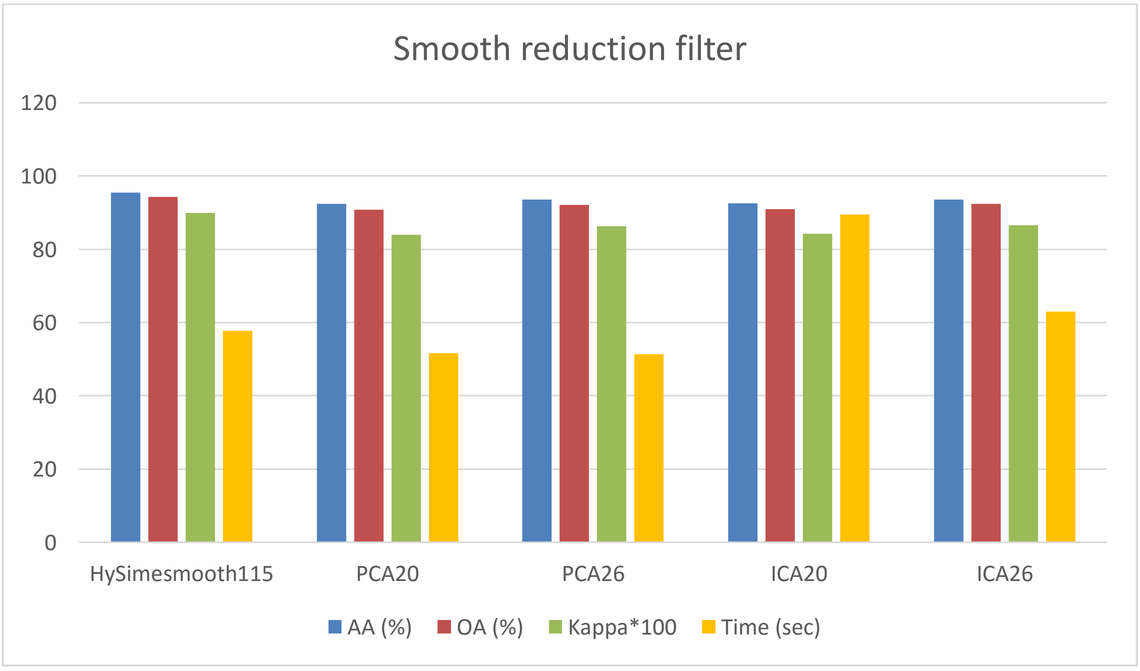


*Figure 5.12: Results obtained for the application of smooth reduction filter prior dimensionality reduction stage.*

As in the first results, in this case the best results obtained were without dimensionality reduction stage. In this regard, we proceeded to check the rest of available filter combinations trying to improve results. The combinations proposed were:

- Combination 1: First apply HySime filter and then apply smooth reduction filter to reduce to 115 bands (as in the previous case).
- Combination 2: Directly apply the smooth reduction filter to reduce to 115 bands.
- Combination 3: First apply original smooth filter and then apply smooth reduction filter to reduce to 115 bands.

The results obtained for these tests are shown in the followings tables and bar graphs:

| **Filtering order** | **Combination 1** | **Combination 2** | **Combination 3** |
|---|---|---|---|
| **AA** | 95.39% (0.0033) | 90.73% (0.0035) | **96.96% (0.0020)** |
| **OA** | 94.33% (0.0044) | 89.02% (0.0064) | **96.08% (0.0022)** |
| **Kappa** | 0.8999 (0.0075) | 0.8097 (0.0097) | **0.9298 (0.0039)** |
| **Time (sec)** | 57.7579 | 105.1719 | **44.7789** |

*Table 5.15: Results of applying the different filter combinations prior classification.*

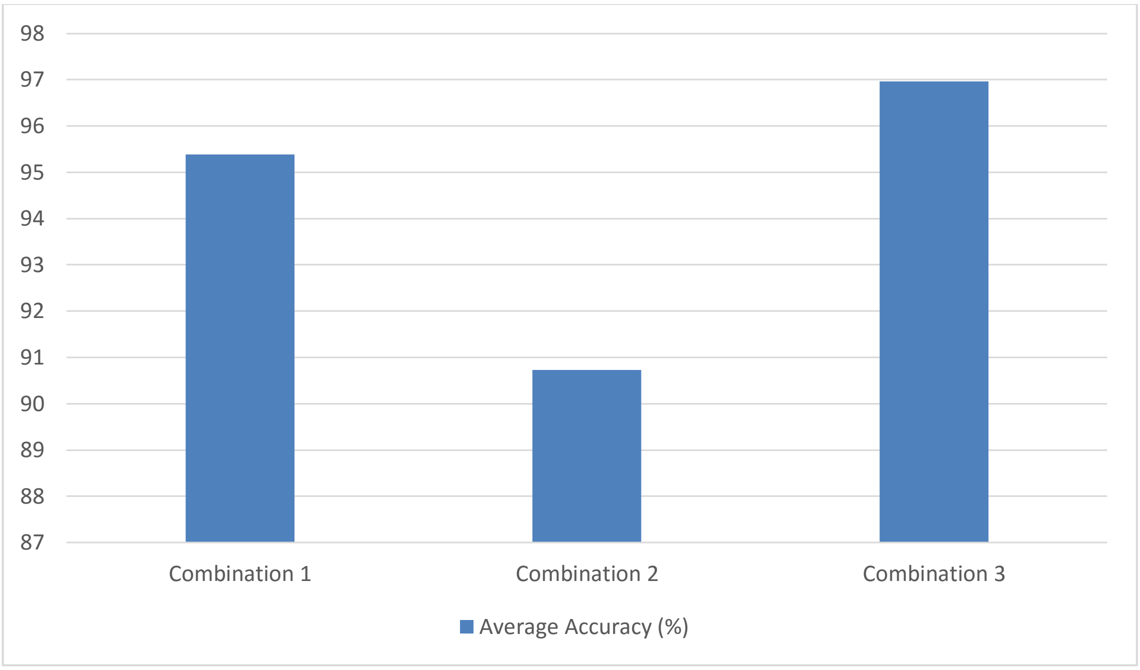


*Figure 5.13: Average Accuracy for the different filter combinations prior classification.*

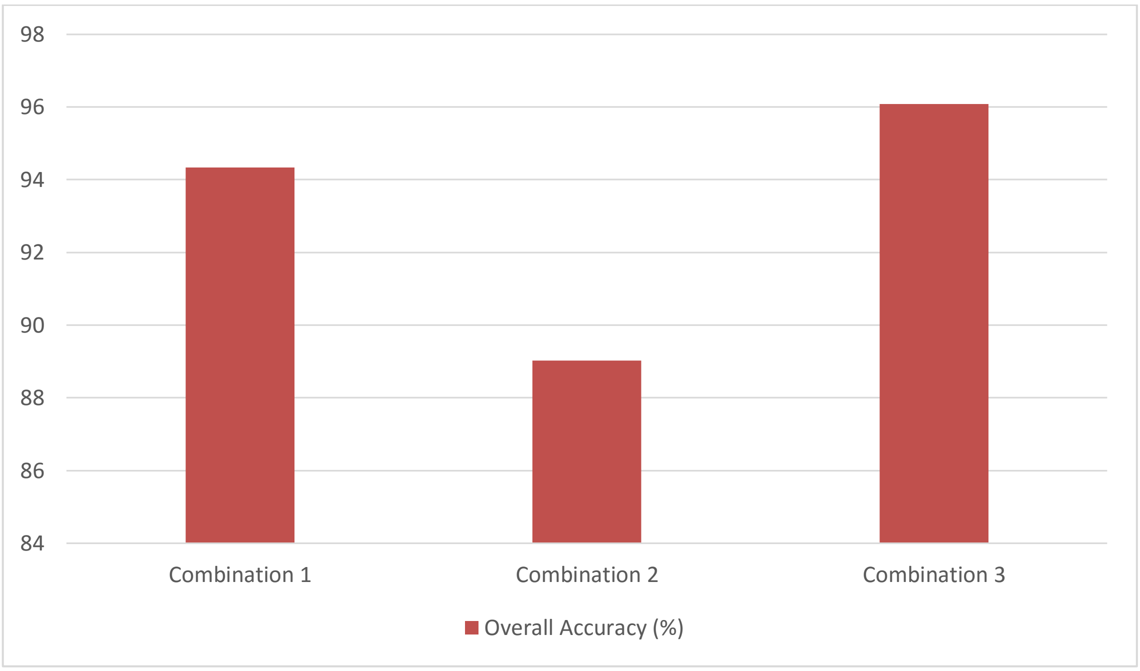


*Figure 5.14*: *Overall Accuracy for the different filter combinations prior classification.*

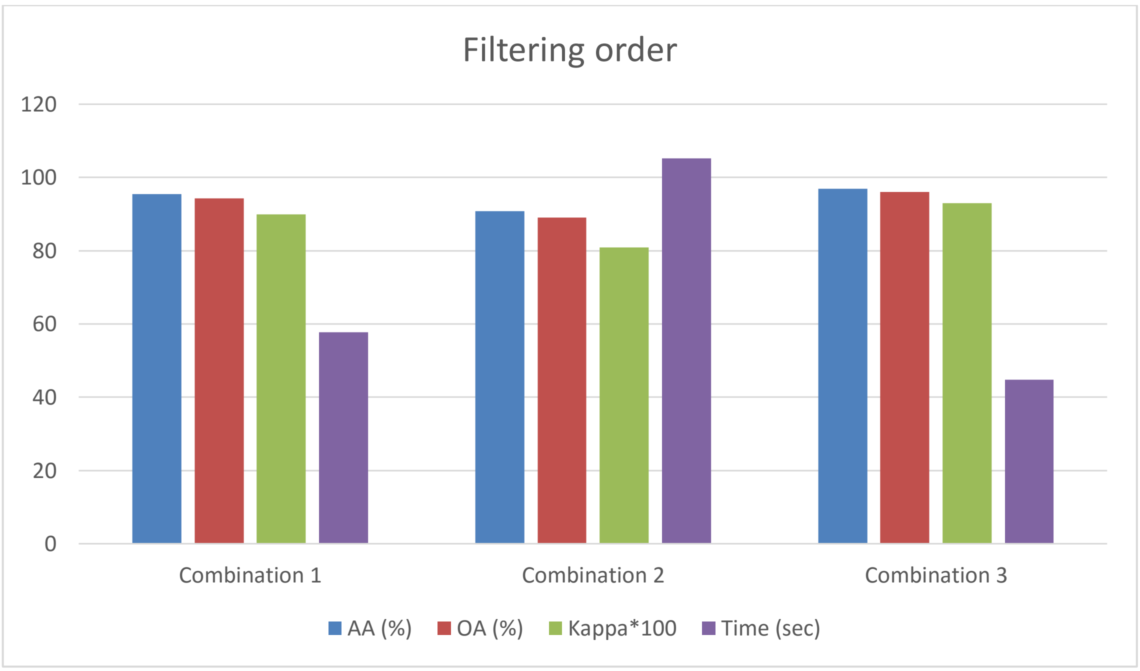


*Figure 5.15*: *Results of applying the different filter combinations prior classification.*

Finally, in order to find the best way to reduce the number of bands, a comparison between the different techniques seen so far is performed. In this regard, the following processes are proposed, combining the best results obtained right now:

- **Process 1:** The dataset is filtered using original smooth, then smooth reduction filter is used to reduce to 115 bands and finally ICA with 21 features is applied.

- **Process 2:** The dataset obtained from process #1 is dimensionally reduced using ICA with 21 features.
- **Process 3:** The dataset is filtered using original smooth, then smooth reduction filter is used to reduce to 21 bands.

The results obtained for this process are shown in the following tables and bar graphs:

| **Comparative** | **Process 1** | **Process 2** | **Process 3** |
|---|---|---|---|
| **AA** | 94.16% (0.0065) | **94.78% (0.0054)** | 85.93% (0.0022) |
| **OA** | 92.49% (0.0087) | **93.34% (0.0080)** | 81.87% (0.0037) |
| **Kappa** | 0.8683 (0.0145) | **0.8826 (0.0134)** | 0.6988 (0.0049) |
| **Time (sec)** | 10.3263 | **9.5911** | 20.7463 |

*Table 5.16: Results of applying different ways to reduce the number of bands.*

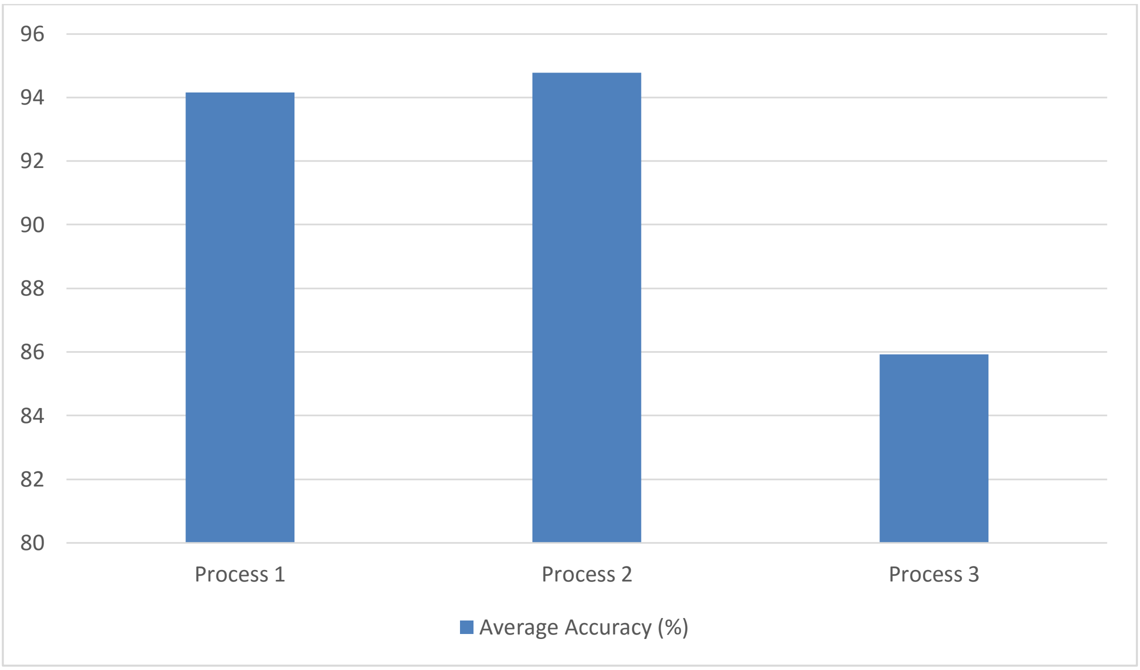


*Figure 5.16: Average Accuracy for different ways to reduce the number of bands.*

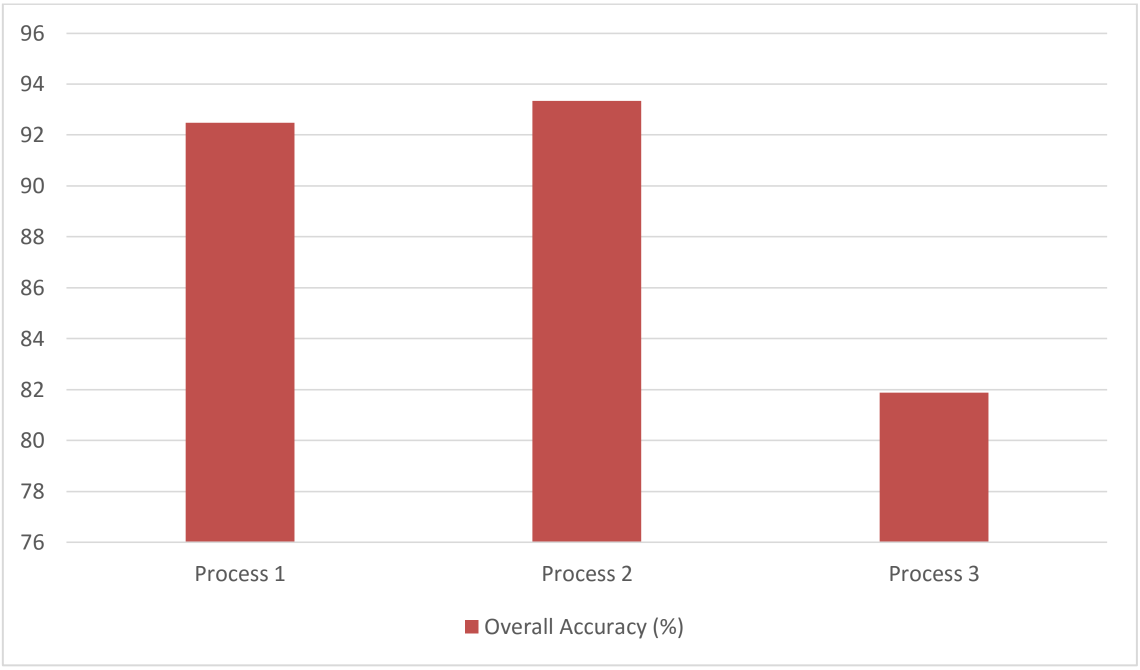


*Figure 5.17: Overall Accuracy for different ways to reduce the number of bands.*

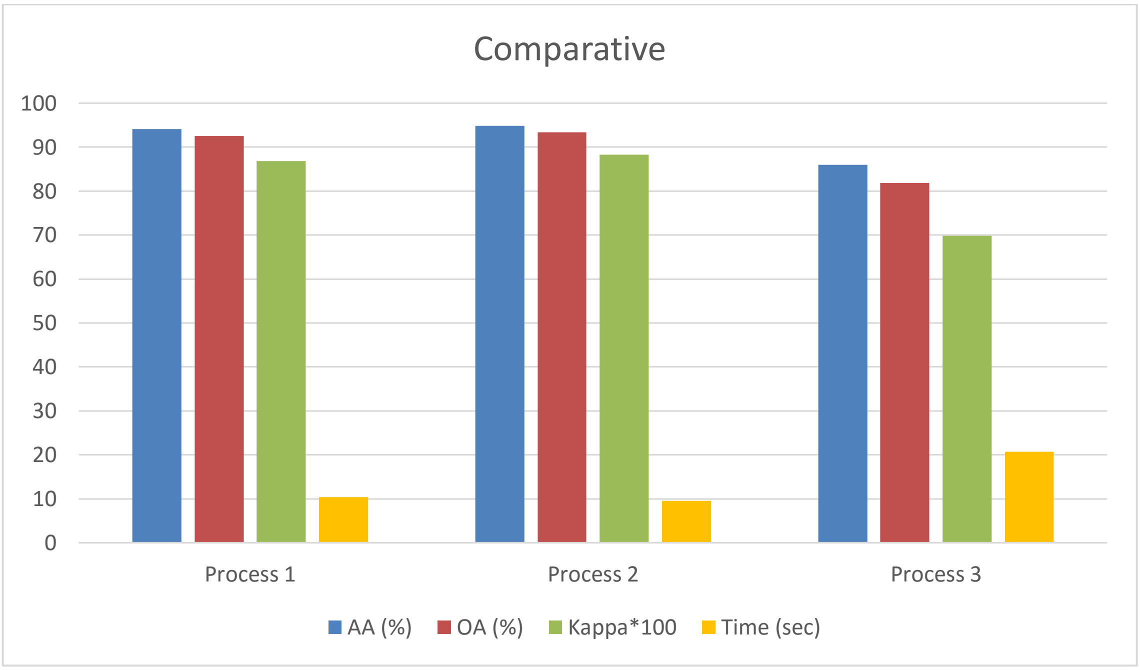


*Figure 5.18: Comparative for different ways to reduce the number of bands.*

Theoretically, applying dimensionality reduction the classification results should be improved by removing the latest bands or features in the transformed dataset, which have redundant information and noise. However, as shown by the above tables, dimensionality reduction prior to classification cannot lead to improved classification results with regards to the process chain #1 and the original spectral information. This indicates that for this type of data, all the bands present useful information even in the case of the last bands, although in these last bands the useful information is lower. In this regard, a final test is done to determine how it affects the elimination of the bands to the accuracy of the results. For this purpose, a PCA transformation has been applied to the dataset obtained from process #1, removing bands progressively. The results obtained are shown in the chart of figure 5.19 to figure 5.23:

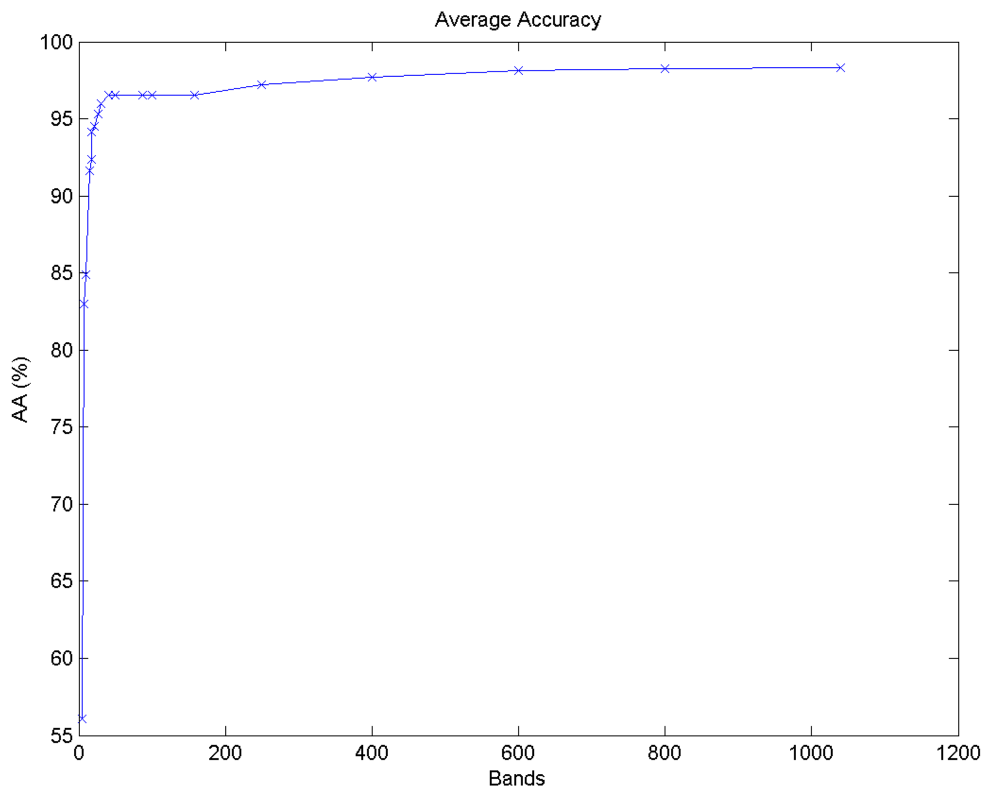


*Figure 5.19: Changes in average accuracy results for different number of bands.*

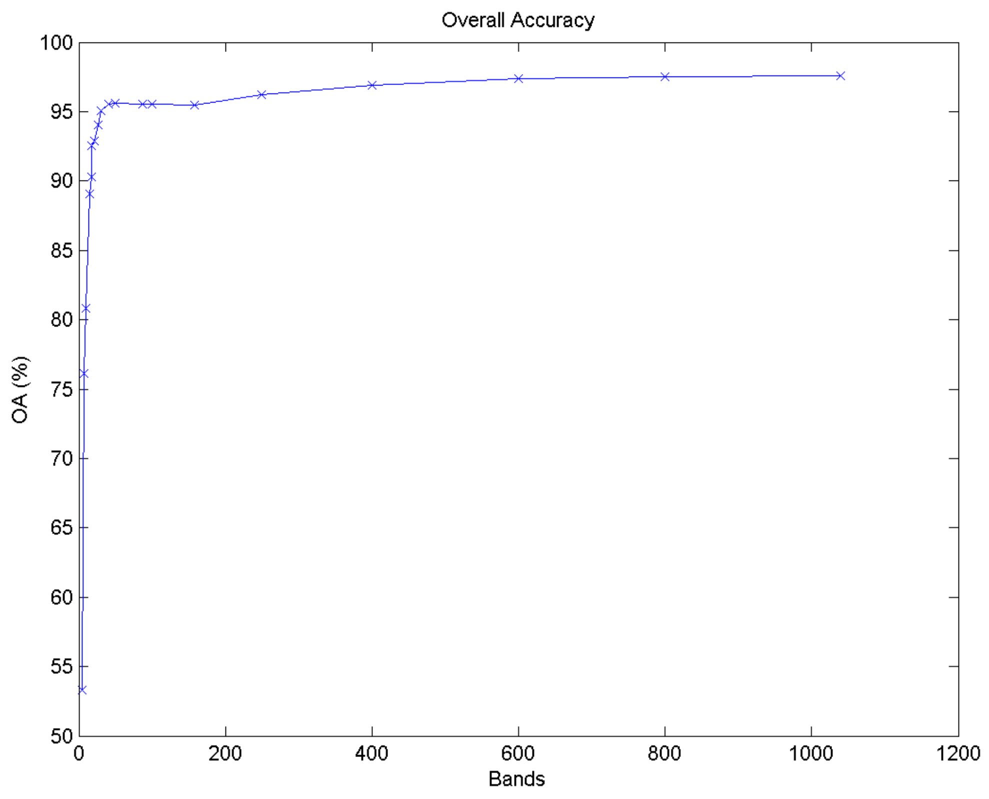


*Figure 5.20: Changes in overall accuracy results for different number of bands.*

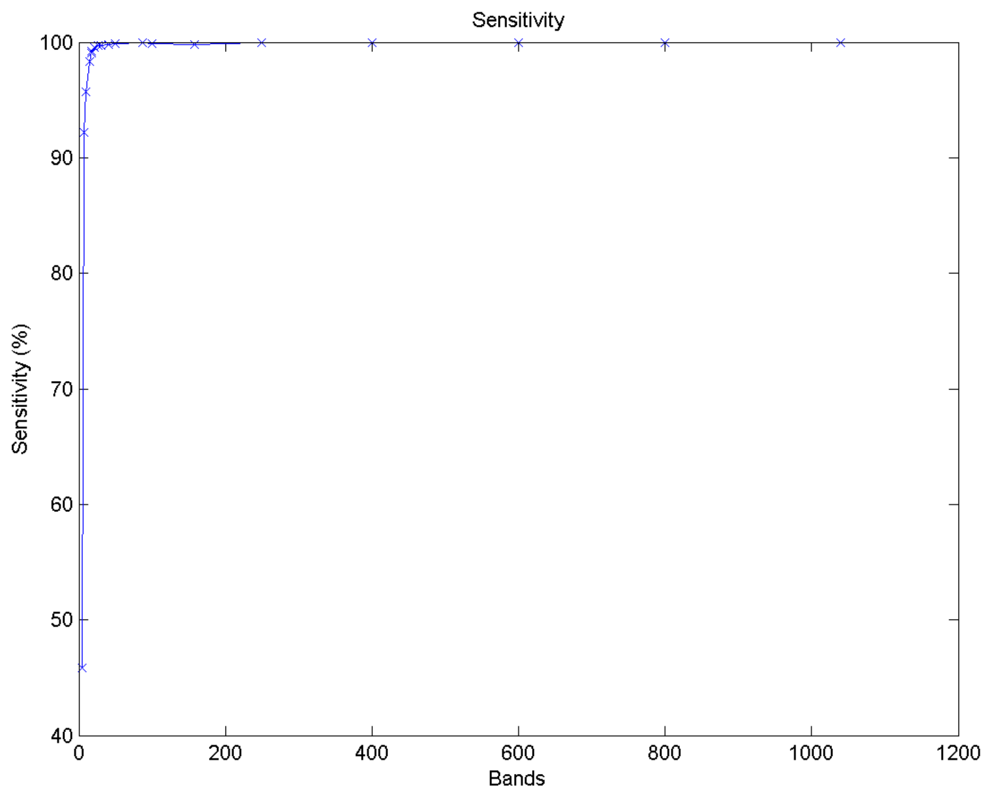


*Figure 5.21: Changes in sensitivity results for different number of bands.*

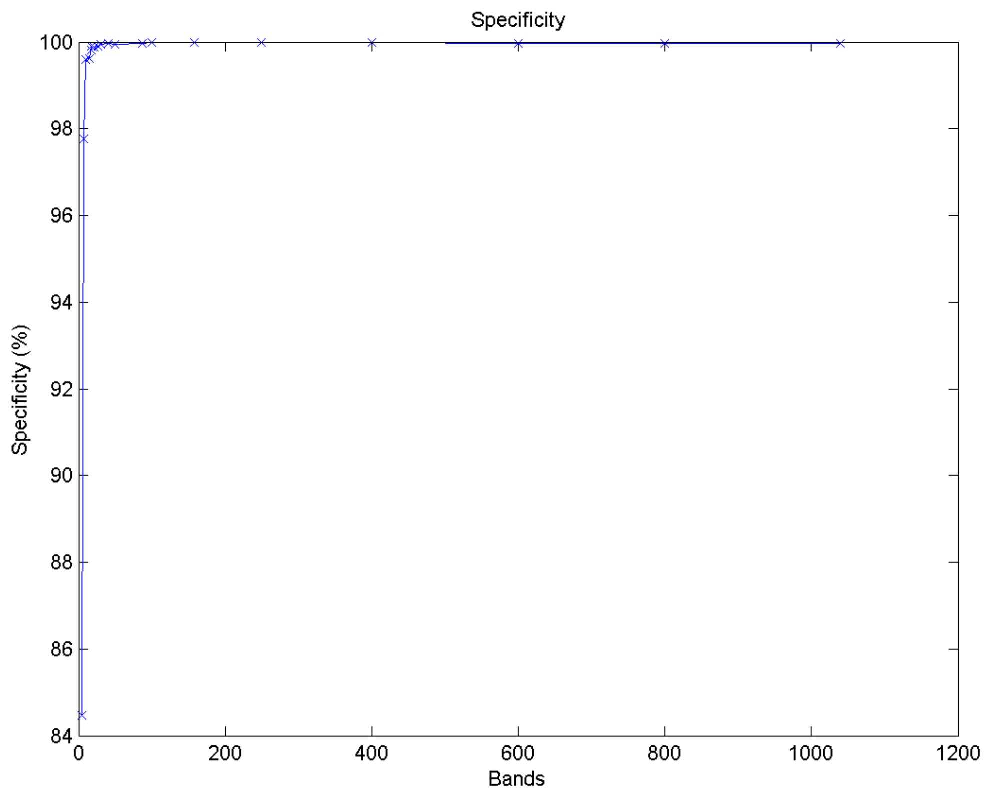


*Figure 5.22: Changes in specificity results for different number of bands.*

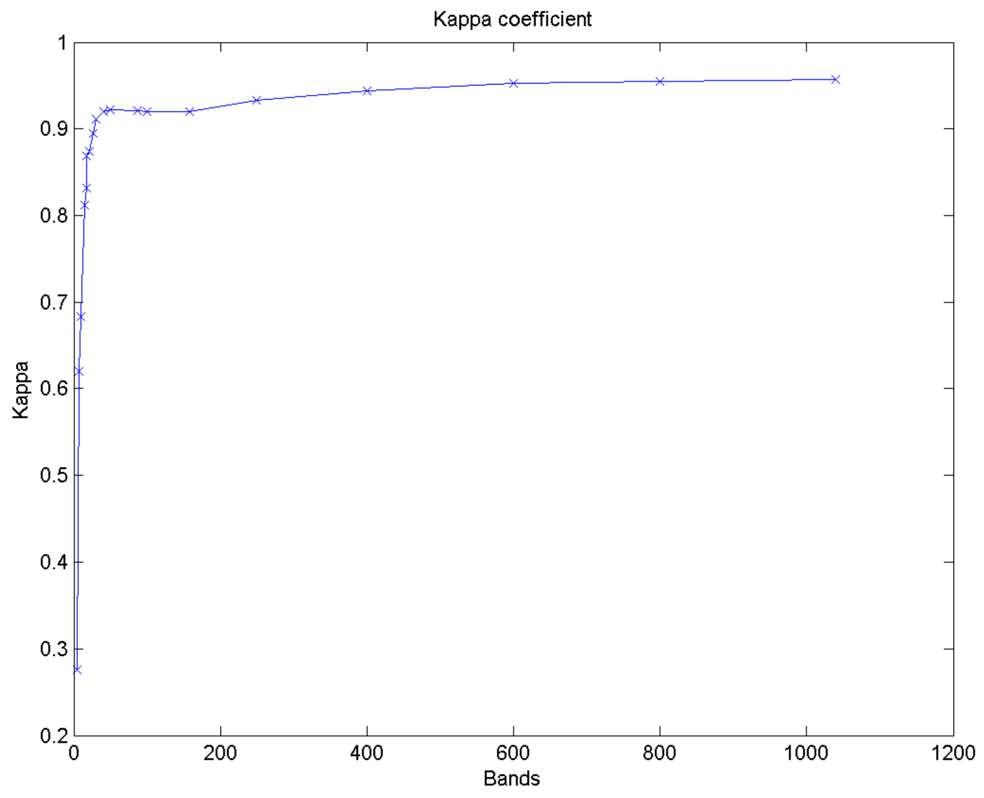


*Figure 5.23: Changes in kappa coefficient results for different number of bands.*

In the studies conducted, we can determine that even though all bands have some useful information, it is possible to remove large number of bands without suffering great losses in terms of the classification results obtained. This data reduction by removing bands is very important when dealing with computing time and computational load; because this system should work in real time and thanks to the dimensional reduction we are very close doing it. For all these reasons, we determine ICA as the best way to reduce this dataset, because ICA gives the best results both in accuracy and in computing time. In future sections, if it is necessary to apply a dimensionality reduction stage, ICA with 21 bands has been selected for this task.

### 5.1.3. Results of Spectral unmixing concepts used in brain cancer detection

In this process, unmixing techniques are applied independently. It means that this process chain has not classification stage. In this case, to measure the accuracy of success in this chain, the results are obtained from to label each pixel with its greatest abundance.

Several unmixing chains were tested in which we used the different techniques described in its theoretical section 3.1.3.2. The results are shown as “number of successes/number of pixels in that class” and its corresponding percentage of success. In some cases the endmembers could not be extracted by using a particular technique, so for these cases the results have not been shown. The unmixing chains proposed are the following:

1. **Spectral unmixing chain 1:** The endmembers are extracted from the dataset and then the abundances are estimated.

| Abundance estimate | FLSU | | LSU | |
|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **NFINDR** | **VCA** |
| **Class 1** | **1260/4469 28.1942%** | 647/4469 14.4775% | 818/4469 18.3039% | 618/4469 13.8286% |
| **Class 2** | **5667/10881 52.0816%** | 794/10881 7.2971% | 4531/10881 41.6414% | 1884/10881 17.3146% |
| **Class 3** | 1045/3230 32.3529% | **2732/3230 84.5820%** | 1019/3230 31.5480% | 2434/3230 75.3560% |
| **Total success** | **7972/18580 42.9064%** | 4173/18580 22.4596% | 6368/18580 34.2734% | 4936/18580 26.5662% |

*Table 5.17: Endmembers extracted from data set using unmixing techniques.*

2. **Spectral unmixing chain 2:** The endmembers are extracted from the normalized dataset and then the abundances are estimated.

| Abundance estimate | FLSU | | LSU | |
|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **NFINDR** | **VCA** |
| **Class 1** | **1650/4469 36.9210%** | 642/4469 14.3656% | 818/4469 18.3039% | 618/4469 13.8286% |
| **Class 2** | 3531/10881 32.4511% | 706/10881 6.4884% | **4531/10881 41.6414%** | 1884/10881 17.3146% |
| **Class 3** | 1213/3230 37.5542% | **2782/3230 86.1300%** | 1019/3230 31.5480% | 2434/3230 75.3560% |
| **Total success** | **6394/18580 34.4133%** | 4130/18580 22.2282% | 6368/18580 34.2734% | 4936/18580 26.5662% |

*Table 5.18: Endmembers extracted from normalized data set using unmixing techniques.*

3. **Spectral unmixing chain 3:** Dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | | **LSU** | |
|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **OSP** | **NFINDR** | **OSP** |
| **Class 1** | 1117/4469 24.9944% | 137/4469 3.0656% | **1159/4469 25.9342%** | 133/4469 2.9761% |
| **Class 2** | 3889/10881 35.7412% | 22/10881 0.2022% | **4083/10881 37.5241%** | 22/10881 0.2022% |
| **Class 3** | 2700/3230 83.5913% | **3224/3230 99.8142%** | 2722/3230 84.2724% | **3224/3230 99.8142%** |
| **Total success** | 7706/18580 41.4747% | 3383/18580 18.2078% | **7964/18580 42.8633%** | 3379/18580 18.1862% |

*Table 5.19: Endmembers extracted from dimensionality reduction data set using unmixing techniques.*

4. **Spectral unmixing chain 4:** The endmembers are extracted from the training dataset and then the abundances are estimated.
   In this case, the endmembers could not be extracted by using any of the techniques proposed.

5. **Spectral unmixing chain 5:** Dimensionality reduction is applied to training dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced training dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | **LSU** |
|---|---|---|
| **Endmembers extraction** | **NFINDR** | **NFINDR** |
| **Class 1** | 1546/4469 34.5939% | 1762/4469 39.4272% |
| **Class 2** | 3956/10881 36.3570% | 4203/10881 38.6270% |
| **Class 3** | 2488/3230 77.0279% | 2415/3230 74.7678% |
| **Total success** | 7990/18580 43.0032% | 8380/18580 45.1023% |

*Table 5.20: Endmembers extracted from training data set with dimensionality reduction using unmixing techniques.*

6. **Spectral unmixing chain 6:** One endmember is extracted from each class of the dataset and then the abundances are estimated.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **Class 1** | 992/4469 22.1974% | 2023/4469 45.2674% | **4469/4469 100%** | 1781/4469 39.8523% | 1903/4469 42.5822% | 4465/4469 99.9105% |
| **Class 2** | 2904/10881 26.6887% | 5862/10881 53.8737% | 32/10881 0.2941% | 643/10881 5.9094% | **6692/10881 61.5017%** | 47/10881 0.4319% |
| **Class 3** | 777/3230 24.0557% | 1421/3230 43.9938% | 4/3230 0.1238% | 892/3230 27.6161% | **1464/3230 45.3251%** | 8/3230 0.2477% |
| **Total success** | 4673/18580 25.1507% | 9306/18580 50.0861% | 4505/18580 24.2465% | 3316/18580 17.8471% | **10059/18580 54.1389%** | 4520/18580 24.3272% |

*Table 5.21: Each endmember extracted from each class using unmixing techniques.*

7. **Spectral unmixing chain 7:** Dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then one endmember is extracted from each class of the dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **Class 1** | 1789/4469 40.0313% | 3408/4469 76.2587% | 148/4469 3.3117% | 1720/4469 38.4874% | **4014/4469 89.8188%** | 141/4469 3.1551% |
| **Class 2** | 7945/10881 73.0172% | **10254/10881 94.2377%** | 22/10881 0.2022% | 9239/10881 84.9095% | 9409/10881 86.4718% | 22/10881 0.2022% |
| **Class 3** | 1145/3230 35.4489% | 1835/3230 56.8111% | 3223/3230 99.7833% | 951/3230 29.4427% | **2914/3230 90.2167%** | 3224/3230 99.8142% |
| **Total success** | 10879/18580 58.5522% | 15497/18580 83.4069% | 3393/18580 18.2616% | 11910/18580 64.1012% | **16337/18580 87.9279%** | 3387/18580 18.2293% |

*Table 5.22: Each endmember extracted from each class with dimensionality reduction using unmixing techniques.*

In general, the use of spectral unmixing in this kind of samples doesn't offer competitive results. This is because the data set is very diverse and a whole class cannot be represented using only one endmember. In order to try to solve this problem and improve the results obtained it is intended to extract the endmembers from an average of several pixels because in this way, using multiple pixels to form the endmembers, the classes will be better represented.

In this case, the results are accompanied with its corresponding abundance maps of each class, displaying the fractional amount of that class present at each pixel. This allows us to visually verify the correct behavior of processes tested. In these maps the pixels distribution is as follows:

- **Class 1:** It is comprised between the pixels 0 and 223.
- **Class 2:** It is comprised between the pixels 224 and 767.
- **Class 3:** It is comprised between the pixels 768 and 929.

Taking into account that for the above unmixing chains the best results are obtained from dimensionality reduction dataset, the processing for these remaining chains is always the same: dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced dataset and the abundances are estimated. In these cases, they are differentiated by its endmembers extraction technique. The chain proposed is the following one:

8. **Spectral unmixing chain 8:** 6 endmembers are extracted from each class, and then the means of these endmembers are calculated for each class resulting in 3 endmembers (one per class). The abundances are estimate using these 3 endmembers formed by the means of extracted endmembers.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **Class 1** | **4192/4469 93.8017%** | 1849/4469 41.3739% | 2881/4469 64.4663% | 3827/4469 85.6344% | 1829/4469 40.9264% | 3730/4469 83.4639% |
| **Class 2** | **7861/10881 72.2452%** | 3948/10881 36.2834% | 80/10881 0.7352% | 7634/10881 70.1590% | 3920/10881 36.0261% | 108/10881 0.9926% |
| **Class 3** | 1263/3230 39.1022% | 2879/3230 89.1331% | **3150/3230 97.5232%** | 1877/3230 58.1115% | 2882/3230 89.2260% | 3080/3230 95.3560% |
| **Total success** | 13316/18580 71.6685% | 8676/18580 46.6954% | 6111/18580 32.8902% | **13338/18580 71.7869%** | 8631/18580 46.4532% | 6918/18580 37.2336% |

*Table 5.23: 6 endmembers extracted from each class and the mean of these endmembers are calculated for each class, resulting in 3 endmembers using unmixing techniques.*

- Abundances map using NFINDR in spectral unmixing chain 8:

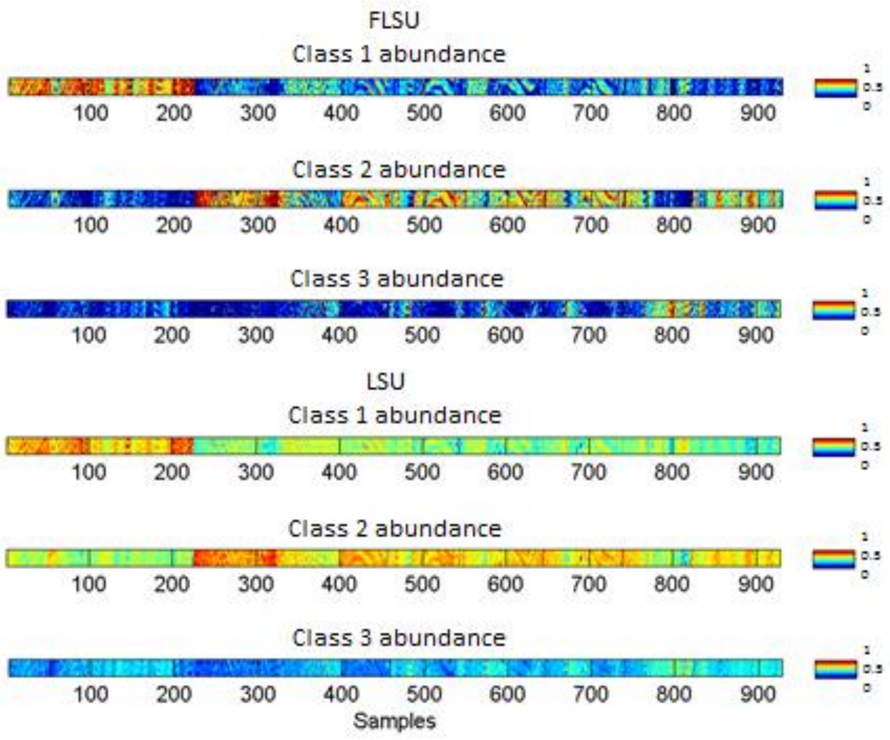


*Figure 5.24: Abundance map using NFINDR* in spectral unmixing chain 8*.*

- Abundances map using VCA in spectral unmixing chain 8:

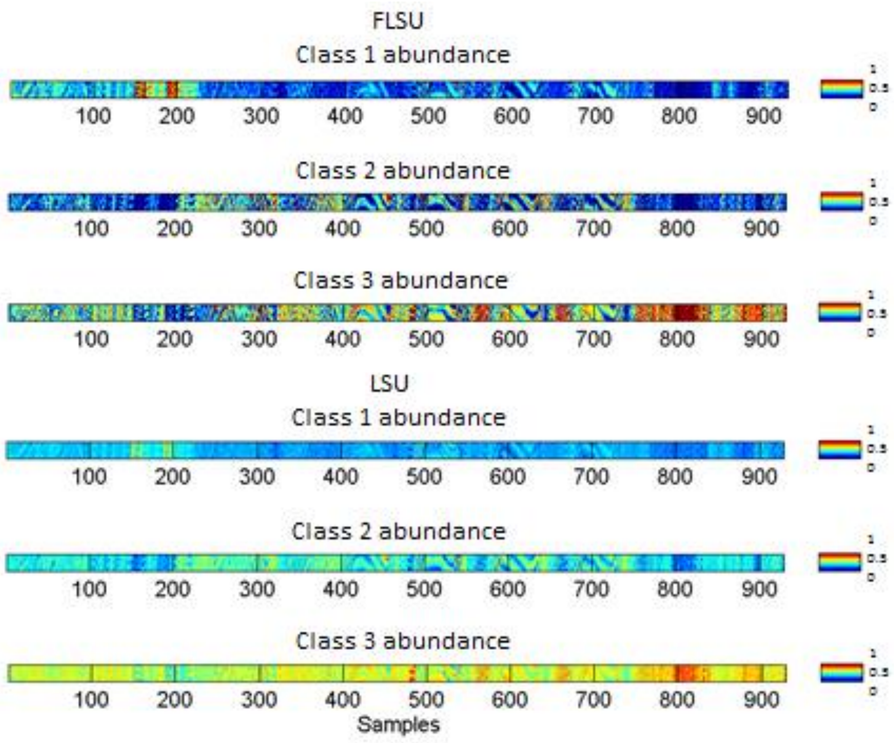


*Figure 5.25: Abundance map using VCA* in spectral unmixing chain 8*.*

- Abundances map using VCA in spectral unmixing chain 8:

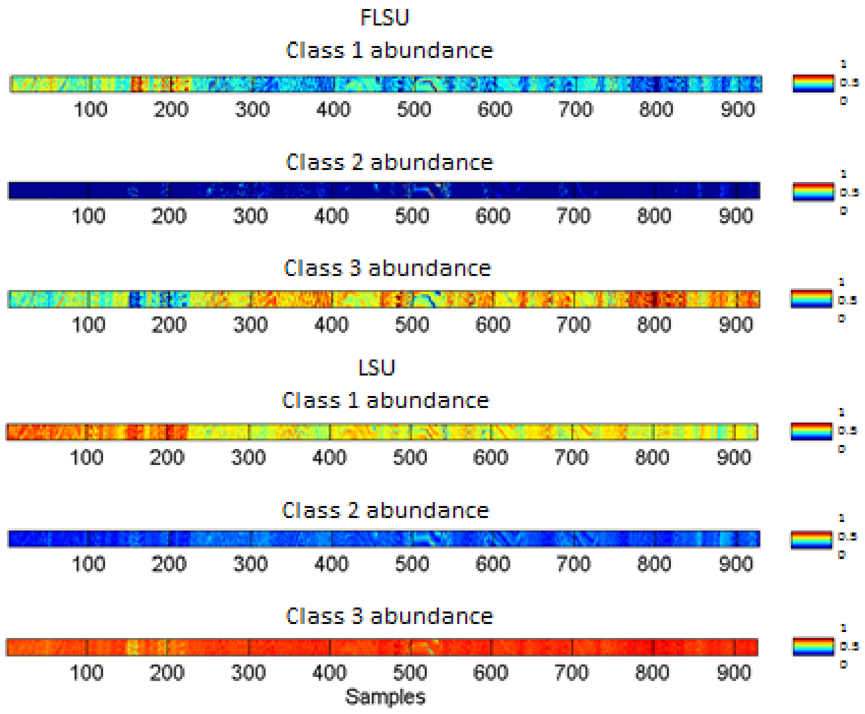


*Figure 5.26: Abundance map using VCA* in spectral unmixing chain 8.

9. **Spectral unmixing chain 9:** 6 endmembers are extracted from each class and the abundances are estimate using the 18 endmembers obtained, then the estimated abundances for each class are added resulting in 3 abundances.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **Class 1** | 4036/4469 90.3110% | **4253/4469 95.1667%** | 4203/4469 94.0479% | 1931/4469 43.2088% | 3500/4469 78.3173% | 1894/4469 42.3808% |
| **Class 2** | 6871/10881 63.1468% | 6881/10881 63.2387% | 2893/10881 26.5876% | 4999/10881 45.9425% | **8437/10881 77.5388%** | 2884/10881 26.5049% |
| **Class 3** | 1913/3230 59.2260% | **3095/3230 95.8204%** | 2709/3230 83.8700% | 1815/3230 56.1920% | 2684/3230 83.0960% | 2817/3230 87.2136% |
| **Total success** | 12820/18580 68.9989% | 14229/18580 76.5823% | 9805/18580 52.7718% | 8745/18580 47.0667% | **14621/18580 78.6921%** | 7595/18580 40.8773% |

*Table 5.24: 6 endmembers extracted from each class and abundances are added per class resulting in 3 abundances using unmixing techniques.*

- Abundances map using NFINDR in spectral unmixing chain 9:

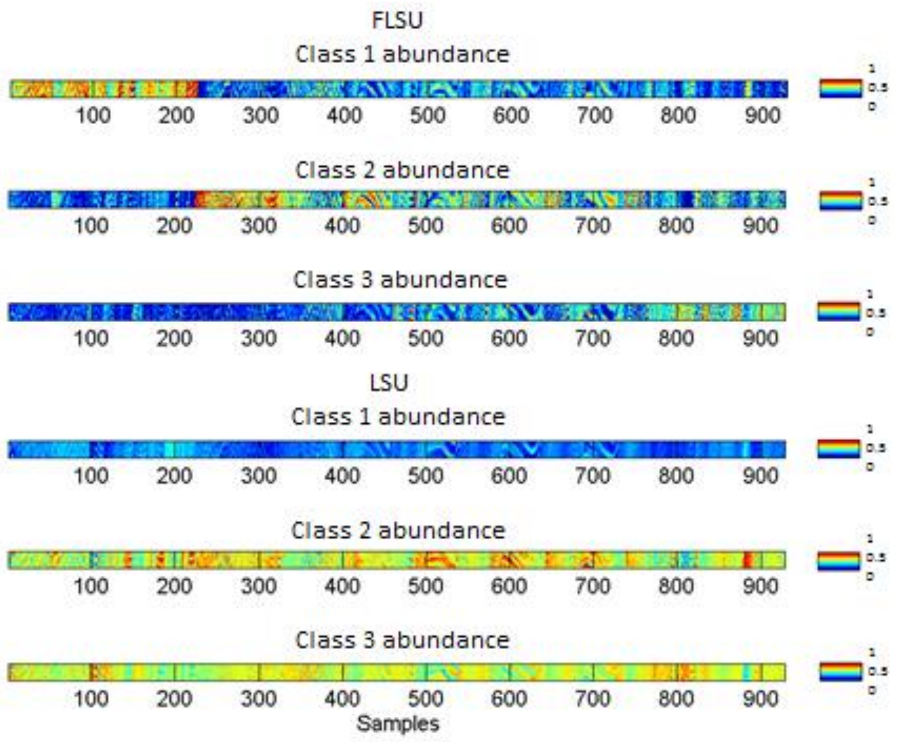


*Figure 5.27: Abundance map using NFINDR* in spectral unmixing chain 9*.*

- Abundances map using VCA in spectral unmixing chain 9:

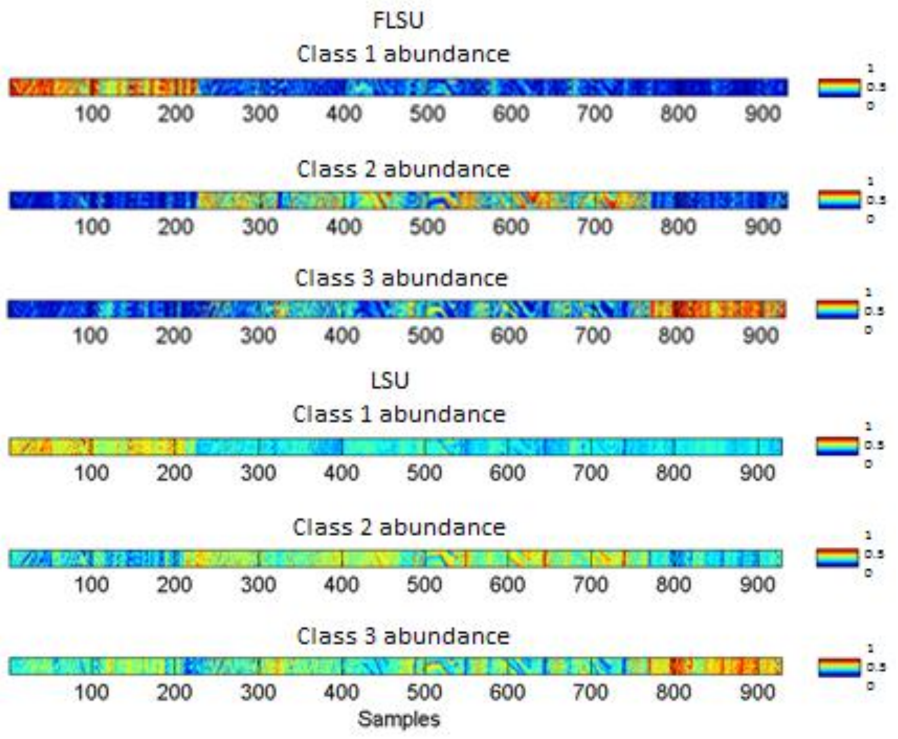


*Figure 5.28: Abundance map using VCA* in spectral unmixing chain 9*.*

- Abundances map using OSP in spectral unmixing chain 9:

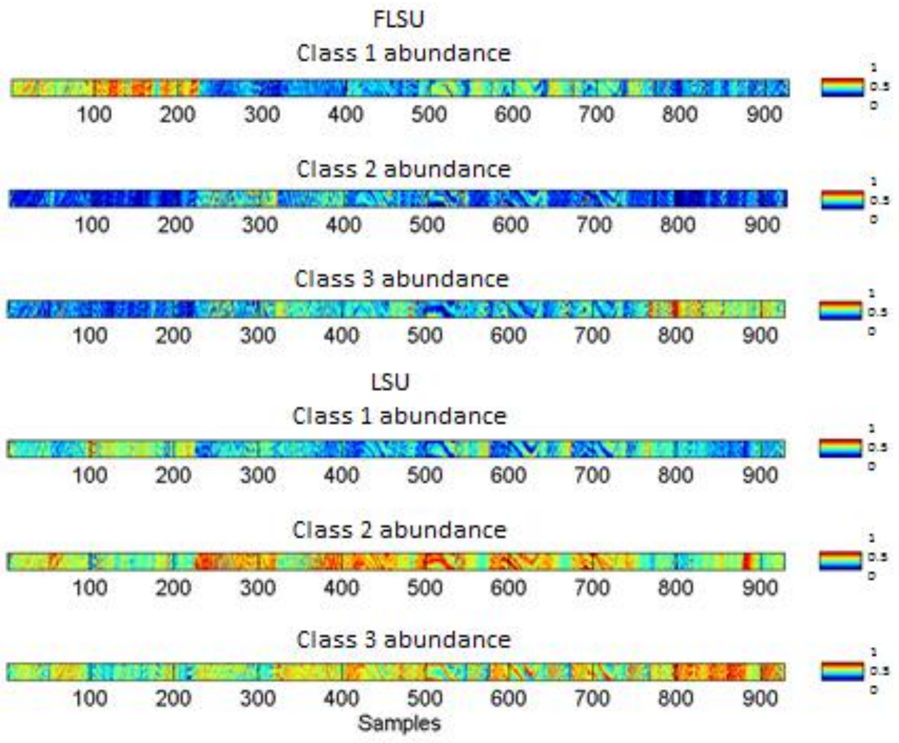


*Figure 5.29: Abundance map using OSP* in spectral unmixing chain 9.

10. **Spectral unmixing chain 10:** Endmembers extraction is not implemented. Instead, each endmember is calculated as the average of all pixels in its class and then the abundances estimation is performed.

| **Abundance estimate** | **FLSU** | | **LSU** | |
|---|---|---|---|---|
| **Datatype** | **Dimensionality reduction** | **Original dataset** | **Dimensionality reduction** | **Original dataset** |
| **Class 1** | **4293/4469 96.0618%** | 2237/4469 50.0559% | 4027/4469 90.1096% | 2037/4469 45.5807% |
| **Class 2** | 9292/10881 85.3966% | 4723/10881 43.4059% | **9375/10881 86.1594%** | 6144/10881 56.4654% |
| **Class 3** | **2956/3230 91.5170%** | 1836/3230 56.8421% | 2918/3230 90.3406% | 1618/3230 50.0929% |
| **Total success** | **16541/18580 89.0258%** | 8796/18580 47.3412% | 16320/18580 87.8364% | 9799/18580 52.7395% |

*Table 5.25: Each endmember is calculated as the average of all pixels in its class using unmixing techniques.*

- Abundances map using dataset with dimensionality reduction in spectral unmixing chain 10:

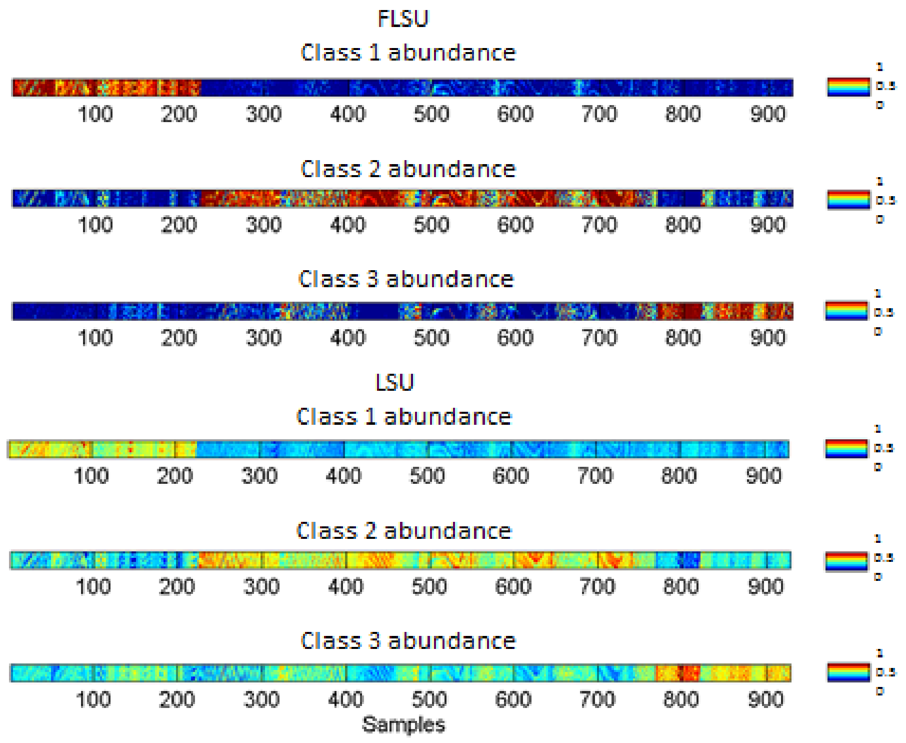


*Figure 5.30: Abundance map using dataset with dimensionality reduction* in spectral unmixing chain 10.

- Abundances map using original dataset in spectral unmixing chain 10:

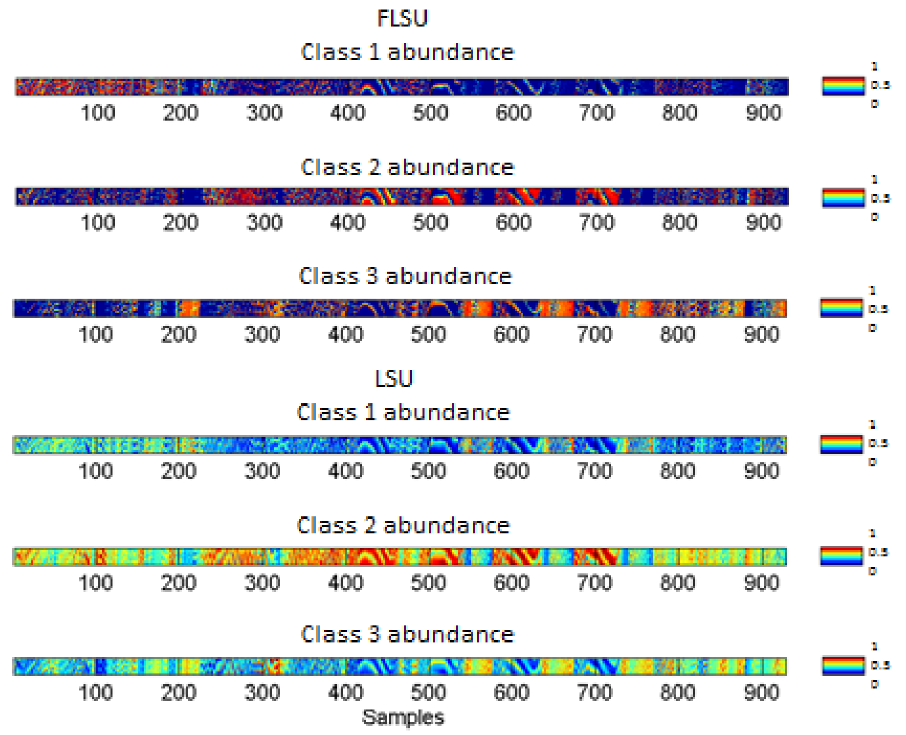


*Figure 5.31: Abundance map using original dataset* in spectral unmixing chain 10.

In general, this new approach to extract the endmembers based on obtaining these as an average of several pixels improves the results obtained, especially in the last unmixing chain with dimensionality reduction. This is because endmember extraction techniques are generally sensitive to outliers and anomalies. The objective of this procedure is to ensure that the pixels are as representative as possible of each class.

Although the results have improved with the latest tests, the results are still worse than in the case of applying only dimensional reduction or in the case of not reducing the size of the data in any way and work on the original data. In the next section we will try to improve the results using combined techniques of unmixing and classification.

### 5.1.4. Results of Mixed techniques Unmixing-SVM used in brain cancer detection

In this section we apply mixed techniques Unmixing-SVM in order to improve the results obtained in the last section in which only unmixing techniques were applied. For this purpose, the process chains proposed in this section are exactly the same as those used in the previous section, except that at the end a classification step is performed by the SVM classifier, which has as input data the estimated abundances. Processing chains are as follows:

1. **Unmixing-SVM chain 1:** The endmembers are extracted from the dataset and then the abundances are estimated.

| Abundance estimate | FLSU | | LSU | |
|---|---|---|---|---|
| Endmembers extraction | NFINDR | VCA | NFINDR | VCA |
| AA | 44.49% (0.0055) | **54.86% (0.0022)** | 52.42% (0.0022) | 53.39% (0.0023) |
| OA | 28.60% (0.0072) | 47.46% (0.0041) | 46.89% (0.0106) | **48.65% (0.0066)** |
| Kappa | 0.0847 (0.0065) | **0.2428 (0.0025)** | 0.2114 (0.0044) | 0.2317 (0.0038) |
| Time (sec) | 36.9339 | 25.3817 | 15.0297 | **9.4290** |

*Table 5.26: Endmembers extracted from data set using mixed unmixing-SVM techniques.*

2. **Unmixing-SVM chain 2:** The endmembers are extracted from the normalized dataset and then the abundances are estimated.

| Abundance estimate | FLSU | | LSU | |
|---|---|---|---|---|
| Endmembers extraction | NFINDR | VCA | NFINDR | VCA |
| AA | 42.83% (0.0126) | **54.10% (0.0032)** | 52.42% (0.0022) | 53.39% (0.0023) |
| OA | 24.89% (0.0067) | 46.71% (0.0027) | 46.89% (0.0106) | **48.65% (0.0066)** |
| Kappa | 0.0581 (0.0117) | **0.2365 (0.0021)** | 0.2114 (0.0044) | 0.2317 (0.0038) |
| Time (sec) | 33.8021 | 25.2906 | 15.1259 | **9.5866** |

*Table 5.27: Endmembers extracted from normalized data set using mixed unmixing-SVM techniques.*

3. **Unmixing-SVM chain 3:** Dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | | **LSU** | |
|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **OSP** | **NFINDR** | **OSP** |
| **AA** | **54.20% (0.0025)** | 41.08% (0.0031) | 53.85% (0.0023) | 41.42% (0.0060) |
| **OA** | 49.98% (0.0051) | **54.47% (0.0152)** | 49.57% (0.0042) | 53.79% (0.0197) |
| **Kappa** | **0.2282 (0.0020)** | 0.0990 (0.0049) | 0.2211 (0.0027) | 0.1057 (0.0058) |
| **Time (sec)** | 23.4082 | 21.7729 | 14.6559 | **13.2206** |

*Table 5.28: Endmembers extracted from dimensionality reduction data set using mixed unmixing-SVM techniques.*

4. **Unmixing-SVM chain 4:** The endmembers are extracted from the training dataset and then the abundances are estimated.

   In this case, the endmembers could not be extracted by using any of the techniques proposed.

5. **Unmixing-SVM chain 5:** Dimensionality reduction is applied to training dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced training dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | **LSU** |
|---|---|---|
| **Endmembers extraction** | **NFINDR** | **NFINDR** |
| **AA** | 53.52% (0.0022) | **54.26% (0.0024)** |
| **OA** | **50.45% (0.0033)** | 49.98% (0.0029) |
| **Kappa** | **0.2332 (0.0021)** | 0.2307 (0.0031) |
| **Time (sec)** | 19.5345 | **11.2028** |

*Table 5.29: Endmembers extracted from training data set with dimensionality reduction using mixed unmixing-SVM techniques.*

6. **Unmixing-SVM chain 6:** One endmember is extracted from each class of the dataset and then the abundances are estimated.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **AA** | 45.25% (0.0037) | 47.75% (0.0010) | 40.97% (0.0231) | 50.54% (0.0028) | 55.56% (0.0019) | **62.71% (0.0034)** |
| **OA** | 26.98% (0.0099) | 50.50% (0) | 29.64% (0.0823) | 46.73% (0.0085) | 50.78% (0.0044) | **58.84% (0.0056)** |
| **Kappa** | 0.0821 (0.0037) | 0.1923 (0.0013) | 0.0756 (0.0454) | 0.1981 (0.0054) | 0.2624 (0.0026) | **0.3611 (0.0061)** |
| **Time (sec)** | 25.7002 | 20.1356 | 19.6977 | 10.6876 | 8.7667 | **5.7681** |

*Table 5.30: Each endmember extracted from each class using mixed unmixing-SVM techniques.*

7. **Unmixing-SVM chain 7:** Dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then one endmember is extracted from each class of the dataset and the abundances are estimated.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **AA** | 52.73% (0.0021) | 85.58% (0.0016) | 40.54% (0.0050) | 52.59% (0.0042) | **91.77% (0)** | 40.63% (0.0072) |
| **OA** | 51.88% (0.0083) | 84.15% (0.0028) | 54.02% (0.0107) | 55.70% (0.0074) | **89.53% (0.0019)** | 53.59% (0.0134) |
| **Kappa** | 0.2357 (0.0051) | 0.7264 (0.0039) | 0.1022 (0.0067) | 0.2505 (0.0052) | **0.8188 (0.0028)** | 0.1074 (0.0085) |
| **Time (sec)** | 20.2335 | 16.8364 | 19.6551 | 11.8099 | **7.2237** | 11.1377 |

*Table 5.31: Each endmember extracted from each class with dimensionality reduction using mixed unmixing-SVM techniques.*

As it happened in the previous section, the last chains proposals share the following processing: dimensionality reduction is applied to dataset using ICA to reduce 21 bands, then the endmembers are extracted from the reduced dataset and the abundances are estimated. Because based on previous chains, this processing is working best. Then, the different cases are differentiated by its endmembers extraction way.

8. **Unmixing-SVM chain 8:** 6 endmembers are extracted from each class, and then the means of these endmembers are calculated for each class resulting in 3 endmembers (one per class). The abundances are estimate using these 3 endmembers formed by the means of extracted endmembers.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **AA** | 74.96% (0.0017) | 63.93% (0.0021) | 65.84% (0.0020) | **76.02% (0.0016)** | 63.10% (0.0023) | 68.81% (0.0016) |
| **OA** | **72.36% (0.0033)** | 58.37% (0.0068) | 60.78% (0.0077) | 71.74% (0.0022) | 59.15% (0.0049) | 65.14% (0.0068) |
| **Kappa** | **0.5463 (0.0038)** | 0.3531 (0.0034) | 0.3822 (0.0059) | 0.5450 (0.0020) | 0.3506 (0.0027) | 0.4418 (0.0065) |
| **Time (sec)** | 32.1258 | 18.7245 | 28.9436 | 23.0316 | **10.8325** | 13.1297 |

*Table 5.32: 6 endmembers extracted from each class and the mean of these endmembers are calculated for each class, resulting in 3 endmembers using mixed unmixing-SVM techniques.*

9. **Unmixing-SVM chain 9:** 6 endmembers are extracted from each class and the abundances are estimate using the 18 endmembers obtained, then the estimated abundances for each class are added resulting in 3 abundances.

| **Abundance estimate** | **FLSU** | | | **LSU** | | |
|---|---|---|---|---|---|---|
| **Endmembers extraction** | **NFINDR** | **VCA** | **OSP** | **NFINDR** | **VCA** | **OSP** |
| **AA** | 73.52% (0.0011) | **88.23% (0)** | 72.71% (0.0017) | 53.28% (0.0036) | 84.23% (0.0015) | 64.59% (0.0021) |
| **OA** | 68.80% (0.0044) | **85.66% (0.0025)** | 66.66% (0.0036) | 53.30% (0.0062) | 81.28% (0.0049) | 63.37% (0.0027) |
| **Kappa** | 0.5026 (0.0042) | **0.7546 (0.0036)** | 0.4728 (0.0026) | 0.2642 (0.0064) | 0.6843 (0.0065) | 0.4030 (0.0031) |
| **Time (sec)** | 59.2323 | 40.2707 | 48.2130 | 25.0914 | **9.1311** | 13.5913 |

*Table 5.33: 6 endmembers extracted from each class and abundances are added per class resulting in 3 abundances using mixed unmixing-SVM techniques.*

10. **Unmixing-SVM chain 10:** Endmembers extraction is not implemented. Instead, each endmember is calculated as the average of all pixels in its class and then the abundances estimation is performed.

| **Abundance estimate** | **FLSU** | | **LSU** | |
|---|---|---|---|---|
| **Datatype** | **Dimensionality reduction** | **Original** | **Dimensionality reduction** | **Original** |
| **AA** | 91.64% (0) | 50.09% (0.0019) | **91.79% (0)** | 54.78% (0.0016) |
| **OA** | 89.41% (0.0023) | 47.33% (0.0012) | **89.55% (0.0020)** | 49.88% (0.0049) |
| **Kappa** | 0.8168 (0.0034) | 0.1991 (0.0019) | **0.8192 (0.0029)** | 0.2507 (0.0031) |
| **Time (sec)** | 13.8637 | 22.0215 | **7.1212** | 10.9168 |

*Table 5.34: Each endmember is calculated as the average of all pixels in its class using mixed unmixing-SVM techniques.*

As it can be seen, we have managed to improve the results obtained thanks to the inclusion of a classification step at the end of unmixing chains. However, the results remain lower than experimental results in process #1 and #2. In addition, no significant improvement was seen in the computing time compared to losses in accuracy.

### 5.1.5. Results of using Different methods for extracting training set using SVM in brain cancer detection

This section has two different objectives:

1. To conduct a study on how the classification results change according to the training dataset size increases.
2. To try to improve the classification results using guided and semi-guided training.

To achieve our first objective, a series of classification systems are performed, where the percentage of training dataset size is progressively increased. These classifications are made using the best cases for process #1, #2 and #4, and these results are shown in several graphs. It is known that when the training sets increase, the computing time and accuracy classification also increases. However, if training dataset continues increasing, there is a tipping point where the accuracy stops, and even if the training dataset continue to increase even more there is a turning point where the accuracy begins to decrease. This study aims to observe this phenomenon and check if there is any optimal size for training dataset. For this purpose the following experiments have been made:

- Process #1:

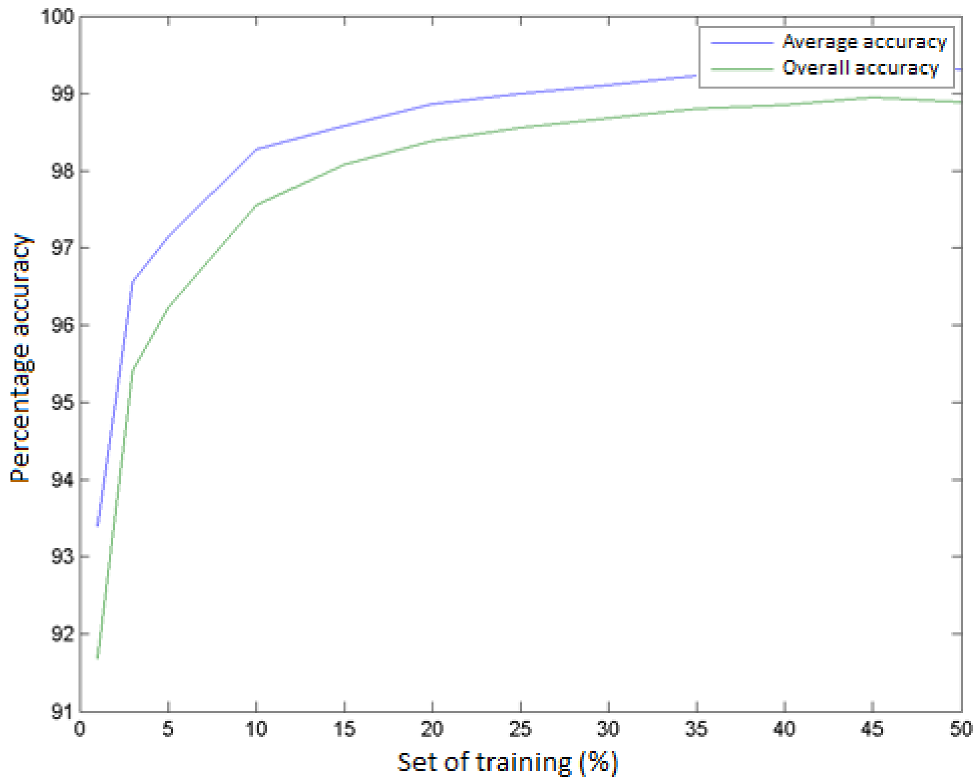


*Figure 5.32: AA (%) and OA (%) using process #1 for training dataset of different size.*

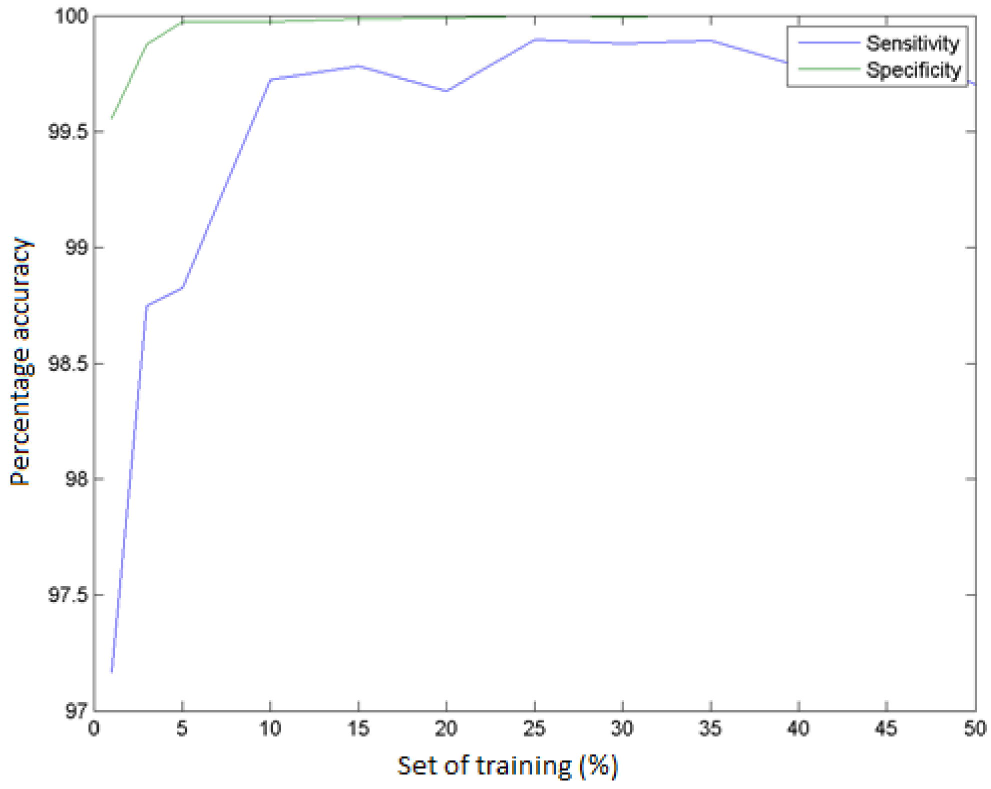


*Figure 5.33: Class 1 sensitivity (%) and specificity (%) using process #1 for training dataset of different size.*

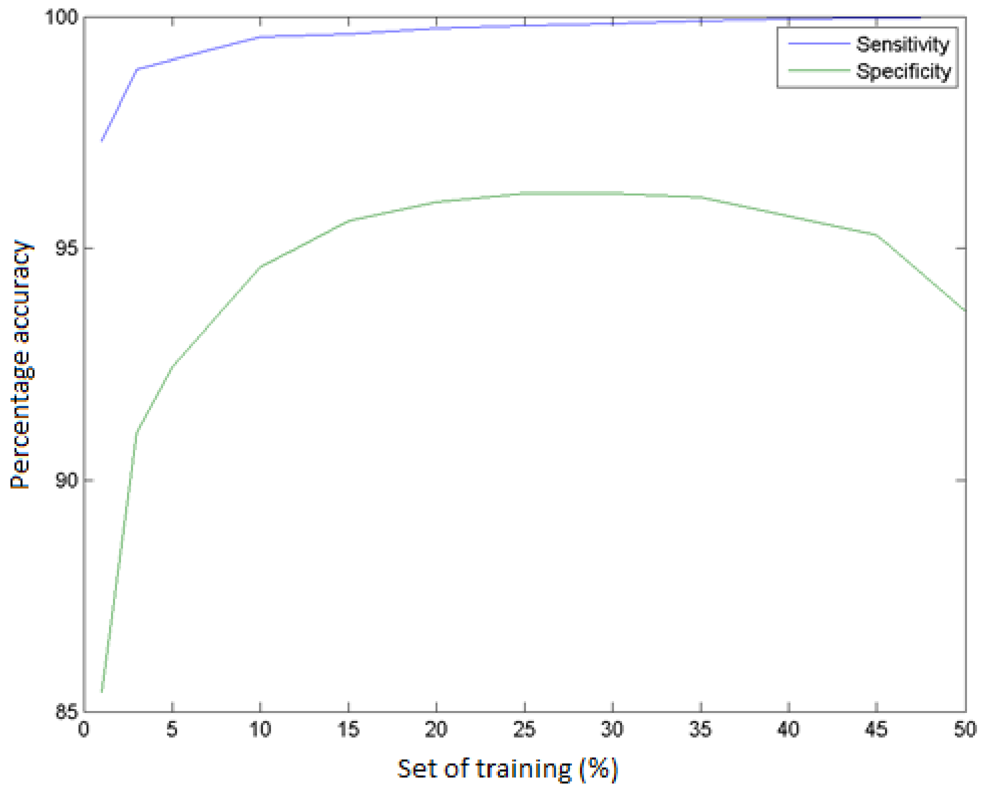


*Figure 5.34: Class 2 sensitivity (%) and specificity (%) using process #1 for training dataset of different size.*

- Process #2:

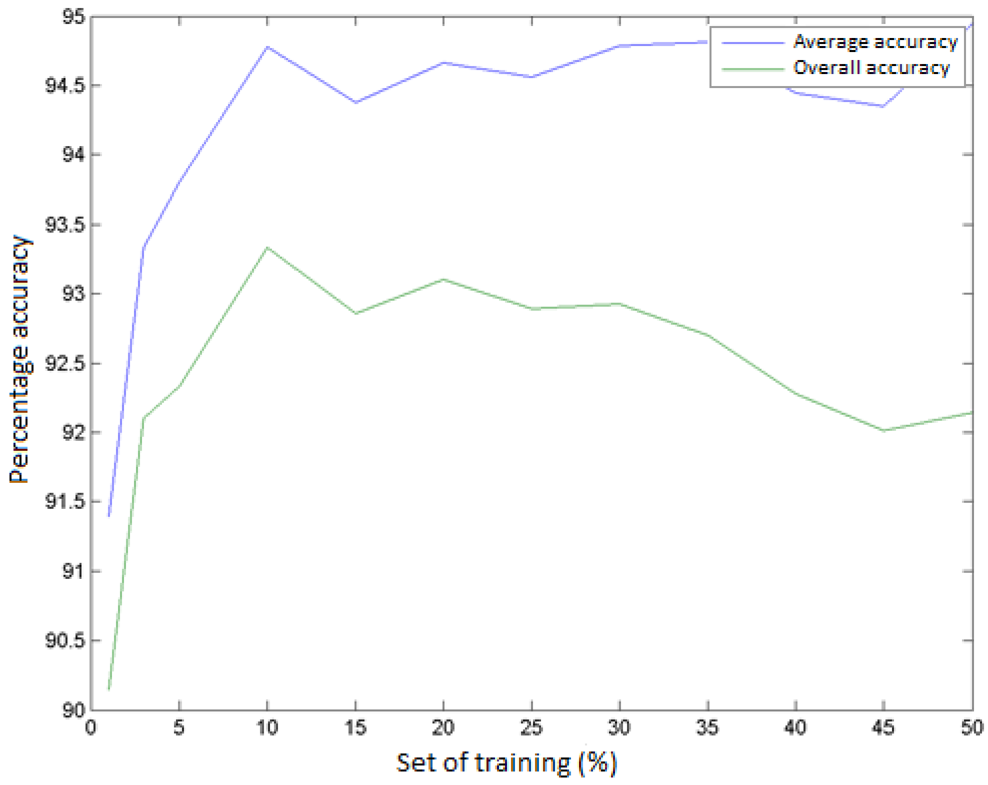


*Figure 5.35: AA (%) and OA (%) using process #2 for training dataset of different size.*

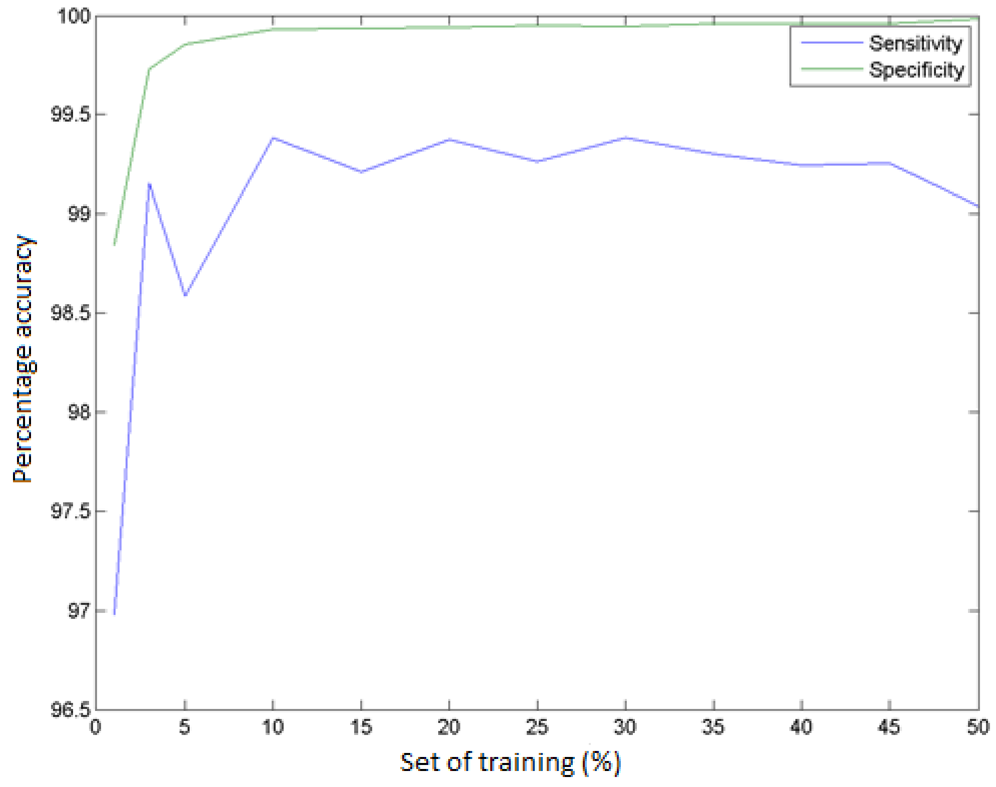


*Figure 5.36 Class 1 sensitivity (%) and specificity (%) using process #2 for training dataset of different size.*

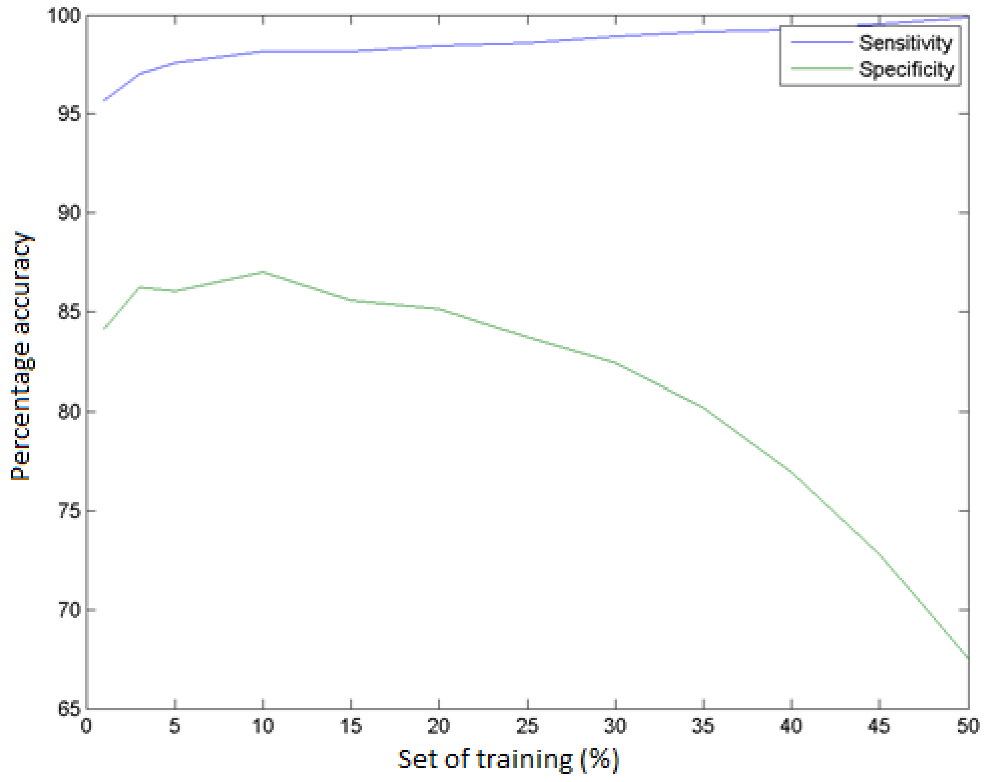


*Figure 5.37: Class 2 sensitivity (%) and specificity (%) using process #2 for training dataset of different size.*

- Process #4:

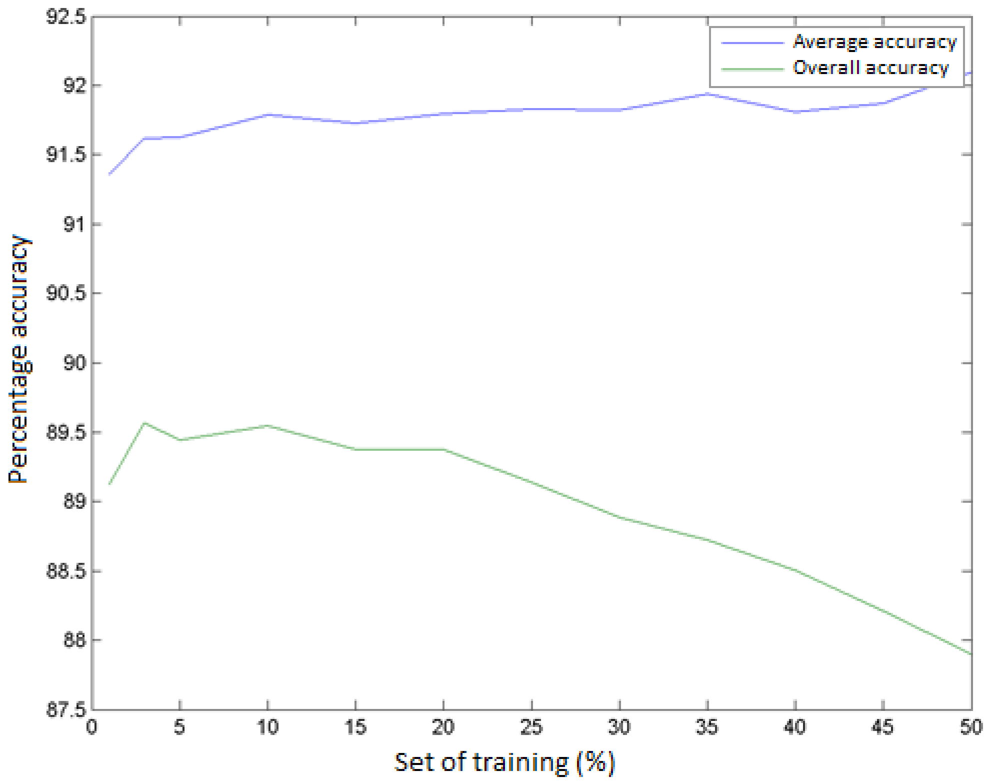


*Figure 5.38: AA (%) and OA (%) using process #4 for training dataset of different size.*

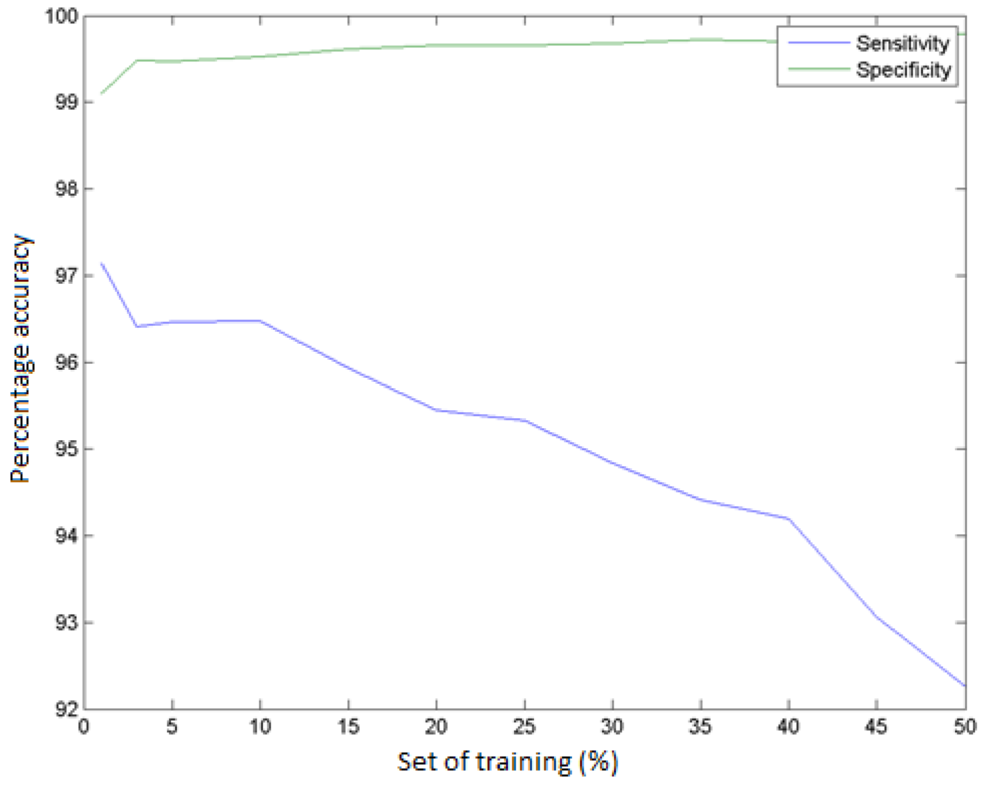


*Figure 5.39: Class 1 sensitivity (%) and specificity (%) using process #4 for training dataset of different size.*

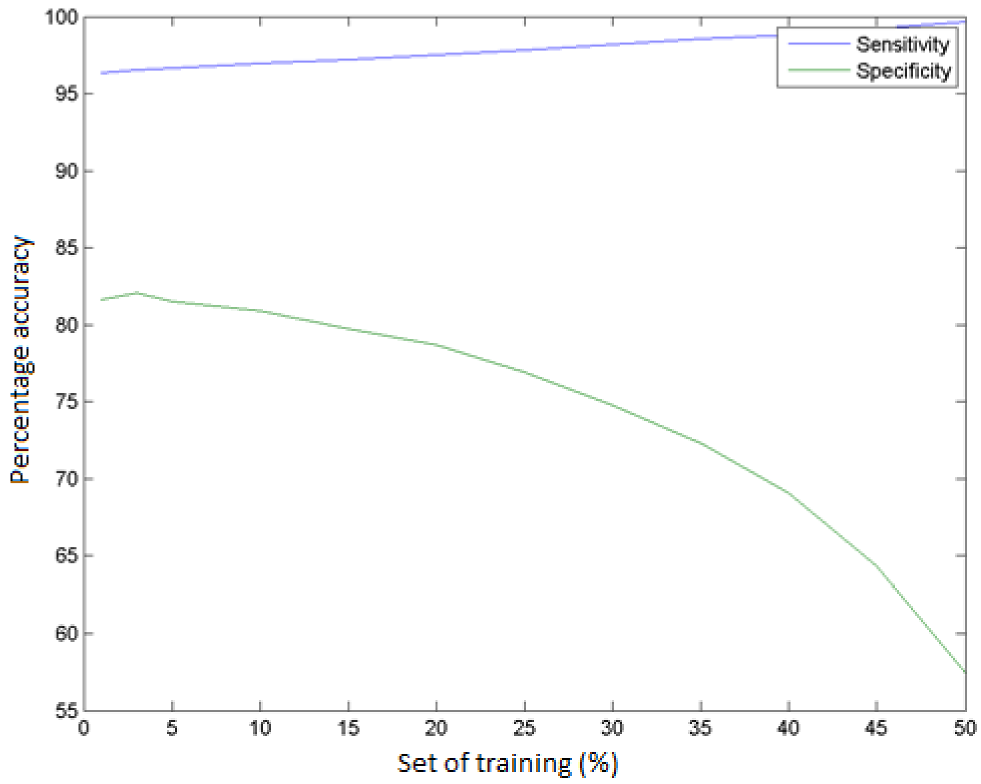


*Figure 5.40: Class 2 sensitivity (%) and specificity (%) using process #4 for training dataset of different size.*

In general, we obtain the expected behavior except in cases of figure 5.37 and figure 5.40 corresponding to sensitivity and specificity for class 2 in process #2 and #4. In these cases, we can see how the specificity for class 2 decreases continuously from the beginning reaching very low values. To find out what happens, we proceed to remove Class 3 in order to simplify the problem. Repeating these calculations with only 2 classes leads to the results shown in the following graphics:

- Process #1:

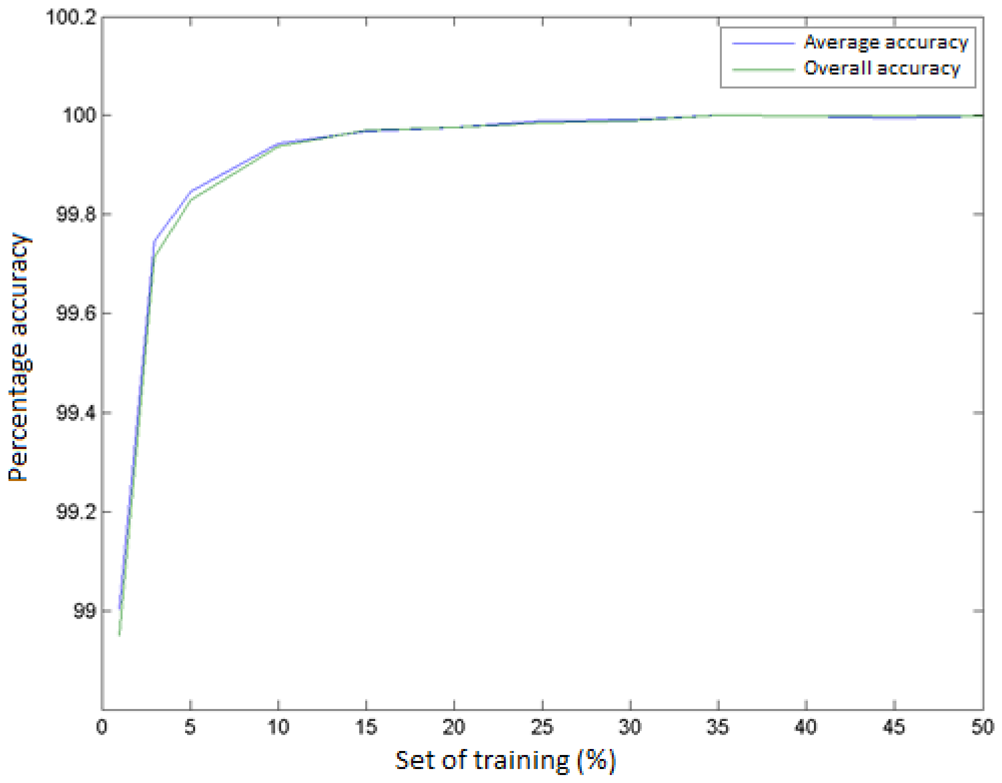


*Figure 5.41: AA (%) and OA (%) using process #1 with class 1 and class 2 for training dataset of different size.*

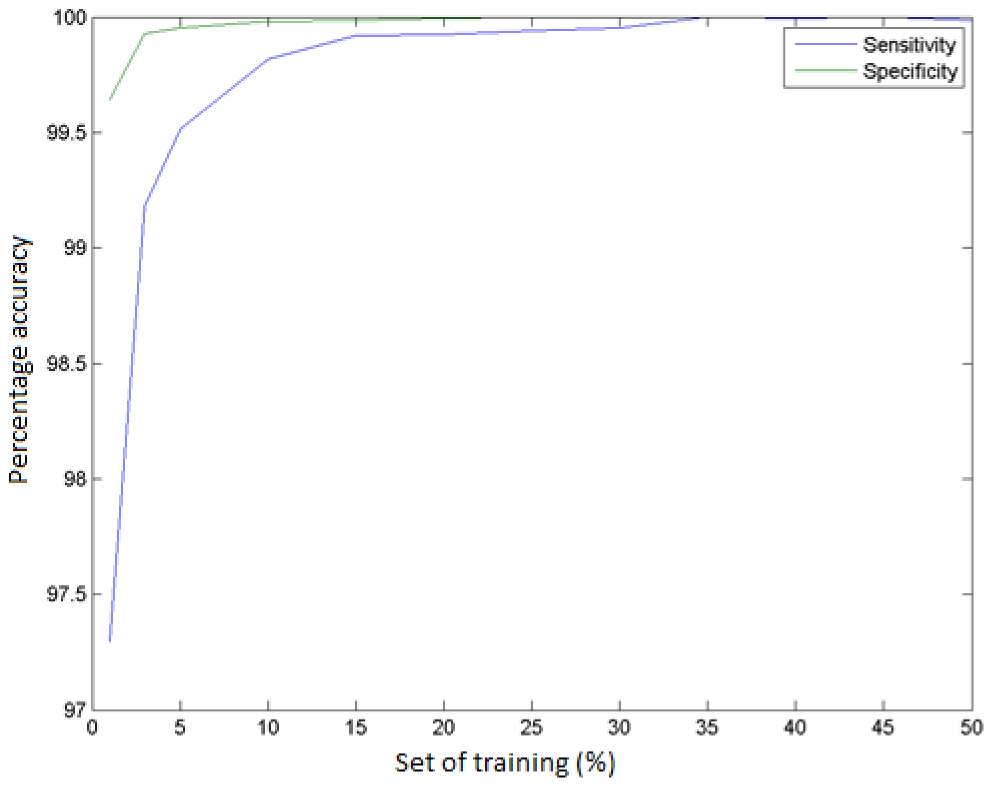


*Figure 5.42: Sensitivity (%) and specificity (%) using process #1 with class 1 and class 2 for training dataset of different size.*

- Process #2:

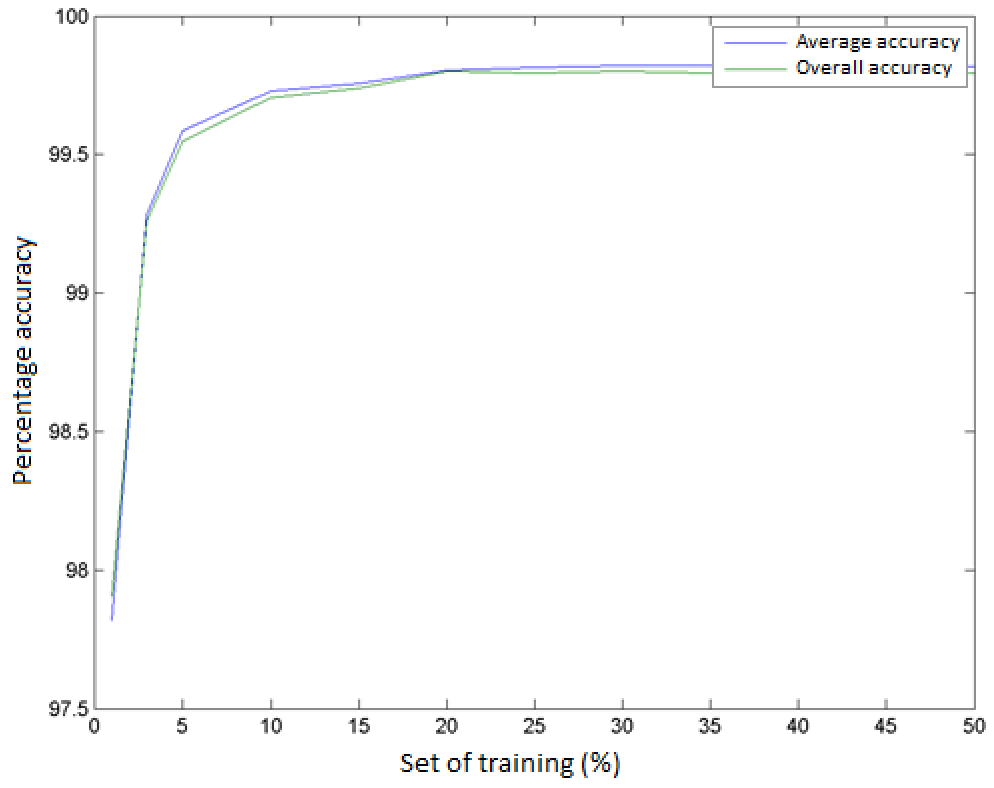


*Figure 5.43: AA (%) and OA (%) using process #2 with class 1 and class 2 for training dataset of different size.*

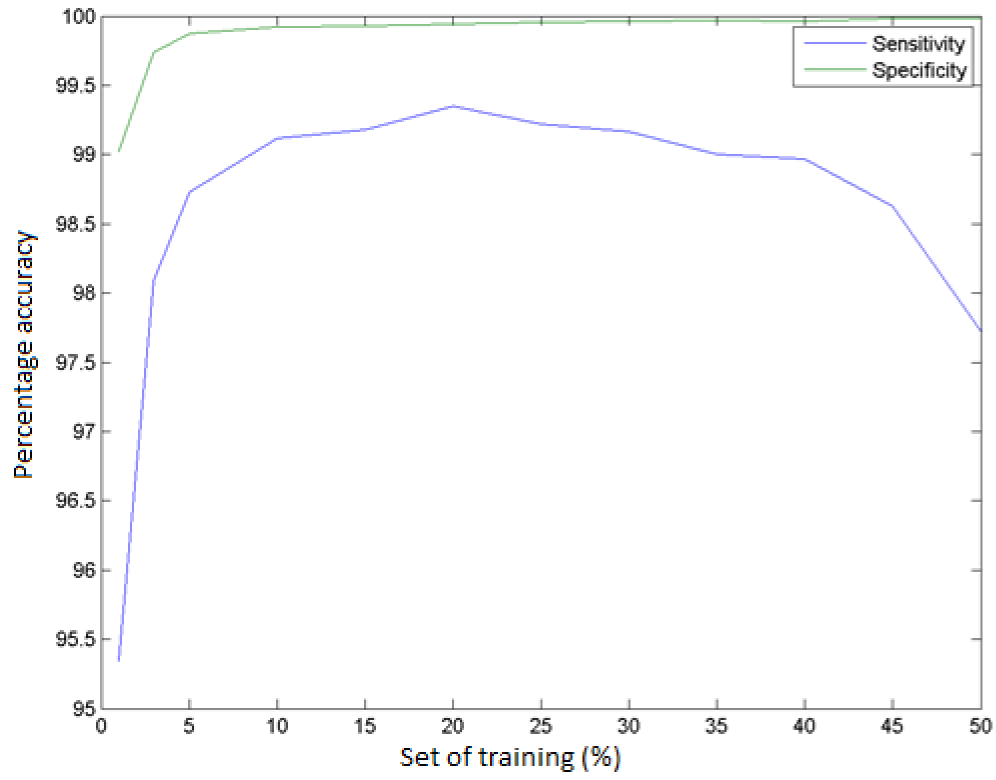


*Figure 5.44: Sensitivity (%) and specificity (%) using process #2 with class 1 and class 2 for training dataset of different size.*

- Process #4:

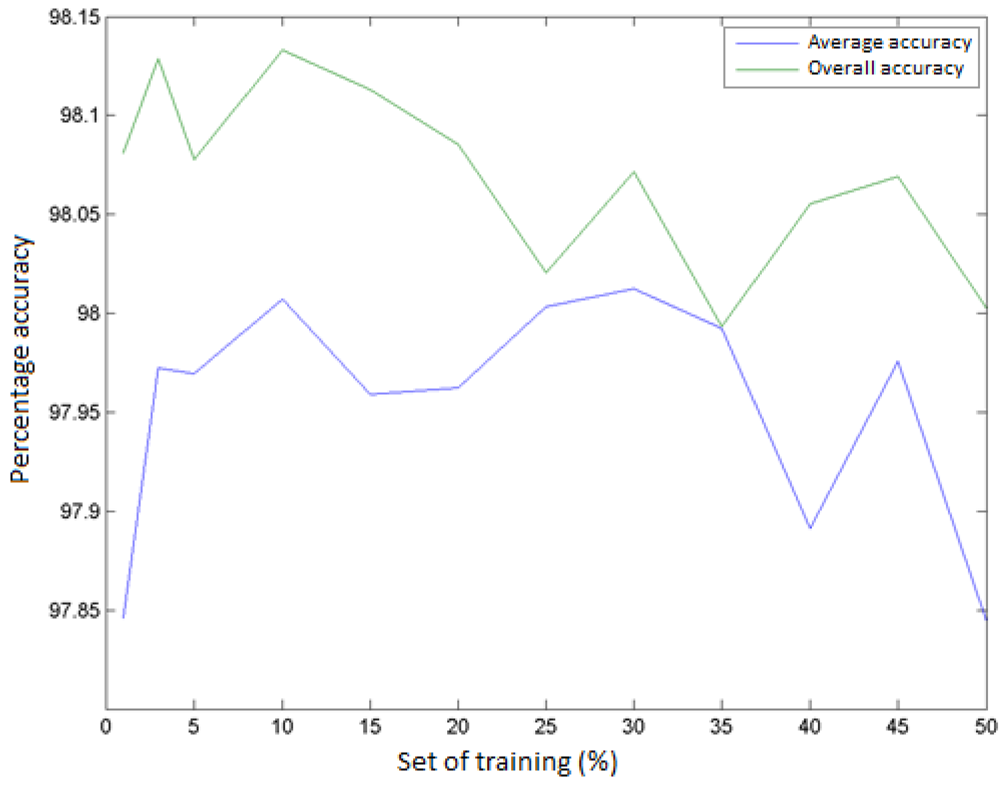


*Figure 5.45: AA (%) and OA (%) using process #4 with class 1 and class 2 for training dataset of different size.*

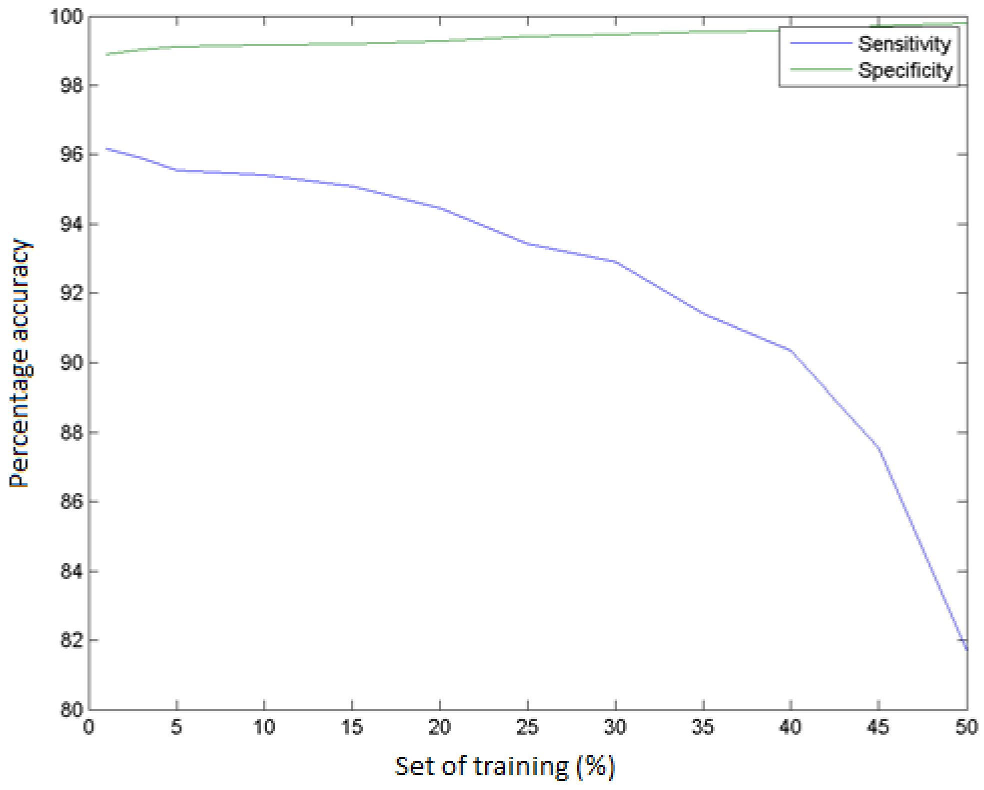


*Figure 5.46: Sensitivity (%) and specificity (%) using process #4 with class 1 and class 2 for training dataset of different size.*

As can be seen in figure 5.44, the problem is solved for the case of process #2. However in the case of process #4, as shown in figure 5.46, although the situation has improved retains the same behavior.

This set of experiments reveals that the problem came from the disparity in the size of classes. This means that Class 2 has many more samples than Class 1 and Class 1 has many more samples than Class 3. However, this fact was not taken into account when extracting training dataset, which were formed by the same number of samples in each class, regardless of the total number of samples of each class. This situation indicates the importance of the balance between the amounts of samples in each class in the training set. To verify this fact we repeated the experiments with two classes but this time using the same number of samples (and trainings) in each class. The results obtained are shown in the following charts:

- Process #1:

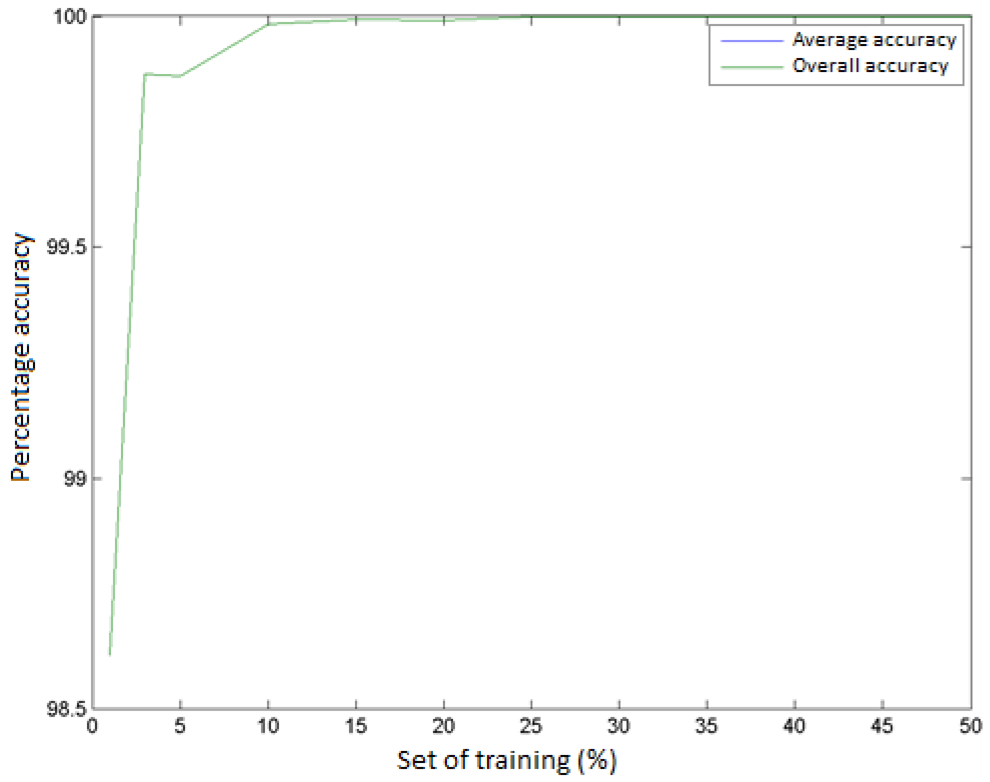


*Figure 5.47: AA (%) and OA (%) using process #1 with class 1 and class 2 of same size for training dataset of different size.*

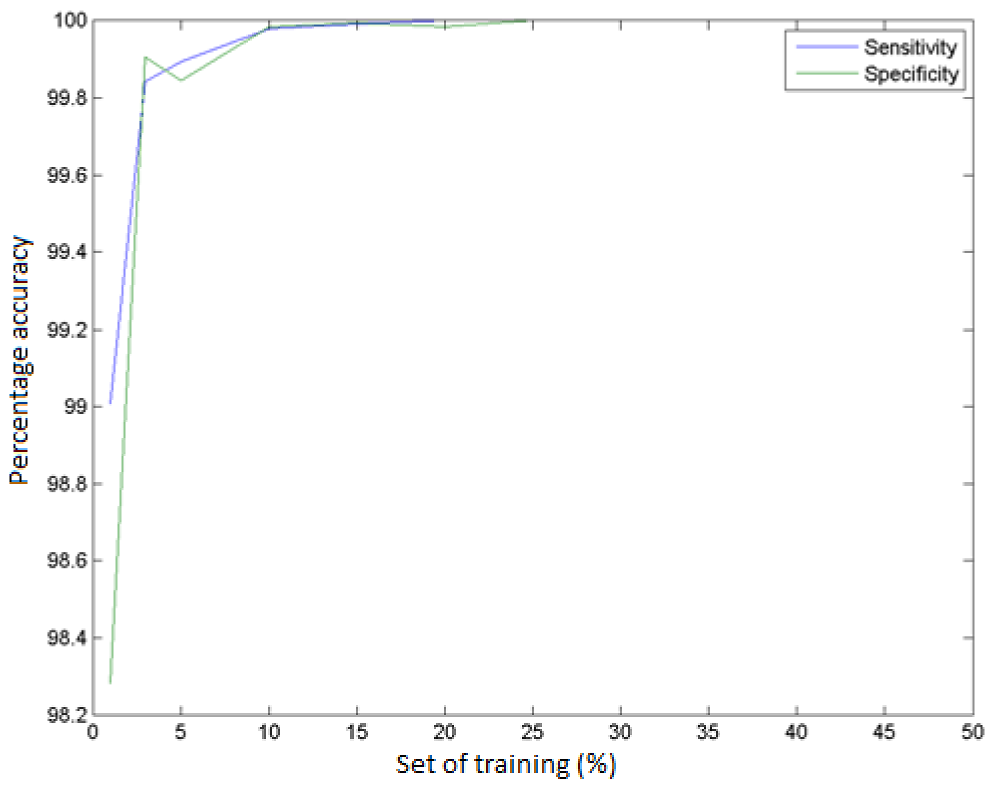


*Figure 5.48: Sensitivity (%) and specificity (%) using process #1 with class 1 and class 2 of same size for training dataset of different size.*

- Process #2:

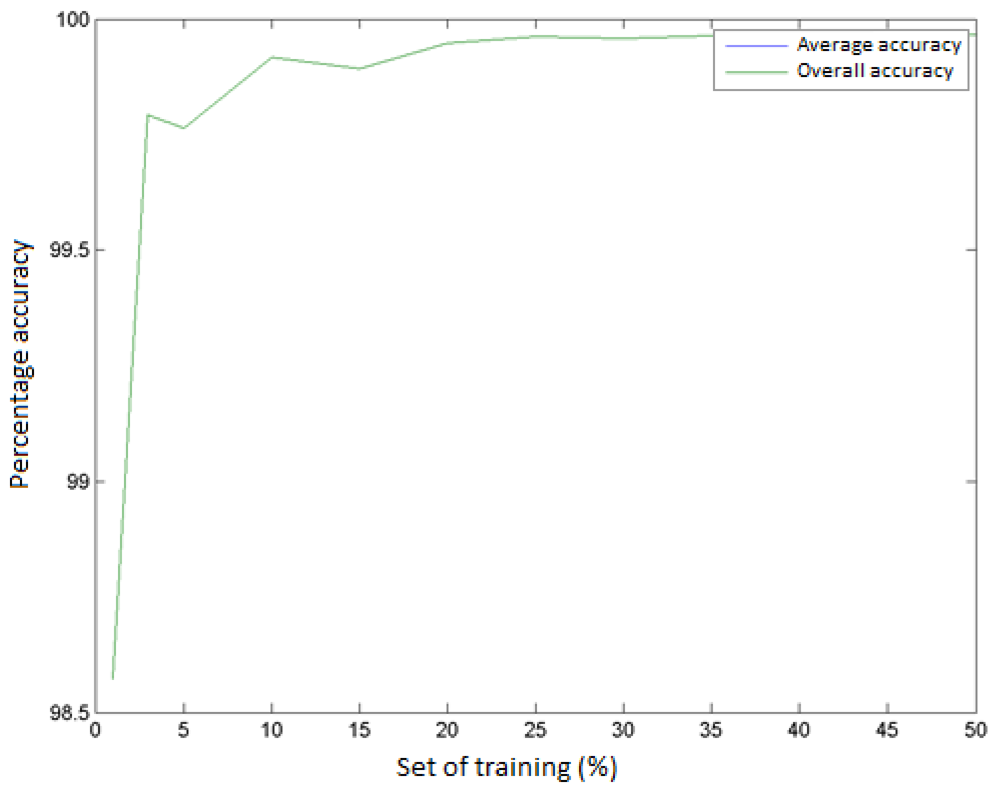


*Figure 5.49: AA (%) and OA (%) using process #2 with class 1 and class 2 of same size for training dataset of different size.*

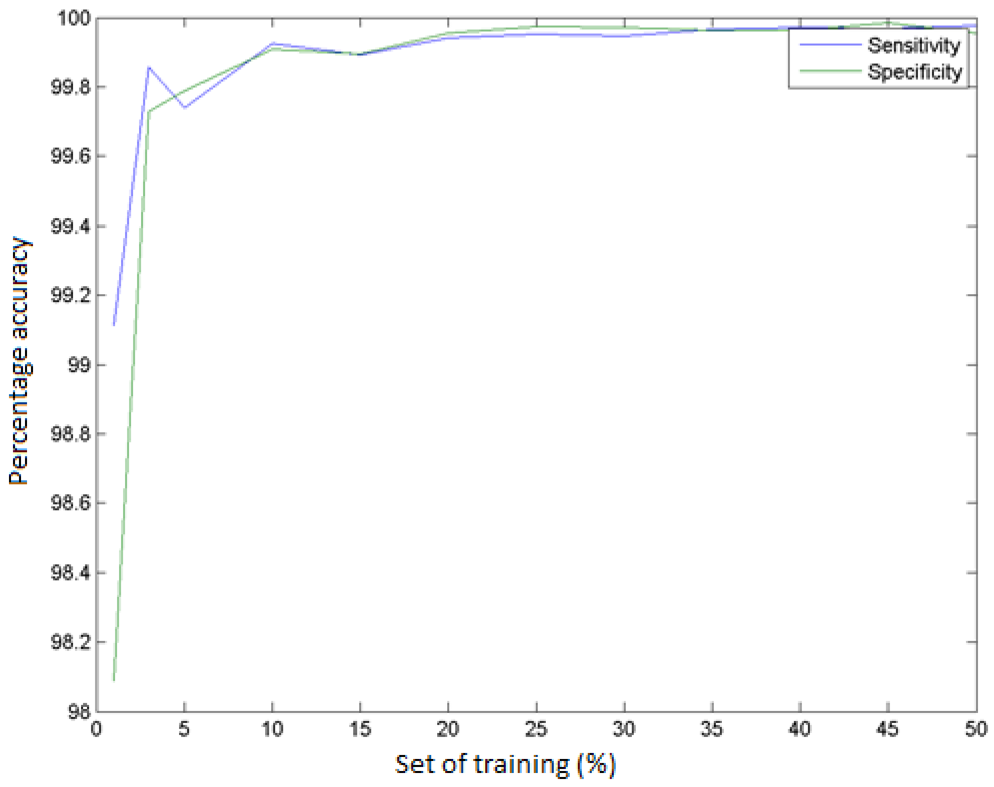


*Figure 5.50: Sensitivity (%) and specificity (%) using process #2 with class 1 and class 2 of same size for training dataset of different size.*

- Process #4:

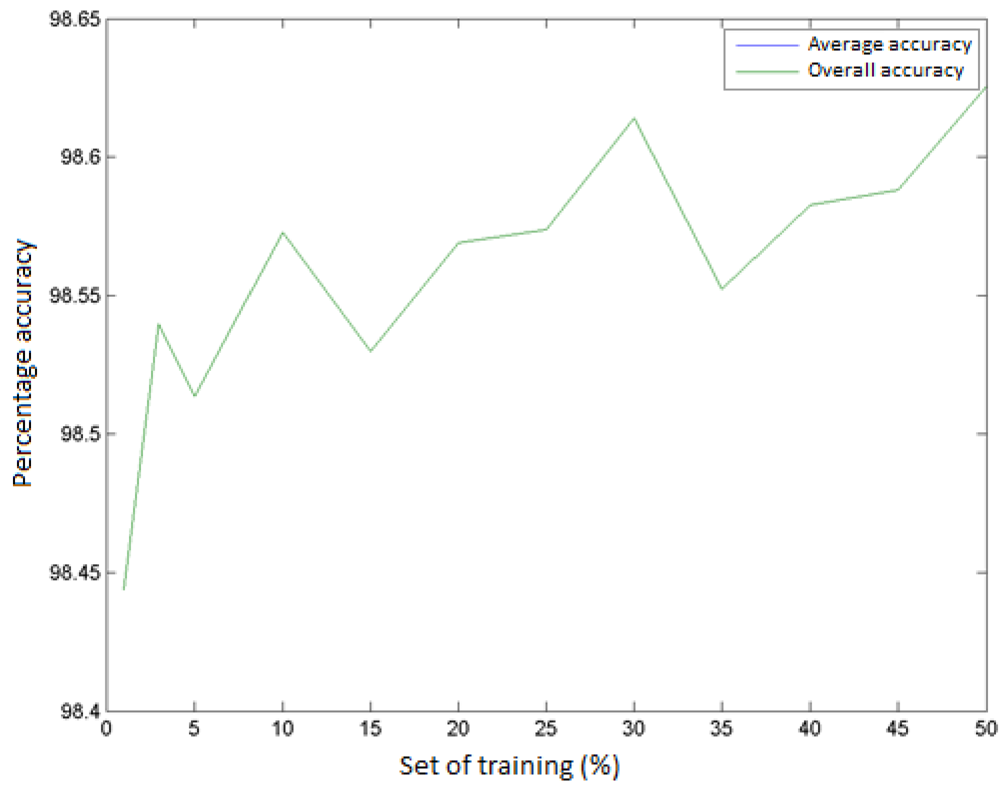


*Figure 5.51: AA (%) and OA (%) using process #4 with class 1 and class 2 of same size for training dataset of different size.*

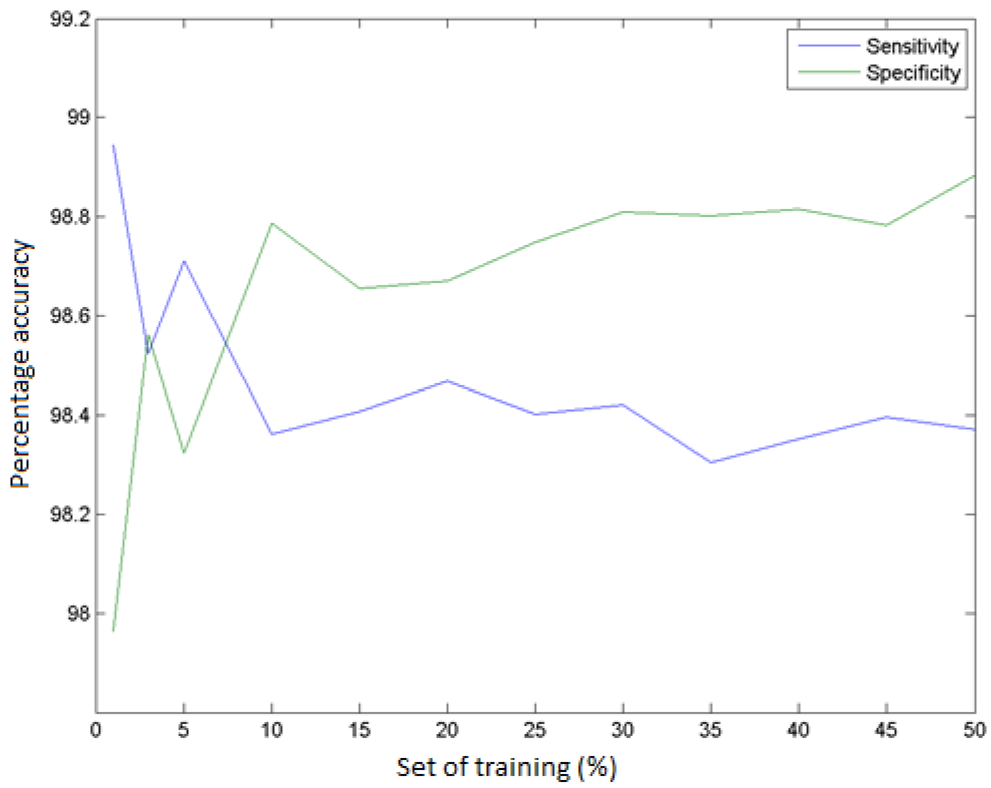


*Figure 5.52: Sensitivity (%) and specificity (%) using process #4 with class 1 and class 2 of same size for training dataset of different size.*

As can be seen in figure 5.52, the problem is solved for the case of process #4. So this problem is attributable to the disparity between classes mentioned above. Below the map of abundances for the latter case is shown, where we can appreciate graphically the correct separation between the two classes:

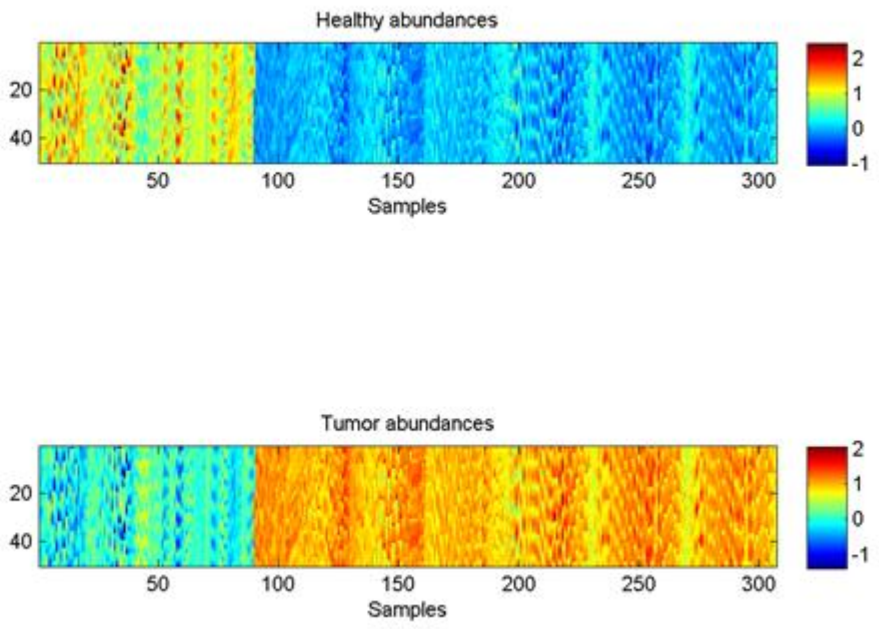


Figure 5.53: Abundance map using process #4 with 2 classes.

Moreover, from the previous graphs we decided that the training formed by 10% of the total samples meets the requirements our experiments, since increasing its size does not significantly improve performance and the computing time would be increased.

Otherwise, in order to test how it affects the number of samples of one class in relation to select the training, two random trainings are proposed:

1. Training 1: It is formed by 10% of total samples randomly selected and divided equally between the number of classes, so that in this training set all classes have the same number of samples regardless of the total number of samples that class.
2. Training 2: It is formed by 10% of samples randomly selected of each class, so that in this training set each class have varying numbers of samples depending on the total number of samples that class.

Regarding the second objective of this section, it intends to apply guided trainings as an alternative method to random trainings discussed above, in order to improve the results obtained. These guided trainings are formed by selecting specific pixels to be included in the trainings instead of be formed randomly. The process to form the guided trainings consists of get "the best pixels" of each class. The selection criteria of these pixels is, in the case of process #1 and #2, to calculate the centroid of each class and to take the most similar pixels to the centroid as part of training set for that class, for this the spectral angles between different pixels are calculated. On the contrary, the most separate pixels from its centroid are considered "the worst pixels". In the case of process #4, the training sets are formed by the pixels that have greater abundance of each class, considered as "the best pixels" of each class. Conversely, the pixels with lower abundance of a class are considered "the worst pixel" of that class, i.e. the pixels which provide a very small portion of this pure material. With the application of these methods it is intended to obtain the purest pixels which are expected to be the most representative of each class. To achieve this objective, the following methods are proposed to extract the training sets in a guided way:

3. Training 3: It is formed by 10% of "best" total samples divided equally between the number of classes, so that in this training set all classes have

the same number of samples regardless of the total number of samples that class.

4. Training 4: It is formed by 10% of “best” samples of each class, so that in this training set each class have varying numbers of samples depending on the total number of samples that class.
5. Training 5: It is formed by 10% of “best” samples of each class, then the number of samples in each class are equalized by averaging the samples in classes with larger number of samples, so that in this training set all classes have the same number of samples.

These random training sets will be used in future sections.

The results obtained for the main processing chains that best results achieved in process #1, #2 and #4 are:

| | **Process #1** | | **Process #2** | | **Process #4** | |
|---|---|---|---|---|---|---|
| **Training type** | **Training 1** | **Training 2** | **Training 1** | **Training 2** | **Training 1** | **Training 2** |
| **Average** | **98.27% (0.0018)** | 97.27% (0.0036) | **94.78% (0.0054)** | 91.84% (0.0062) | **91.79% (0)** | 88.94% (0.0038) |
| **Overall** | 97.55% (0.0028) | **98.12% (0.0017)** | 93.34% (0.0080) | **94.32% (0.0023)** | 89.55% (0.0020) | **91.89% (0.0012)** |
| **Kappa** | 0.9558 (0.0049) | **0.9668 (0.0031)** | 0.8826 (0.0134) | **0.8989 (0.0043)** | 0.8192 (0.0029) | **0.8555 (0.0020)** |

Table 5.35: Results of applying different ways to extract randomly trainings using the main processing chains.

| **Process #1** | **Training 3** | **Training 4** | **Training 5** |
|---|---|---|---|
| **AA** | **68.61%** | 66.11% | 61.45% |
| **OA** | **68.99%** | 68.96% | 61.40% |
| **Kappa** | 0.4851 | **0.4855** | 0.3824 |

Table 5.36: Results obtained using guided training in process #1.

| **Process #2** | **Training 3** | **Training 4** | **Training 5** |
|---|---|---|---|
| **AA** | **90.19%** | 89.28% | 86.29% |
| **OA** | 88.63% | **90.22%** | 84.06% |
| **Kappa** | 0.8019 | **0.8297** | 0.7365 |

Table 5.37: Results obtained using guided training in process #2.

| **Process #4** | **Training 3** | **Training 4** | **Training 5** |
|---|---|---|---|
| **AA** | 86.66% | 87.73% | **88.27%** |
| **OA** | 79.87% | 84.30% | **85.03%** |
| **Kappa** | 0.6765 | 0.7428 | **0.7532** |

Table 5.38: Results obtained using guided training in process #4.

As shown in table 5.55, significant results as to opt for one of the two methods are not found.

In the case of table 5.56, table 5.57 and table 5.58 only one classification is performed instead of 10, because it cannot be extracted several different training from guided mode since the best pixels are selected. Thus the standard deviation is not shown in these results.

As it can be seen, the best results are obtained by the process #2. However, these results are worse than those obtained using random training, because these pixels do not represent well its class. In order to demonstrate this problem, we present two variants of guided training 2:

6. Training 6: It is formed by 10% of less representative samples of each class.
7. Training 7: It is formed by 5% of more representative samples of each class and 5% of less representative samples of each class.

The results obtained for this test using process #1 are shown in table 5.59:

| Process #1 | Training 4 | Training 6 | Training 7 |
|---|---|---|---|
| **AA** | 66.11% | 88.16% | **89.38%** |
| **OA** | 68.96% | 81.57% | **93.09%** |
| **Kappa** | 0.4855 | 0.7100 | **0.8755** |

Table 5.39: Results obtained using variants of guided training 2 in process #1.

As it can be observed, the results obtained using training 6 are better than the results obtained using training 4, because the set of samples to classify has more samples which are identified by what we call "the worst pixels" that by what we call "the best pixels". However, the best results are obtained by combining "good pixels" with "bad pixels". This occurs because the training set should be as representative as possible of the set of samples to be classified, and the best way to get that is to have a variety of sample types.

Finally, for the purpose of try to exploit the success of random trainings with the idea of choosing the most representative pixels in a data set, it is intended to combine random trainings with guided trainings. For this purpose, it is applied one of the random trainings explained above but taking 20% of the samples rather than 10% of the samples. Guided trainings are applied to random training sets created previously taking half of its samples, so that the final training sets are formed by 10% of the total samples taken of different ways. Once again it is tested using trainings 4, 6 and 7. The results are as follows:

| Process #1 | Training 1 (20%) | | | Training 2 (20%) | | |
|---|---|---|---|---|---|---|
| | **Training 4** | **Training 6** | **Training 7** | **Training 4** | **Training 6** | **Training 7** |
| **AA** | 95.82% (0.0056) | 94.25% (0.0056) | **97.72% (0.0021)** | 94.45% (0.0063) | 94.65% (0.0027) | 96.73% (0.0052) |
| **OA** | 95.23% (0.0081) | 91.09% (0.0099) | 97.29% (0.0028) | 95.36% (0.0070) | 93.18% (0.0050) | **97.61% (0.0027)** |
| **Kappa** | 0.9146 (0.0139) | 0.8469 (0.0158) | 0.9510 (0.0048) | 0.9185 (0.0119) | 0.8840 (0.0079) | **0.9578 (0.0049)** |

Table 5.40: Results using Random-Guided trainings in process #1.

| Process #2 | Training 1 (20%) | | | Training 2 (20%) | | |
|---|---|---|---|---|---|---|
| | Training 4 | Training 6 | Training 7 | Training 4 | Training 6 | Training 7 |
| **AA** | 93.21% (0.0026) | **93.35% (0.0048)** | 92.15% (0.0028) | 92.84% (0.0088) | 89.34% (0.0094) | 90.68% (0.0035) |
| **OA** | 91.97% (0.0071) | 91.82% (0.0057) | 91.35% (0.0028) | 93.08% (0.0090) | **93.18% (0.0036)** | 92.89% (0.0035) |
| **Kappa** | 0.8589 (0.0114) | 0.8566 (0.0094) | 0.8474 (0.0046) | **0.8802 (0.0151)** | 0.8772 (0.0070) | 0.8743 (0.0059) |

Table 5.41: Results using Random-Guided trainings in process #2.

| Process #4 | Training 1 (20%) | | | Training 2 (20%) | | |
|---|---|---|---|---|---|---|
| | Training 4 | Training 6 | Training 7 | Training 4 | Training 6 | Training 7 |
| **AA** | 84.48% (0.0080) | 85.05% (0.0091) | **91.50% (0.0042)** | 80.73% (0.0092) | 84.27% (0.0107) | 89.27% (0.0085) |
| **OA** | 85.41% (0.0092) | 79.43% (0.0115) | 89.36% (0.0043) | 86.45% (0.0045) | 83.67% (0.0081) | **91.70% (0.0043)** |
| **Kappa** | 0.7376 (0.0153) | 0.6665 (0.0078) | 0.8159 (0.0071) | 0.7489 (0.0091) | 0.7254 (0.0136) | **0.8531 (0.0080)** |

Table 5.42: Results using Random-Guided trainings in process #4.

According to expected, the results have improved significantly. Nevertheless, it still provides better results using training sets obtained of purely random way.

In conclusion of this section, the best possible results are obtained when the set of samples to classify is duly represented by the training set. It is achieved when the training set is formed by samples as diverse as possible.

### 5.1.6. Results of patient simulation using SVM in brain cancer detection

The classification systems work extracting a set of training samples and then classifying the rest of samples. However, in the final system proposed in this dissertation it is intended to classify one patient using the training formed by the samples of other patients previously classified.

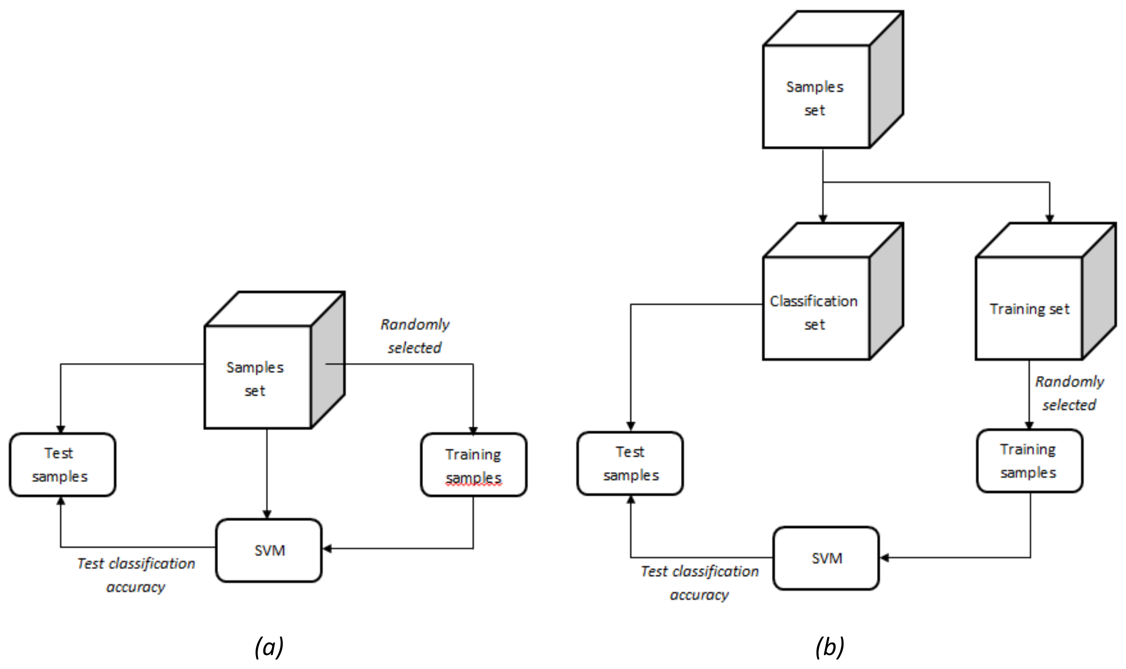


*Figure 5.54: (a) Conventional classification system and (b) Final classification system proposed.*

In this section is intended to simulate this situation with *ex vivo* samples, as a prior step before to start working in a real situation with *in vivo* samples. For this reason, it was necessary to change the inner workings of SVM, which is ready to work classifying the dataset from which the training set was obtained. In order to simulate the situation in question, the *ex vivo* samples dataset was divided into two subsets, one of them to obtain the training set and the other to perform the classification. Then, the total samples have been divided in order to simulate several patients and these divisions were made as follows:

1. Patient simulation 1: 25% of total samples to obtain the training and the remaining 75% to classify.
2. Patient simulation 2: 50% of total samples to obtain the training and the remaining 50% to classify.
3. Patient simulation 3: 75% of total samples to obtain the training and the remaining 25% to classify.

The training sets are obtained applying random training 1 to subset for training, obtaining training sets with a size equivalent to 10% of samples to classify. The results for this experiment are shown below:

| Process #1 | Patient simulation 1 | Patient simulation 2 | Patient simulation 3 |
|---|---|---|---|
| **AA** | 95.82% (0.0053) | 97.31% (0.0034) | 97.78% (0.0014) |
| **OA** | 94.65% (0.0077) | 96.28% (0.0046) | 97.15% (0.0029) |
| **Kappa** | 0.9084 (0.0127) | 0.9359 (0.0077) | 0.9506 (0.0048) |

*Table 5.43: Results patient simulation using process #1.*

| Process #2 | Patient simulation 1 | Patient simulation 2 | Patient simulation 3 |
|---|---|---|---|
| **AA** | 92.96% (0.0054) | 94.34% (0.0066) | 94.58% (0.0056) |
| **OA** | 91.45% (0.0112) | 93.45% (0.0061) | 93.22% (0.0057) |
| **Kappa** | 0.8552 (0.0174) | 0.8879 (0.0100) | 0.8846 (0.0094) |

*Table 5.44: Results patient simulation using process #2.*

| Process #4 | Patient simulation 1 | Patient simulation 2 | Patient simulation 3 |
|---|---|---|---|
| **AA** | 91.55% (0.0024) | 91.62% (0.0016) | 91.82% (0.0036) |
| **OA** | 89.89% (0.0059) | 89.90% (0.0045) | 89.74% (0.0062) |
| **Kappa** | 0.8296 (0.0086) | 0.8298 (0.0066) | 0.8279 (0.0095) |

*Table 5.45: Results patient simulation using process #4.*

As we can see, the results are similar to the corresponding processing chains without applying patient simulation. Evidently, the results improve as the training subset increases and the subset of samples to classify decreases. These results are encouraging for testing the following processing chain based on the same procedure as for this chain but using the set of *in vivo* samples.

### 5.1.7. Results of Consolidation of processing chains tested using *in vivo* data in brain cancer detection

In this last section it is intended to test the best chains but using *in vivo* samples. We have 3 different classes: healthy, tumor and vein, and five different patients: patient 4, 5, 7, 8 and 10 associated with the number of operation that they belong. In addition we use two different cameras to take the samples: VNIR and NIR. Along this section we will work with only two classes.

The process starts performing a classification system of each patient separately following by the conventional method of classification and finally without any mixing between patients. This means that each patient is trained and classified using his or her own samples. First we make a short test to check the results to classify tumor and vein using process #1. This is because its spectral signatures are very similar and there are fears that it may be difficult to differentiate between tumor and vein when classifying. The results are shown below:

| **Patient 5 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 99.30% (0.0068) | **99.34% (0.0067)** | 98.00% (0.0211) | 96.46% (0.0492) |
| **OA** | 99.16% (0.0042) | **99.40% (0.0040)** | 96.30% (0.1949) | 96.30% (0.0302) |
| **Kappa** | 0.9615 (0.0188) | **0.9782 (0.0143)** | 0.8151 (0.0390) | 0.8419 (0.1208) |
| **Sensitivity** | 94.03% (0.0372) | **97.19% (0.0273)** | 75.00% (0.2635) | 80.50% (0.1787) |
| **Specificity** | 99.93% (0.0022) | 99.86% (0.0030) | **100% (0)** | 99.60% (0.0126) |

*Table 5.46: Results of vein vs tumor in patient 5 using process #1.*

| **Patient 7 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 99.88% (0) | **99.92% (0)** | 95.20% (0.0459) | 95.09% (0.0407) |
| **OA** | 99.87% (0) | **99.92% (0)** | 95.48% (0.0404) | 94.88% (0.0502) |
| **Kappa** | 0.9973 (0) | **0.9983 (0.0012)** | 0.9046 (0.0867) | 0.8966 (0.0979) |
| **Sensitivity** | 99.74% (0) | **99.86% (0.0015)** | 96.37% (0.0479) | 97.40% (0.0158) |
| **Specificity** | **99.96% (0)** | **99.96% (0)** | 94.94% (0.0580) | 92.87% (0.1002) |

*Table 5.47: Results of vein vs tumor in patient 7 using process #1.*

| **Patient 8 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **99.69% (0.0066)** | 99.31% (0.0037) |
| **OA** | 99.51% (0.0121) | **99.78% (0)** |
| **Kappa** | 0.9699 (0.0700) | **0.9895 (0.0034)** |
| **Sensitivity** | 95.34% (0.1051) | **99.47% (0.0028)** |
| **Specificity** | **99.99% (0)** | 99.83% (0.0010) |

*Table 5.48: Results of vein vs tumor in patient 8 using process #1.*

The experimental results are very competitive differentiating between tumor and vein in the classification step. As it can be seen in table 5.46, 5.47 and 5.48, it has been used only patients 5, 7 and 8, because these are the only patients from which we have been able to extract samples from the vein class. Furthermore, it should be noted for future research that there are no samples obtained with type NIR camera for patient 8. From now on, all the classifications of this section will be conducted with healthy and tumor classes.

Below, patients are classified independently, using the main processing chains tested until now.

| **Patient 4 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 99.94% (0) | **99.96% (0)** | 96.94% (0.0268) | 96.20% (0.0267) |
| **OA** | 99.93% (0) | **99.97% (0)** | 96.67% (0.0314) | 96.45% (0.0238) |
| **Kappa** | 0.9986 (0.0018) | **0.9993 (0.0015)** | 0.9319 (0.0631) | 0.9268 (0.0493) |
| **Sensitivity** | 99.87% (0.0022) | **100% (0)** | 98.95% (0.0222) | 96.42% (0.0415) |
| **Specificity** | **99.97% (0)** | 99.95% (0.0011) | 94.26% (0.0709) | 97.24% (0.0476) |

*Table 5.49: Results of healthy vs tumor in patient 4 using process #1.*

| **Patient 4 Process #4** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | **99.89% (0)** | **99.89% (0)** | 95.42% (0.0293) | 94.10% (0.0380) |
| **OA** | 99.87% (0) | **99.88% (0)** | 95.67% (0.0316) | 94.52% (0.0342) |
| **Kappa** | 0.9971 (0.0015) | **0.9975 (0.0017)** | 0.9101 (0.0642) | 0.8865 (0.0711) |
| **Sensitivity** | 99.64% (0.0019) | **99.78% (0.0023)** | 96.34% (0.0346) | 94.56% (0.0573) |
| **Specificity** | **100% (0)** | 99.95% (0.0011) | 95.74% (0.0739) | 96.08% (0.0688) |

*Table 5.50: Results of healthy vs tumor in patient 4 using process #4.*

| **Patient 5 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 99.83% (0.0019) | **99.91% (0.0015)** | 98.24% (0.0117) | 97.73% (0.0116) |
| **OA** | 99.81% (0.0022) | **99.89% (0.0019)** | 98.24% (0.0111) | 97.65% (0.0124) |
| **Kappa** | 0.9959 (0.0048) | **0.9977 (0.0040)** | 0.9646 (0.0224) | 0.9529 (0.0247) |
| **Sensitivity** | 99.96% (0.0014) | **100% (0)** | 98.60% (0.0234) | 99.29% (0.0151) |
| **Specificity** | 99.56% (0.0062) | **99.72% (0.0049)** | 98.00% (0.0211) | 96.09% (0.0314) |

*Table 5.51: Results of healthy vs tumor in patient 5 using process #1.*

| **Patient 5 Process #4** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 99.66% (0.0031) | **99.73% (0.0012)** | 97.45% (0.0096) | 98.29% (0.0061) |
| **OA** | 99.64% (0.0039) | **99.75% (0)** | 97.45% (0.0095) | 98.24% (0.0062) |
| **Kappa** | 0.9924 (0.0083) | **0.9948 (0.0018)** | 0.9489 (0.0190) | 0.9646 (0.0124) |
| **Sensitivity** | **99.83% (0.0022)** | 99.78% (0.0023) | 97.82% (0.0188) | 99.29% (0.0151) |
| **Specificity** | 99.35% (0.0109) | **99.72% (0.0037)** | 97.15% (0.0197) | 97.20% (0.0193) |

*Table 5.52: Results of healthy vs tumor in patient 5 using process #4.*

| **Patient 7 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | **99.27% (0.0025)** | 99.24% (0.0031) | 96.14% (0.0394) | 94.87% (0.0426) |
| **OA** | **99.61% (0)** | 99.57% (0.0015) | 96.49% (0.0336) | 95.38% (0.0378) |
| **Kappa** | **0.9896 (0.0026)** | 0.9892 (0.0038) | 0.9274 (0.0704) | 0.9047 (0.0785) |
| **Sensitivity** | 99.89% (0.0024) | **99.95% (0.0016)** | 95.72% (0.0541) | 93.64% (0.0555) |
| **Specificity** | **99.52% (0.0019)** | 99.43% (0.0025) | 98.25% (0.0281) | 98.46% (0.0264) |

*Table 5.53: Results of healthy vs tumor in patient 7 using process #1.*

| **Patient 7 Process #4** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 84.51% (0.0122) | 84.30% (0.0082) | 89.02% (0.0164) | **89.19% (0.0285)** |
| **OA** | **92.10% (0.0061)** | 91.21% (0.0045) | 90.26% (0.0140) | 90.26% (0.0243) |
| **Kappa** | 0.7671 (0.0198) | 0.7578 (0.0136) | 0.7968 (0.0301) | **0.7979 (0.0512)** |
| **Sensitivity** | 99.26% (0.0026) | **99.40% (0.0031)** | 86.91% (0.0190) | 87.02% (0.0404) |
| **Specificity** | 90.58% (0.0069) | 89.26% (0.0051) | **96.36% (0.0014)** | 96.22% (0.0172) |

*Table 5.54: Results of healthy vs tumor in patient 7 using process #4.*

| **Patient 8 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 99.85% (0.0012) | **99.92% (0)** |
| **OA** | 99.82% (0.0017) | **99.91% (0)** |
| **Kappa** | 0.9957 (0.0040) | **0.9979 (0)** |
| **Sensitivity** | 99.45% (0.0056) | **99.77% (0.0015)** |
| **Specificity** | **99.97% (0)** | **99.97% (0)** |

*Table 5.55: Results of healthy vs tumor in patient 8 using process #1.*

| **Patient 8 Process #2** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 99.45% (0.0058) | **99.52% (0.0030)** |
| **OA** | 99.38% (0.0079) | **99.60% (0.0020)** |
| **Kappa** | 0.9853 (0.0187) | **0.9907 (0.0047)** |
| **Sensitivity** | 98.35% (0.0257) | **99.43% (0.0038)** |
| **Specificity** | **99.85% (0.0023)** | 99.68% (0.0028) |

*Table 5.56: Results of healthy vs tumor in patient 8 using process #2.*

| **Patient 8 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **99.51% (0.0021)** | 99.10% (0.0051) |
| **OA** | **99.65% (0.0012)** | 99.39% (0.0031) |
| **Kappa** | **0.9916 (0.0028)** | 0.9859 (0.0073) |
| **Sensitivity** | 99.63% (0.0021) | **99.75% (0.0013)** |
| **Specificity** | **99.66% (0.0019)** | 99.24% (0.0046) |

*Table 5.57: Results of healthy vs tumor in patient 8 using process #4.*

| **Patient 10 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | **99.90% (0.0021)** | 99.21% (0.0087) | 93.77% (0.0800) | 97.80% (0.0382) |
| **OA** | **99.84% (0.0034)** | 99.64% (0.0040) | 93.48% (0.0899) | 97.92% (0.0354) |
| **Kappa** | **0.9950 (0.0105)** | 0.9897 (0.0114) | 0.8719 (0.1694) | 0.9576 (0.0722) |
| **Sensitivity** | 99.22% (0.0165) | **100% (0)** | 92.40% (0.1317) | 99.17% (0.0264) |
| **Specificity** | **100% (0)** | 99.54% (0.0051) | 97.24% (0.0476) | 97.33% (0.0562) |

*Table 5.58: Results of healthy vs tumor in patient 10 using process #1.*

| Patient 10 Process #4 | VNIR | | NIR | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | **99.98% (0)** | 99.82% (0.0037) | 99.62% (0.0122) | 97.80% (0.0382) |
| **OA** | **99.96% (0.0013)** | 99.92% (0.0017) | 99.57% (0.0137) | 97.92% (0.0354) |
| **Kappa** | **0.9987 (0.0040)** | 0.9977 (0.0048) | 0.9913 (0.0277) | 0.9576 (0.0722) |
| **Sensitivity** | 99.80% (0.0063) | **100% (0)** | 99.09% (0.0287) | 99.17% (0.0264) |
| **Specificity** | **100% (0)** | 99.90% (0.0022) | **100% (0)** | 97.33% (0.0562) |

*Table 5.59: Results of healthy vs tumor in patient 10 using process #4.*

As shown before, the results using *in vivo* samples and classifying each patient independently are very accurate. It should be noted that using *in vivo* samples it has not been possible to apply process #2, and therefore, the dimensionality reduction techniques. This is because the ICA algorithm that we have available us working only with data in which the number of samples exceeds the number of bands. As we have very few *in vivo* samples, this requirement is fulfilled only in the case of Patient 8, in which ICA has been applied reducing to 5 bands. Finally it is noteworthy that in this case, applying the unmixing techniques does not worsen the results as happened with *ex vivo* samples.

Heretofore, all classifications have been made using the same patient to train and to classify, since *ex vivo* samples belong all to a single patient and *in vivo* samples have been classified with each patient independently. From this moment, all tests have been conducted training from some patients and classifying different ones. So we have two different sets formed by different patients, one set for training and other one for classifying. These tests are very important because if we want to classify each new patient using the trainings sets of previous patients, we should know if the spectral signatures of different classes that we can find in the classification processes are similar between patients and others, since these spectral signatures could depend on patterns such as gender, race, age, etc.

Moreover, as it was explained in section 4.7, it is not possible to apply dimensionality reduction in two independent datasets and compare them, because the transformations are different and dependent on the dataset to which it applies.

The first experiment of this type involves training with a patient and making the classification using remaining patients. This means that the training set was obtained from a particular patient and the remaining patients were randomly mixed in another set of data to be classified. This experiment aims to determine if any of the patients is especially good to generate training sets for other patients. The results are shown below:

- Training is obtained from patient 4 and the rest of patients are classified.

| **Patient 4 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **73.00% (0.0014)** | 72.79% (0.0043) |
| **OA** | **74.99% (0)** | 74.85% (0.0029) |
| **Kappa** | **0.4505 (0.0025)** | 0.4468 (0.0076) |
| **Sensitivity** | **61.71% (0.0010)** | 61.57% (0.0031) |
| **Specificity** | **82.58% (0.0013)** | 82.40% (0.0037) |

*Table 5.60: Results of healthy vs tumor training patient 4 and classifying the rest of patients using process #1.*

| **Patient 4 Process #4** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 62.24% (0.0014) | 62.10% (0.0014) | 76.47% (0.0906) | **78.98% (0.0710)** |
| **OA** | 68.11% (0) | 68.10% (0.0013) | 77.08% (0.0922) | **79.77% (0.0652)** |
| **Kappa** | 0.2550 (0.0025) | 0.2528 (0.0030) | 0.5355 (0.1828) | **0.5865 (0.1390)** |
| **Sensitivity** | 52.93% (0) | 52.94% (0.0023) | 75.67% (0.0780) | **76.99% (0.0807)** |
| **Specificity** | 74.06% (0.0011) | 73.94% (0) | 85.47% (0.1421) | **88.91% (0.0764)** |

*Table 5.61: Results of healthy vs tumor training patient 4 and classifying the rest of patients using process #4.*

- Training is obtained from patient 5 and the rest of patients are classified.

| **Patient 5 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **74.08% (0)** | 71.97% (0.0235) |
| **OA** | **76.75% (0)** | 75.44% (0.0146) |
| **Kappa** | **0.4709 (0)** | 0.4340 (0.0411) |
| **Sensitivity** | **61.81% (0)** | 60.20% (0.0181) |
| **Specificity** | **84.40% (0)** | 82.78% (0.0180) |

*5.62: Results of healthy vs tumor training patient 5 and classifying the rest of patients using process #1.*

| **Patient 5 Process #4** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 73.97% (0) | 73.93% (0.0028) | 80.78% (0.0048) | **80.80% (0.0055)** |
| **OA** | 76.46% (0.0011) | 76.45% (0.0037) | **82.09% (0.0048)** | **82.09% (0.0048)** |
| **Kappa** | 0.4666 (0.0020) | 0.4661 (0.0067) | **0.6301 (0.0097)** | **0.6303 (0.0103)** |
| **Sensitivity** | 61.21% (0.0019) | 61.21% (0.0064) | 77.57% (0.0067) | **77.67% (0.0072)** |
| **Specificity** | 84.45% (0) | 84.41% (0.0010) | **91.33% (0.0211)** | 90.94% (0.0119) |

*Table 5.63: Results of healthy vs tumor training patient 5 and classifying the rest of patients using process #4.*

- Training is obtained from patient 7 and the rest of patients are classified.

| **Patient 7 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **65.23% (0.0287)** | 64.70% (0.0217) |
| **OA** | **74.97% (0.0206)** | 74.60% (0.0156) |
| **Kappa** | **0.3584 (0.0629)** | 0.3473 (0.0474) |
| **Sensitivity** | 99.63% (0.0049) | **99.97% (0)** |
| **Specificity** | **71.95% (0.0165)** | 71.62% (0.0126) |

*Table 5.64: Results of healthy vs tumor training patient 7 and classifying the rest of patients using process #1.*

| **Patient 7 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **64.80% (0.0037)** | 62.26% (0.0071) |
| **OA** | **74.68% (0.0027)** | 72.85% (0.0051) |
| **Kappa** | **0.3500 (0.0082)** | 0.2937 (0.0158) |
| **Sensitivity** | **100% (0)** | **100% (0)** |
| **Specificity** | **71.66% (0.0021)** | 70.22% (0.0039) |

*Table 5.65: Results of healthy vs tumor training patient 7 and classifying the rest of patients using process #4.*

- Training is obtained from patient 8 and the rest of patients are classified.

| **Patient 8 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 60.83% (0.0140) | **62.32% (0.0088)** |
| **OA** | 51.35% (0.0142) | **53.00% (0.0107)** |
| **Kappa** | 0.1718 (0.0224) | **0.1964 (0.0148)** |
| **Sensitivity** | 43.02% (0.0081) | **43.93% (0.0057)** |
| **Specificity** | 93.98% (0.0478) | **96.03% (0.0225)** |

*Table 5.66: Results of healthy vs tumor training patient 8 and classifying the rest of patients using process #1.*

| **Patient 8 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 63.49% (0.0060) | **64.70% (0.0060)** |
| **OA** | 55.88% (0.0076) | **57.43% (0.0077)** |
| **Kappa** | 0.2218 (0.0107) | **0.2436 (0.0108)** |
| **Sensitivity** | 45.29% (0.0046) | **46.23% (0.0047)** |
| **Specificity** | 88.87% (0.0030) | **89.48% (0.0028)** |

*Table 5.67: Results of healthy vs tumor training patient 8 and classifying the rest of patients using process #4.*

- Training is obtained from patient 10 and the rest of patients are classified.

| **Patient 10 Process #1** | **VNIR** | **NIR** | |
|---|---|---|---|
| **Training type** | **All samples** | **Random training 1** | **Random training 2** |
| **AA** | **76.32% (0)** | 58.11% (0.0088) | 58.79% (0.0058) |
| **OA** | **77.62% (0)** | 54.71% (0.0194) | 56.63% (0.0234) |
| **Kappa** | **0.5178 (0)** | 0.1505 (0.0177) | 0.1671 (0.0159) |
| **Sensitivity** | 67.02% (0) | **81.26% (0.1115)** | 74.68% (0.1186) |
| **Specificity** | **84.07% (0)** | 49.63% (0.0143) | 51.13% (0.0181) |

*Table 5.68: Results of healthy vs tumor training patient 10 and classifying the rest of patients using process #1.*

| **Patient 10 Process #4** | **VNIR** | **NIR** | |
|---|---|---|---|
| **Training type** | **All samples** | **Random training 1** | **Random training 2** |
| **AA** | **77.86% (0)** | 55.24% (0.0054) | 55.14% (0.0049) |
| **OA** | **77.62% (0)** | 50.51% (0.0072) | 50.39% (0.0065) |
| **Kappa** | **0.5319 (0)** | 0.0946 (0.0100) | 0.0928 (0.0092) |
| **Sensitivity** | 65.12% (0) | 87.46% (0.0686) | **87.71% (0.0853)** |
| **Specificity** | **86.89% (0)** | 47.21% (0.0033) | 47.15% (0.0031) |

*Table 5.69: Results of healthy vs tumor training patient 10 and classifying the rest of patients using process #4.*

As it can be seen, the results have significantly worsened after mixing some patient with others. However, except for some bad values, in general the results are acceptable and we can also find high values. It is noteworthy that the results obtained with the VNIR camera are somewhat better than those obtained with the NIR camera. In many case the results obtained with the NIR camera have been very poor, obtaining that all samples belong to a single class or all healthy or all tumor and for this reason they have not been shown. This situation is being repeated in the following results and that is the reason because in some cases the results obtained are not shown. Moreover, in the case that the training is obtained from patient 10 and the rest of patients are classified using the VNIR camera, since the training sets are obtained as the 10% of the sample sets to classify, in this case there were not enough samples for covering this number so all samples were taken to form the training set. Finally, as in the previous experiments, to apply the unmixing techniques did not worsen the results.

As in none of the cases the results has been exceptionally good, we can conclude that we have no patient which is particularly good for generating training sets for other patients.

In the following experiment, only one patient is classified and the training sets are obtained from a dataset formed by the remaining patients mixed randomly. This would be a possible situation of operating mode of the system in real surgeries. The results are shown below:

- Patient 4 is classified and the training is obtained from the rest of patients.

| **Patient 4 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **85.00% (0.0647)** | 81.98% (0.0547) |
| **OA** | **81.60% (0.0797)** | 78.97% (0.0672) |
| **Kappa** | **0.6477 (0.1430)** | 0.5821 (0.1156) |
| **Sensitivity** | **69.01% (0.1024)** | 64.43% (0.0716) |
| **Specificity** | **99.87% (0.0040)** | 99.53% (0.0050) |

*Table 5.70: Results of healthy vs tumor classifying patient 4 and training from the rest of patients using process #1.*

| **Patient 4 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 98.68% (0.0140) | **99.50% (0.0038)** |
| **OA** | 98.37% (0.0173) | **99.44% (0.0033)** |
| **Kappa** | 0.9662 (0.0354) | **0.9883 (0.0069)** |
| **Sensitivity** | 96.10% (0.0386) | **98.85% (0.0080)** |
| **Specificity** | **100% (0)** | 99.83% (0.0053) |

*Table 5.71: Results of healthy vs tumor classifying patient 4 and training from the rest of patients using process #4.*

- Patient 5 is classified and the training is obtained from the rest of patients.

| **Patient 5 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **99.10% (0.0167)** | 96.83% (0.0781) |
| **OA** | **99.31% (0.0128)** | 97.55% (0.0596) |
| **Kappa** | **0.9853 (0.0276)** | 0.9437 (0.1400) |
| **Sensitivity** | **98.94% (0.0193)** | 96.91% (0.0734) |
| **Specificity** | **100% (0)** | 99.81% (0.0031) |

*Table 5.72: Results of healthy vs tumor classifying patient 5 and training from the rest of patients using process #1.*

| **Patient 5<br>Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **99.51% (0.0033)** | 99.37% (0.0027) |
| **OA** | **99.46% (0.0041)** | **99.46% (0.0025)** |
| **Kappa** | **0.9886 (0.0087)** | **0.9886 (0.0054)** |
| **Sensitivity** | **99.84% (0.0020)** | 99.37% (0.0042) |
| **Specificity** | 98.87% (0.0116) | **99.62% (0.0079)** |

*Table 5.73: Results of healthy vs tumor classifying patient 5 and training from the rest of patients using process #4.*

- Patient 7 is classified and the training is obtained from the rest of patients.

| **Patient 7<br>Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **47.90% (0.0465)** | 36.66% (0.1066) |
| **OA** | **33.55% (0.0212)** | 27.82% (0.0547) |
| **Kappa** | -0.0274 (0.0578) | -0.1810 (0.1484) |
| **Sensitivity** | **26.61% (0.0226)** | 20.10% (0.0640) |
| **Specificity** | **69.85% (0.1245)** | 53.39% (0.1399) |

*Table 5.74: Results of healthy vs tumor classifying patient 7 and training from the rest of patients using process #1.*

| **Patient 7<br>Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **32.70% (0.0417)** | 31.99% (0.0660) |
| **OA** | 24.26% (0.0267) | **25.12% (0.0376)** |
| **Kappa** | -0.2247 (0.0568) | -0.2409 (0.0942) |
| **Sensitivity** | **18.63% (0.0239)** | 17.72% (0.0412) |
| **Specificity** | 42.85% (0.0520) | **45.84% (0.0645)** |

*Table 5.75: Results of healthy vs tumor classifying patient 7 and training from the rest of patients using process #4.*

- Patient 8 is classified and the training is obtained from the rest of patients.

| **Patient 8 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **64.02% (0.0122)** | 60.57% (0.0366) |
| **OA** | **77.40% (0.0075)** | 75.25% (0.0230) |
| **Kappa** | **0.3480 (0.0272)** | 0.2669 (0.0886) |
| **Sensitivity** | 99.84% (0.0036) | **100% (0)** |
| **Specificity** | **75.24% (0.0063)** | 73.53% (0.0177) |

*Table 5.76: Results of healthy vs tumor classifying patient 8 and training from the rest of patients using process #1.*

| **Patient 8 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **62.35% (0.0059)** | 57.07% (0.0258) |
| **OA** | **76.36% (0.0037)** | 73.05% (0.0162) |
| **Kappa** | **0.3103 (0.0136)** | 0.1832 (0.0633) |
| **Sensitivity** | **100% (0)** | 100% (0) |
| **Specificity** | **74.38% (0.0030)** | 71.81% (0.0123) |

*Table 5.77: Results of healthy vs tumor classifying patient 8 and training from the rest of patients using process #4.*

- Patient 10 is classified and the training is obtained from the rest of patients.

| Patient 10 Process #1 | VNIR | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 80.42% (0.1705) | **85.51% (0.1852)** |
| **OA** | 69.68% (0.2641) | **77.56% (0.2867)** |
| **Kappa** | 0.5055 (0.3739) | **0.6499 (0.4193)** |
| **Sensitivity** | 55.33% (0.2944) | **69.37% (0.3416)** |
| **Specificity** | **100% (0)** | **100% (0)** |

*Table 5.78: Results of healthy vs tumor classifying patient 10 and training from the rest of patients using process #1.*

| Patient 10 Process #4 | VNIR | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **97.21% (0.0638)** | 82.14% (0.2525) |
| **OA** | **98.21% (0.0310)** | 91.94% (0.1140) |
| **Kappa** | **0.9454 (0.1000)** | 0.6912 (0.4367) |
| **Sensitivity** | **97.15% (0.0594)** | 100% (0) |
| **Specificity** | **98.79% (0.0338)** | 91.83% (0.1219) |

*Table 5.79: Results of healthy vs tumor classifying patient 10 and training from the rest of patients using process #4.*

The results obtained are quite diverse: for example the results for classifying patient 7 are very inaccurate, but the results for classifying patient 5 are highly accurate. Moreover, we can see that is difficult to obtain results from the NIR camera. Furthermore, in this experiment applying unmixing techniques not only did not worsen the results, but these are remarkably improved in some cases.

With the aim of trying to improve the results, the following experiments are proposed, in which a single patient is classified and the training set is obtained by mixing 10% of the samples from each of the remaining patients, in order to make the training set as diverse as possible. The results are shown below:

- Patient 4 is classified and the training is obtained from the rest of de patients equally.

| **Patient 4 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 78.53% (0.0631) | **82.49% (0.0400)** |
| **OA** | 73.63% (0.0775) | **78.52% (0.0489)** |
| **Kappa** | 0.5102 (0.1312) | **0.5907 (0.0848)** |
| **Sensitivity** | 60.22% (0.0780) | **64.64% (0.0528)** |
| **Specificity** | **99.96% (0.0014)** | 99.81% (0.0061) |

*Table 5.80: Results of healthy vs tumor classifying patient 4 and training from the rest of patients equally using process #1.*

| **Patient 4 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 75.05% (0.1349) | **84.63% (0.1085)** |
| **OA** | 69.34% (0.1657) | **81.11% (0.1333)** |
| **Kappa** | 0.4528 (0.2635) | **0.6468 (0.2269)** |
| **Sensitivity** | 58.68% (0.1411) | **70.04% (0.1412)** |
| **Specificity** | **100% (0)** | **100% (0)** |

*Table 5.81: Results of healthy vs tumor classifying patient 4 and training from the rest of patients equally using process #4.*

- Patient 5 is classified and the training is obtained from the rest of de patients equally.

| **Patient 5 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 99.57% (0.0057) | **99.63% (0.0045)** |
| **OA** | 99.53% (0.0072) | **99.71% (0.0034)** |
| **Kappa** | 0.9902 (0.0150) | **0.9938 (0.0073)** |
| **Sensitivity** | **99.84% (0.0020)** | 99.57% (0.0057) |
| **Specificity** | 99.08% (0.0185) | **99.94% (0.0020)** |

*Table 5.82: Results of healthy vs tumor classifying patient 5 and training from the rest of patients equally using process #1.*

| **Patient 5 Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **99.73% (0.0019)** | 99.64% (0.0011) |
| **OA** | **99.73% (0.0018)** | 99.68% (0.0012) |
| **Kappa** | **0.9943 (0.0038)** | 0.9933 (0.0025) |
| **Sensitivity** | **99.84% (0.0028)** | 99.68% (0.0025) |
| **Specificity** | 99.56% (0.0052) | **99.68% (0.0054)** |

*Table 5.83: Results of healthy vs tumor classifying patient 5 and training from the rest of patients equally using process #4.*

- Patient 7 is classified and the training is obtained from the rest of de patients equally.

| **Patient 7 Process #1** | **VNIR** | | **NIR** | |
|---|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** | **Random training 2** |
| **AA** | 56.82% (0.0191) | 47.71% (0.1189) | **72.91% (0.1626)** | 69.09% (0.1046) |
| **OA** | 38.85% (0.0108) | 33.55% (0.0675) | **70.11% (0.1780)** | 65.52% (0.1114) |
| **Kappa** | 0.0819 (0.0227) | -0.0346 (0.1547) | **0.4401 (0.3275)** | 0.3550 (0.1967) |
| **Sensitivity** | 30.84% (0.0086) | 26.17% (0.0632) | 92.00% (0.2024) | **92.31% (0.1538)** |
| **Specificity** | **94.38% (0.0698)** | 74.55% (0.2528) | 64.24% (0.2061) | 56.56% (0.0748) |

*Table 5.84: Results of healthy vs tumor classifying patient 7 and training from the rest of patients equally using process #1.*

| **Patient 7 Process #4** | **VNIR** | | **NIR** |
|---|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** | **Random training 1** |
| **AA** | 37.95% (0.0263) | 31.38% (0.0416) | 73.81% (0.0792) |
| **OA** | 28.08% (0.0176) | 23.70% (0.0280) | 70.50% (0.0892) |
| **Kappa** | -0.1553 (0.0348) | -0.2436 (0.0560) | 0.4455 (0.1595) |
| **Sensitivity** | 21.48% (0.0143) | 17.80% (0.0233) | 100% (0) |
| **Specificity** | 50.93% (0.0394) | 42.04% (0.0561) | 60.52% (0.0831) |

*Table 5.85: Results of healthy vs tumor classifying patient 7 and training from the rest of patients equally using process #4.*

- Patient 8 is classified and the training is obtained from the rest of de patients equally.

| **Patient 8 Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 61.12% (0.0285) | **63.96% (0.0120)** |
| **OA** | 75.59% (0.0179) | **77.37% (0.0075)** |
| **Kappa** | 0.2806 (0.0686) | **0.3468 (0.0270)** |
| **Sensitivity** | **100% (0)** | **100% (0)** |
| **Specificity** | 73.78% (0.0139) | **75.21% (0.0062)** |

*Table 5.86: Results of healthy vs tumor classifying patient 8 and training from the rest of patients equally using process #1.*

| **Patient 8<br>Process #4** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | 60.43% (0.0183) | **62.35% (0.0204)** |
| **OA** | 75.16% (0.0115) | **76.36% (0.0128)** |
| **Kappa** | 0.2651 (0.0437) | **0.3098 (0.0475)** |
| **Sensitivity** | **100% (0)** | **100% (0)** |
| **Specificity** | 73.43% (0.0089) | **74.39% (0.0103)** |

*Table 5.87: Results of healthy vs tumor classifying patient 8 and training from the rest of patients equally using process #4.*

- Patient 10 is classified and the training is obtained from the rest of de patients equally.

| **Patient 10<br>Process #1** | **VNIR** | |
|---|---|---|
| **Training type** | **Random training 1** | **Random training 2** |
| **AA** | **97.24% (0.0290)** | 96.53% (0.0614) |
| **OA** | **96.42% (0.0470)** | 94.98% (0.0967) |
| **Kappa** | **0.9097 (0.1127)** | 0.8900 (0.1832) |
| **Sensitivity** | **89.66% (0.1425)** | 87.88% (0.1812) |
| **Specificity** | 99.63% (0.0067) | **99.82% (0.0032)** |

*Table 5.88: Results of healthy vs tumor classifying patient 10 and training from the rest of patients equally using process #1.*

As we can see, the overall results remain similar compared to the previous experiment. Only in some specific cases, such as in the patient 4, worse results have be seen. This may be because in this experiment the training sets are formed equally by 10% of the patient samples that are not classified, thus losing total randomness of training sets. That is to say, it may be that a given patient samples are very different from other patient samples which are training, and in this case we force that the training sets have a fixed number of those training samples that worsen the classification, when in the previous case the training sets could have fewer of these samples. Moreover, we can see how the patient 5 still gives very accurate results. Most troubling is that the patient 7 is still very difficult to classify given an extremely poor results. In addition, we see how the camera 2 is still causing problems in the classifications.

Then, we conducted an experiment in which all samples from all patients are used to training except the samples from the patient which is classified. Thus we have a large training set with full range of samples available. The results are shown below:

- Patient 4 is classified and the training is formed by all samples from all patients.

| **Patient 4**<br>**Process #1** | **VNIR** |
|---|---|
| **AA** | 72.97% |
| **OA** | 66.87% |
| **Kappa** | 0.3964 |
| **Sensitivity** | 53.80% |
| **Specificity** | 99.47% |

*Table 5.89: Results of healthy vs tumor classifying patient 4 and training with all samples from all patients using process #1.*

| **Patient 4**<br>**Process #4** | **VNIR** |
|---|---|
| **AA** | 99.75% |
| **OA** | 99.70% |
| **Kappa** | 0.9937 |
| **Sensitivity** | 99.22% |
| **Specificity** | 100% |

*Table 5.90: Results of healthy vs tumor classifying patient 4 and training with all samples from all patients using process #4.*

- Patient 5 is classified and the training is formed by all samples from all patients.

| Patient 5<br>Process #1 | VNIR | NIR |
|---|---|---|
| **AA** | **99.21%** | 95.00% |
| **OA** | **99.02%** | 94.74% |
| **Kappa** | **0.9793** | 0.8950 |
| **Sensitivity** | **100%** | **100%** |
| **Specificity** | **97.50%** | 90.00% |

*Table 5.91: Results of healthy vs tumor classifying patient 5 and training with all samples from all patients using process #1.*

| Patient 5<br>Process #4 | VNIR |
|---|---|
| **AA** | 99.60% |
| **OA** | 99.51% |
| **Kappa** | 0.9896 |
| **Sensitivity** | 100% |
| **Specificity** | 98.73% |

*Table 5.92: Results of healthy vs tumor classifying patient 5 and training with all samples from all patients using process #4.*

- Patient 7 is classified and the training is formed by all samples from all patients.

| Patient 7<br>Process #1 | VNIR | NIR |
|---|---|---|
| **AA** | 39.45% | **64.29%** |
| **OA** | 28.50% | **59.77%** |
| **Kappa** | -0.1342 | **0.2589** |
| **Sensitivity** | 22.39% | **100%** |
| **Specificity** | 51.88% | **52.05%** |

*Table 5.93: Results of healthy vs tumor classifying patient 7 and training with all samples from all patients using process #1.*

| Patient 7<br>Process #4 | VNIR |
|---|---|
| AA | 36.39% |
| OA | 26.17% |
| Kappa | -0.1733 |
| Sensitivity | 20.82% |
| Specificity | 46.30% |

*Table 5.94: Results of healthy vs tumor classifying patient 7 and training with all samples from all patients using process #4.*

- Patient 8 is classified and the training is formed by all samples from all patients.

| Patient 8<br>Process #1 | VNIR |
|---|---|
| AA | 65.83% |
| OA | 78.55% |
| Kappa | 0.3888 |
| Sensitivity | 100% |
| Specificity | 76.18% |

*Table 5.95: Results of healthy vs tumor classifying patient 8 and training with all samples from all patients using process #1.*

| Patient 8<br>Process #4 | VNIR |
|---|---|
| AA | 64.20% |
| OA | 77.53% |
| Kappa | 0.3524 |
| Sensitivity | 100% |
| Specificity | 75.33% |

*Table 5.96: Results of healthy vs tumor classifying patient 8 and training with all samples from all patients using process #4.*

- Patient 10 is classified and the training is formed by all samples from all patients.

| Patient 10 Process #1 | VNIR |
|---|---|
| **AA** | 85.65% |
| **OA** | 77.78% |
| **Kappa** | 0.5287 |
| **Sensitivity** | 50.40% |
| **Specificity** | 100% |

*Table 5.97: Results of healthy vs tumor classifying patient 10 and training with all samples from all patients using process #1.*

| Patient 10 Process #4 | VNIR |
|---|---|
| **AA** | 100% |
| **OA** | 100% |
| **Kappa** | 1 |
| **Sensitivity** | 100% |
| **Specificity** | 100% |

*Table 5.98: Results of healthy vs tumor classifying patient 10 and training with all samples from all patients using process #4.*

As there were many more samples to train than to classify, it could be expected to this information overload complicates the decisions of classifier getting worse the results. However, overall the results have improved, getting even an acceptable result for the patient classification 7 using NIR camera.

Finally, in order to try to improve the results obtained by classifying the patient 7, a final experiment is performed, in which all the samples of all patients are normalized. This is done to check if the poor results obtained of classifying patient 7 are due to its samples are very different from those of other patient samples. For this experiment, the trainings are formed by 10% of samples of each class and these samples are obtained in the following ways:

1. Training 1: 10% randomly from the rest of patients that are not going to classify.
2. Training 2: 10% of samples from each patient that is not going to classify.

In addition, in this experiment the NIR camera has been discarded due to the bad results obtained by it. The results are shown below:

- Patient 4 classification:

| **Patient 4 Process #1** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | **69.94% (0.1964)** | 60.29% (0.1089) |
| **OA** | **63.10% (0.2416)** | 51.20% (0.1338) |
| **Kappa** | **0.3715 (0.3855)** | 0.1753 (0.1939) |
| **Sensitivity** | **57.04% (0.2215)** | 45.19% (0.0803) |
| **Specificity** | 99.83% (0.0045) | **100% (0)** |

*Table 5.99: Results of healthy vs tumor classifying patient 4 with normalized data using process #1.*

- Patient 5 classification:

| **Patient 5 Process #1** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | 94.51% (0.1422) | **97.02% (0.0853)** |
| **OA** | 95.78% (0.1087) | **97.70% (0.0651)** |
| **Kappa** | 0.8949 (0.2796) | **0.9461 (0.1541)** |
| **Sensitivity** | 95.34% (0.1117) | **97.24% (0.0790)** |
| **Specificity** | **99.87% (0.0040)** | 99.81% (0.0060) |

*Table 5.100: Results of healthy vs tumor classifying patient 5 with normalized data using process #1.*

| **Patient 5<br>Process #4** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | 74.68% (0.2095) | **86.75% (0.2073)** |
| **OA** | 80.64% (0.1602) | **89.87% (0.1585)** |
| **Kappa** | 0.5185 (0.4201) | **0.7523 (0.3928)** |
| **Sensitivity** | 78.47% (0.1590) | **88.45% (0.1730)** |
| **Specificity** | **100% (0)** | **100% (0)** |

*Table 5.101: Results of healthy vs tumor classifying patient 5 with normalized data using process #4.*

- Patient 7 classification:

| **Patient 7<br>Process #1** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | 42.34% (0.0932) | **45.43% (0.1172)** |
| **OA** | **38.10% (0.0696)** | 36.57% (0.0379) |
| **Kappa** | -0.1230 (0.1308) | **-0.0787 (0.1670)** |
| **Sensitivity** | 21.47% (0.0606) | **23.65% (0.0763)** |
| **Specificity** | 68.04% (0.1764) | **74.65% (0.2107)** |

*Table 5.102: Results of healthy vs tumor classifying patient 7 with normalized data using process #1.*

| **Patient 7<br>Process #4** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | **58.33% (0.0034)** | 56.93% (0.0186) |
| **OA** | **39.77% (0.0050)** | 37.73% (0.0268) |
| **Kappa** | **0.0998 (0.0044)** | 0.0822 (0.0232) |
| **Sensitivity** | **31.52% (0.0018)** | 30.83% (0.0090) |
| **Specificity** | **100% (0)** | **100% (0)** |

*Table 5.103: Results of healthy vs tumor classifying patient 7 with normalized data using process #4.*

- Patient 8 classification:

| **Patient 8 Process #1** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | 56.00% (0.0613) | **58.98% (0.0481)** |
| **OA** | 72.38% (0.0385) | **74.25% (0.0302)** |
| **Kappa** | 0.1519 (0.1516) | **0.2275 (0.1199)** |
| **Sensitivity** | **100% (0)** | **100% (0)** |
| **Specificity** | 71.40% (0.0291) | **72.78% (0.0227)** |

*Table 5.104: Results of healthy vs tumor classifying patient 8 with normalized data using process #1.*

- Patient 10 classification:

| **Patient 10 Process #1** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | 88.54% (0.1878) | **96.13% (0.0596)** |
| **OA** | 82.26% (0.2908) | **94.58% (0.0930)** |
| **Kappa** | 0.7262 (0.3996) | **0.8801 (0.1955)** |
| **Sensitivity** | 75.48% (0.3221) | **87.61% (0.2079)** |
| **Specificity** | **100% (0)** | 99.70% (0.0090) |

*Table 5.105: Results of healthy vs tumor classifying patient 10 with normalized data using process #1.*

| **Patient 10 Process #4** | **Training 1** | **Training 2** |
|---|---|---|
| **AA** | **84.89% (0.2296)** | 78.99% (0.2391) |
| **OA** | **76.60% (0.3556)** | 67.47% (0.3702) |
| **Kappa** | **0.6684 (0.4532)** | 0.5362 (0.4555) |
| **Sensitivity** | **71.33% (0.3407)** | 59.92% (0.3330) |
| **Specificity** | **100% (0)** | **100% (0)** |

*Table 5.106: Results of healthy vs tumor classifying patient 10 with normalized data using process #4.*

As shown in the tables 5.99-5.106, overall results have worsened without obtaining a significant improvement in patient 7 classification, so it seems that the poor results achieved with the patient 7 are not because its samples are very different from those of other patients. As *in vivo* samples are labeled manually, it might have errors in the labeling of samples of patient 7.

Previously, it was said that in many of the results obtained by NIR camera all samples were classified as healthy or all as tumor, being very poor results. To conclude this section it is necessary to explain that this was not always so, in fact this situation was quite exceptional. All results shown are the average of the results obtained by performing 10 classifications using 10 different training sets randomly obtained by the procedures outlined in each section. What was actually happening is that almost all these 10 classifications gave the result that all samples were from the same class but not all classifications gave that result. In fact, the most common situation was to find really good classifications within those 10. For example, it was easy to find a couple of classifications with amazingly good results, even 100% accuracy, and the other classifications with one of the worst possible results in which all samples were of the same class. This situation typically replicated in the results obtained by NIR camera and in some cases with VNIR camera.

This fact is very important: if using 10 classifications with 10 random trainings, at least a couple of them have very good results, this means that in general for each patient there are certain sets of training that offer very high accurate results for that patient. This leads us back to the idea of guided trainings, the idea to be able to select a specific and personalized training for each patient, so that the training set chosen maximizes the results obtained for a particular patient.

## 5.2. Global discussion of results

After analyzing the results obtained for the *ex vivo* and *in vivo* samples, we proceed to discuss in general terms the results obtained with both types of samples with the idea of offering an overview about the operation of processing chains analyzed. From the results obtained, it may be performed the following general observations, which result in a set of specific recommendations when each of the processing chains are used.

Regarding the **process #1**, the results obtained for *ex vivo* samples indicate that filtering the data for denoising the samples prior to the application of the SVM classifier is more effective than no filtering the data. In particular, the smooth filter is more effective than HySime filter. Moreover, the results for this processing chain using different kernels featuring SVM indicate that using linear kernel noticeably improves the classification results obtained by this chain, increasing the success rate significantly. Therefore our specific recommendations when using this processing chain, for this type of application, is to apply the smooth filter to the sample set before classifying using SVM with linear kernel.

Regarding the **process #2**, the results obtained for *ex vivo* samples indicate that dimension reduction stage used as pre-processing prior to the application of SVM classifier slightly worsens results obtained with respect to processing chain #1. However, with the implementation of this stage a little precision is lost but the computation time is greatly reduced, which can be a huge advantage. It is worth mentioning that the dimensionality reduction techniques that better work are those based on principal component analysis, being the techniques based on nearest neighbor the ones which offer the worst results. Specifically, the MNF technique provides the best results, but could not be used in this project due to the absence of its code under University license, so that the ICA technique was used in place, also providing very good results. In this regard, if the computing time is an issue, it is recommended to apply a dimensional reduction stage using MNF if possible and if not ICA. Otherwise, if the goal is to get the best possible results, it is recommended not to include a dimensionality reduction stage and directly applying the process #1.

Regarding the **process #3**, the results obtained for *ex vivo* samples indicate that applying unmixing techniques instead of SVM classifier worsens results obtained with respect to processing chains #1 and 2#. Overall, the results are highly variable and dependent on each class, some classes are classified acceptably and other classes on the contrary give very poor results. This module labels each sample with the class that has the greatest abundance in that sample. Perhaps using another decision criterion, such as setting a threshold to abundances, better results could be obtained. The implementation of this module for such applications is not recommended, but if necessary, we recommend to extract the endmember following a general procedure and to use unconstrained linear spectral unmixing due to better accuracy and faster computation.

Regarding the **process #4**, the results obtained for *ex vivo* samples indicate that unmixing stage used as pre-processing prior to the application of SVM classifier worsens results obtained with respect to processing chains #1 and 2#. However, the results obtained with this chain are very good, despite they do not improve the results of the first two chains. Moreover, due to a significant reduction in computation time is not obtained and the results are slightly worse than those obtained by processing chain #2, we recommend using process #2 before this. However, if mixed techniques Unmixing-SVM are going to be used, it is recommended to apply a dimensionality reduction stage prior the unmixing stage because this improves the results. Furthermore, as in the previous processing chain #3, it is recommended to extract the endmember using a general process and to use unconstrained linear spectral unmixing.

Regarding the **process #5**, the results obtained for *ex vivo* samples indicate that using guided training to train SVM classifier do not improve results in none of the cases with the strategies used for obtaining guided training sets. The training set should be as representative as possible of the data set to classify. The training set should be as generic as possible so that it includes all types of samples in each class and large diversity of samples. For this reason, random training sets offered always better results than guided training sets.

Regarding the **process #6**, this processing chain is only a simulation of processing chain #7. The results obtained were very good for all processing chain tested and these served as an incentive to address the processing chain #7 using *in vivo* samples.

Regarding the **process #7**, to be the first experiment simulating a real situation and using *in vivo* samples the results are quite good and promising for future experiments. Note that apparently there are patients who are easier to classify than other patients. This means that there are patients who for all the experiments were successfully and others that it was impossible to classify correctly. It should be noted the importance of mixed Unmixing-SVM techniques in this last processing chain, in which these techniques improve outcomes in many cases. For this reason it is recommended to use mixed Unmixing-SVM techniques for this application.

## 5.3. Summary

In this chapter, we have investigated several strategies to extract relevant features from hyperspectral imaging prior to classification. For classification scenarios, using SVM trained with wide variety of guided and random training sets, our experimental results reveal that for *ex vivo* samples the best results are obtained with process #1, being able to obtain high speed computing in exchange for slightly lower accuracy by using the process

#2. Moreover, in the case of *in vivo* samples, the use of mixed Unmixing-SVM techniques it is highly recommended. In any case, it is recommended to always work using random training sets.

# Chapter 6

# Conclusions and future research lines

## 6.1. Conclusions

In this work we have developed a quantitative and comparative detailed analysis of different processing chains for the analysis and classification of hyperspectral images. Considered processing chains are based on the combination of different pre-processing modules (especially focusing on dimensionality reduction and unmixing techniques) which have been applied prior a highly consolidated classifier in hyperspectral analysis applications (SVM) that is trained using different approaches (random and guided trainings). In this regard, the different strategies considered cover a range of highly representative techniques of the state of the art in hyperspectral data analysis, all combined with advanced classifiers able to work very precisely with high-dimensional data sets and using diverse training sets. This study was performed using two different samples sets (*ex vivo* and *in vivo*) which has made possible a detailed study of different processing chains based on classifiers and pre-processing techniques mentioned above providing an important approximation to the final system developed by HELiCoiD. It should be noted that the study conducted includes topics of great interest, such as the impact of using dimensionality reduction techniques as well as mixed Unmixing-SVM techniques whose concepts have been seldom studied together, although they exhibit complementary properties that can offer several advantages when they are applied to hyperspectral image analysis. In this regard, it should be noted that in the literature there are few comparative studies of this kind to date, therefore the variety of results achieved and interesting conclusions produced by the analysis of these results may represent a valuable contribution to the existing literature regarding classification of hyperspectral data.

## 6.2. Future research lines

As regards the future research lines derived from the present project, we can make the following considerations. Once already it is taken the leap to the use of *in vivo* samples simulating a situation of real surgery, future research lines are developed around to continue improving the final system with *in vivo* samples. As discussed in the results section 5.1.7, due to the fact that among the random training sets with bad results were found isolated cases with 100% accuracy it can delve into the idea of guided training, trying to perfecting the strategies proposed in this work in order to find the best training sets for each individual patient. Conduct a thorough study of the spectral signatures of the classes that we have in the classification problem could help in finding the best samples to form these training sets. Namely, we currently work with a large number of spectral bands, performing an in-depth study of the spectral signatures of the main classes we might find the spectral bands in which these classes differ more. This would allow us to greatly simplify the problem only working directly with these bands, reducing the computation time because of fewer bands and facilitating the classification since classes would be further apart. Thanks to this, we could create a database or library sufficiently robust formed by spectral signatures generic enough to use it to train the classifier to classify future patients with good results. It would also be interesting to train the classifier not only from the database or library, but also with the patient samples which is being classified, so that the training set is formed by samples of the database or library and samples of patient to be classified. Additionally it could include a step of post-processing using techniques that combine spectral and spatial information in order to further improve the results of the classification. Of course the whole system obtained to work with brain tumors could be extended to other tumor types and conditions.

# Bibliography

[1] Gamba P., Plaza A., Benediktsson J. A. and Chanussot. J., “European Perspectives in hyperspectral Data Analysis”, *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'07)*, Barcelona, Spain, 2007.

[2] C. I. Chang. Recent advances in hyperspectral signal and image processing. John Wiley & Sons: New York, 2007.

[3] J. A. Richards and X. Jia. Remote sensing digital image analysis: an introduction. Springer, 2006.

[4] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354-379, 2012.

[5] Marta Rojas Muriel. *Caracterización de imágenes hiperespectrales utilizando Support Vector Machines y técnicas de extracción de características*. December 2009.

[6] Clark, R.N. “Spectroscopy of Rocks and Minerals, and Principles of Spectroscopy”. Chapter 1 in *Manual of Remote Sensing*, John Wiley and Sons, New York, 1999a.

[7] Landgrebe, D., “Hyperspectral Image Data Analysis”, *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 17-28, 2002.

[8] Boardman, J.W., Kruse, F.A., Green, R.O., "Mapping target signatures via partial unmixing of AVIRIS data", *Summaries of the VI JPL Airborne Earth Science Workshop*, 1995.

[9] Hsieh, P.-F., Landgrebe, D., "Classification of High Dimensional Data." Tesis Doctoral, School of Electrical and Computer Engineering, Purdue University, 1998.

[10] Green, R.O. y col., "Imaging Spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)", *Remote Sensing of Environment*, vol. 65, pp. 227-248, 1998.

[11] Chein-I Chang (31 July 2003). *Hyperspectral Imaging: Techniques for Spectral Detection and Classification.* Springer Science & Business Media. ISBN 978-0-306-47483-5.

[12] Hans Grahn; Paul Geladi (27 September 2007). *Techniques and Applications of Hyperspectral Image Analysis*. John Wiley & Sons. ISBN 978-0-470-01087-7.

[13] Guolan Lu and Baowei Fei "Medical Hyperspectral Imaging: a review" (ONLINE FULL TEXT ARTICLE AVAILABLE). *Journal of Biomedical Optics* 19 (1): 10901. January 2014. Bibcode: 2014JBO....19a0901L. doi:10.1117/1.JBO.19.1.010901. PMID 24441941.

[14] S. Chaudhuri, K. Kotwal. *Hyperspectral Image Fusion*. Springer, 2013.

[15] C.-I. Chang. *Hyperspectral Data Processing: Algorithm Design and Analysis*. Wiley, 2013.

[16] D. Glotsos, P. Spyridonos, P. Petalas, D. Cavouras, V.Zolota, P. Dadioti, I. Lekka, G. Nikiforidis, "A hierarchical decision tree classification scheme for brain tumour astrocytoma grading using support vector machines" in Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, Vol. 2, pp 1034-1038, 2003.

[17] *Brain Cancer Health Center, WebMD*. [En línea]. Disponible en: http://www.webmd.com/cancer/brain-cancer/malignant-gliomas [visitado el 8 de noviembre de 2014].

[18] *Multiform glioblastoma. National Brain Tumour Society.* [En línea]. Disponible en: http://www.braintumor.org/patients-family-friends/about-brain-tumors/tumor-types/glioblastoma-multiforme.html [visitado el 5 de octubre de 2014].

[19] Qingli Li, Xiaofu He, Yiting Wang, Hongying Liu, Dongrong Xu and Fangmin Guo. “Review of spectral imaging technology in biomedical engineering: achievements and challenges”. *Journal of Biomedical Optics* 18(10), 100901. October 2013.

[20] G. Zonios et al., “Diffuse reflectance spectroscopy of human adenomatous colon polyps *in vivo*,” *Appl. Opt.* 38(31), 6628–6637 (1999).

[21] L. V.Wang and H.-I.Wu, “Introduction,” in *Biomedical Optics*, pp. 1–15, John Wiley & Sons Inc., Hoboken, New Jersey (2009).

[22] B. Costas, P. Christos, and E. George, “Multi/hyper-spectral imaging,” in Handbook of Biomedical Optics, pp. 131–164, CRC Press (2011).

[23] V. V. Tuchin and V. Tuchin, *Tissue Optics: Light Scattering Methods and Instruments forMedical Diagnosis*, SPIE Press, Bellingham (2007).

[24] D. G. Ferris et al., “Multimodal hyperspectral imaging for the noninvasive diagnosis of cervical neoplasia,” *J. Low. Genit. Tract Dis.* 5(2), 65–72 (2001).

[25] M. C. Pierce et al., “Accuracy of *in vivo* multimodal optical imaging for detection of oral neoplasia,” *Cancer Prev. Res.* 5(6), 801–809 (2012).

[26] G. Bellisola and C. Sorio, “Infrared spectroscopy and microscopy in cancer research and diagnosis,” *Am. J. Cancer Res.* 2(1), 1–21 (2012).

[27] A. Nouvong et al., “Evaluation of diabetic foot ulcer healing with hyperspectral imaging of oxyhemoglobin and deoxyhemoglobin,” *Diabetes Care* 32(11), 2056–2061 (2009).

[28] R. Salzer et al., “Infrared and Raman imaging of biological and biomimetic samples,” *Fresenius J. Anal. Chem.* 366(6–7), 712–726 (2000).

[29] A. M. Siddiqi et al., “Use of hyperspectral imaging to distinguish normal, precancerous, and cancerous cells,” *Cancer Cytopathol*. 114(1), 13–21 (2008).

[30] P. Y. C. N. Mazzer et al., “Morphologic and morphometric evaluation of experimental acute crush injuries of the sciatic nerve of rats,” *J. Neurosci. Methods* 173(2), 249–258 (2008).

[31] Shaw, G., Manolakis, D. “Signal processing for hyperspectral image exploitation”. *IEEE Signal Processing Magazine*, vol. 19, pp. 12-16, 2002.

[32] Stein, D.W., Beaven, S.G., Hoff, L.E., Winter, E.M., Schaum, A.P., Stocker, A.D., “Anomaly Detection from Hyperspectral Imagery”. *IEEE Signal Processing Magazine*, vol. 19, pp. 58-69, 2002.

[33] Plaza, P. Gamba, K. Bakos, B. Waske, J. B. Diaz, “HYPERINET_D4.1 – Processing Chain Definition Report”, January, 2008 (http://www.hyperinet.eu).

[34] Plaza, P. Gamba, K. Bakos, B. Waske, “HYPERINET_D4.2 – Processing Chain Implementation Report”, January, 2009 (http://www.hyperinet.eu).

[35] Chang, C.-I, Ren, H., “An Experiment-Based Quantitative and Comparative Analysis of Target Detection and Image Classification Algorithms for Hyperspectral Imagery”. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 2, pp. 1044- 1063, 2000.

[36] Chen, J. M., “Spatial Scaling of a Remotely Sensed Surface Parameter by Contexture”. Remote Sensing of Environment, vol. 69, pp. 30-42, 1999.

[37] Stehman, S. V., “Selecting and Interpreting Measures of Thematic Classification Accuracy”. *Remote Sensing of Environment*, vol. 62, pp. 77-89, 1997.

[38] Chiang, S.-S., Chang, C.-I., Ginsberg, I.W., “Unsupervised target detection in hyperspectral images using projection pursuit”. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, pp. 1380-1391, 2001.

[39] Manolakis, D., Shaw, G., “Detection algorithms for hyperspectral imaging applications”. *IEEE Signal Processing Magazine*, vol. 19, pp. 29-43, 2002.

[40] Theiler, J., Gisler, G., “A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation”, in: *Proc. SPIE*, vol. 3159, pp. 108-118, 1997.

[41] Richards, J. A., *Remote Sensing Digital Image Analysis: An Introduction.* Springer-Verlag, Berlin, 1993.

[42] Sweet, J., Granaham, J., Sharp, M., "An Objective Standard for Hyperspectral Image Quality", in: *Proc. IX -ASA/JPL Airborne Earth Science Workshop*, Pasadena, CA, 2000.

[43] Plaza, A. Mueller, T. Skauli, Z. Malenovsky, J. Bioucas, S. Hofer, J. Chanussot, Carrere, I. Baarstad, J. Nieke, T. Hyvarinen, P. Gamba, J. A. Benediktsson, M. E. chaepman and B. Zagajewski. "HYPER-I-NET: European Research Network on Hyperspectral Imaging", *IEEE International Geoscience and Remote Sensing Symposium (IGARSS'07)*, Barcelona, Spain, 2007.

[44] Rellier, G., Descombes, X., Zerubia, J., "Local registration and deformation of a road cartographic database on a SPOT satellite image". *Pattern Recognition*, vol. 35, pp. 2213-2221, 2002.

[45] Madhok, V., Landgrebe, D., *Spectral-Spatial Analysis of Remote Sensing Data: An Image Model and A Procedural Design*. PhD thesis, School of Electrical Engineering and Computer Science, Purdue University, 1998.

[46] Tadjudin, S., Landgrebe, D., *Classification of High Dimensional Data with Limited Training Samples*. PhD thesis, School of Electrical Engineering and Computer Science, Purdue University, 1998.

[47] Naesset, E., "Predicting forest stand characteristics with airborne scanning laser using a practical two-stage procedure and field data". *Remote Sensing of Environment*, vol. 80, pp. 88-99, 2002.

[48] Vaughan, R.G., Calvin, W.M., Taranik, J., "Analysis of Sub-Pixel Mixing in High Altitude AVIRIS Data Over Virginia City, Nevada, Using Systematic Field-Based Observations", in: *Proc. XI -ASA/JPL Airborne Earth Science Workshop*, Pasadena, CA, 2001.

[49] Congalton, R.G., "Considerations and Techniques for Assessing the Accuracy of Remotely Sensed Data", in: *Proc. International Geoscience and Remote Sensing Symposium IGARSS*, vol. 3, pp. 1847-1850, 1989.

[50] Stehman, S.V., “Practical Implications of Design-Based Sampling Inference for Thematic Map Accuracy Assessment”. *Remote Sensing of Environment*, vol. 72, pp. 35-45, 2000.

[51] Steele, B.M., Winne, J.C., Redmond, R.L., “Estimation and Mapping of Misclassification Probabilities for Thematic Land Cover Maps”, *Remote Sensing of Environment*, vol. 66, pp. 192-202, 1998.

[52] Jäger, G., Benz, U., “Measures of classification accuracy based on fuzzy similarity”. *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 2, pp. 1462-1467, 2000.

[53] John A. Richards. *Remote Sensing Digital Image Analysis An Introduction, Fifth Edition*. 2013.

[54] Rafael C. Gonzalez; Richard E. Woods (2008). *Digital Image Processing*. Prentice Hall. pp. 1–3. ISBN 978-0-13-168728-8.

[55] W. Philpot. *Digital Image Processing*. Cornell University.

[56] Gabriel Martín Hernández. *Desarrollo de algoritmos para identificación de componentes puros en imágenes hiperespectrales.* June 2010.

[57] Damien Letexier and Salah Bourennane. *Estimation of N-mode ranks of hyperspectral images for tensor denoising.* August 2009.

[58] Sourabh Pargal. “Hyperspectral Subspace Identification and End member Extraction by Integration of Spatial-Spectral Information”, Indian Institute Of Remote Sensing, April 2011.

[59] J. Bioucas-Dias and J. Nascimento, “Hyperspectral Subspace Identification,” *Geoscience and Remote Sensing, IEEE Transactions on,* vol. 46, no. 8, pp. 2435-2445, Aug. 2008.

[60] M. Farzam and S. Beheshti, “The Noiseless code-length concept in subspace estimation for low SNR hyperspectral signals,” in Communications, Computers and Signal Processing, 2009. PacRim 2009. IEEE Pacific Rim Conference on, pp. 425-430, 2009.

[61] J. Jensen, *Introductory digital image processing: a remote sensing perspective, 2nd ed. Upper Saddle River N.J.: Prentice Hall, 1996.*

[62] Matlab help, Smooth filter. Matlab version 7.10.0.499 (R2010a). The Mathworks, Inc. February 5, 2010.

[63] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Ph.D.

[64] Eduardo Roldán Arroita. *Estudio y análisis de filtros aplicados a señales vibratorias de ejes ferroviarios para la detección de fallos*. September 2012.

[65] Prof. Tom O'Haver. *Introduction to Signal Processing: Smoothing*. [On-line]. Available on: http://terpconnect.umd.edu/~toh/spectrum/Smoothing.html [Visited March 8, 2015].

[66] Samet, H. (2006) *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann.

[67] C. Ding, , X. He , H. Zha , H.D. Simon, Adaptive Dimension Reduction for Clustering High Dimensional Data, Proceedings of International Conference on Data Mining, 2002.

[68] Hughes G. F., "On The Mean Accuracy Of Statistical Pattern Recognizers", *IEEE Trans. Infor. Theory*, Vol. IT-14, 1968.

[69] Fukunaga K., *"Introduction to Statistical Pattern Recognition"* Published by Academic Press, 1990.

[70] Kaarna A., Zemcik P., Kalviainen H., Parkkinen J., "Compression of multispectral remote sensing images using clustering and spectral reduction", *IEEE Transactions on Geoscience and Remote Sensing,* vol. 38, 2000.

[71] Chang, C.-I, Du, Q., Sun, T., Althouse, L.G., "A Joint Band Prioritization and Band-Decorrelation Approach to Band Selection for Hyperspectral Image Classification", *IEEE Transactions on Geoscience Remote Sensing,* vol. 37, no.6, pp.2631-2641, 1999b.

[72] Roweis, S. T.; Saul, L. K. (2000). *“Nonlinear Dimensionality Reduction by Locally Linear Embedding”.* Science 290 (5500): 2323–2326.

[73] Pudil, P.; Novovičová, J. (1998). “Novel Methods for Feature Subset Selection with Respect to Problem Knowledge”. In Liu, Huan; Motoda, Hiroshi. *Feature Extraction, Construction and Selection.* p. 101.

[74] Qian, S.-E y col., “Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression”. *IEEE Transactions on Geoscience and Remote Sensing,* vol. 38, pp. 1183 –1190, 2000.

[75] Plaza A., Chang, C.-I, High Performance Computing in Remote Sensing, CRC Press, 2007.

[76] Ifarraguerri, A., Chang, C.-I, "Multispectral and hyperspectral image analysis with projection pursuit," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 38, pp. 2529-2538, 2000.

[77] Subramanian, S., Gat, N., Ratcliff, A., Eismann, M., “Hyperspectral Data Reduction Using Principal Components Transformation”, in *Proc. X ASA/JPL Airborne Earth Science Workshop,* 2000.

[78] Lee, J., Woodyatt, A., Bergman, M., “Enhancement of high spectral resolution remote sensing data by noise adjusted principal components transform”. *IEEE Transactions on Geoscience and Remote Sensing,* vol. 28, pp. 295–304, 1990.

[79] Green, A.A., Berman, M., Switzer, P., Craig, M.D., “A transformation for ordering multispectral data in terms of image quality with implications for noise removal”. *IEEE Transactions on Geoscience and Remote Sensing,* vol. 26, pp. 65-74, 1988.

[80] Curran, P.J. Dungan, J.L., “Estimation of Signal-to-Noise: A New Procedure Applied to AVIRIS Data”, *IEEE Transactions on Geoscience and Remote Sensing,* vol. 27, pp. 620-628, 1989.

[81] Gordon, C., “A Generalization of the Maximum Noise Fraction Transform”. *IEEE Transactions on Geoscience and Remote Sensing,* vol. 38, pp. 612–615, 2000.

[82] Stone, James V. (2004). *Independent component analysis: a tutorial introduction.* Cambridge, Mass. [u.a.]: MIT Press.

[83] Aapo Hyvarinen, Juha Karhunen, Erkki, Oja (2001*). Independent component analysis* (1st ed.). New York: J. Wiley.

[84] Gaël Varoquaux. *PCA and ICA: Identifying combinations of variables.* Fri 05 February 2010.

[85] Qian Du, Ivica Kopriva, Harold Szu. *Independent-component analysis for hyperspectral remote sensing imagery classification.* Optical Engineering 45(1), 017008 (January 2006).

[86] Xiaofei He and Partha Niyogi. "Locality Preserving Projections". *Advances in Neural Information Processing Systems 16*, Vancouver, British Columbia, Canada, 2003.

[87] M. Belkin and P. Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering", *Advances in Neural Information Processing Systems 14*, Vancouver, British Columbia, Canada, 2002.

[88] C. L. Blake and C. J. Merz. *"UCI repository of machine learning databases",* http://www.ics.uci.edu/mlearn/MLRepository.html. Irvine, CA, University of California, Department of Information and Computer Science, 1998.

[89] Xiaofei He, Deng Cai, Shuicheng Yan and Hong-Jiang Zhang. *outliers borhood Preserving Embedding*. Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05), 1550-5499/05, 2005.

[90] Sam Roweis, and Lawrence K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". *Science*, vol 290, 22 December 2000.

[91] Ke Yan. "Compute the supervised NPE (neighborhood preserving embedding) subspace for dimension reduction". 26 March 2011.

[92] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller. *Nonlinear Component Analysis as a Kernel Eigenvalue Problem.* Neural Computation, July 1, 1998, Vol. 10, No. 5, Pages 1299-1319. Posted Online March 13, 2006.

[93] Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. *Nonlinear Component Analysis as a Kernel Eigenvalue Problem.* Max-Planck-Institut, Technical Report No. 44, December 1996.

[94] Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. *Kernel Principal Component Analysis*. Max-Planck-Institut, Berlin, Germany.

[95] Max Welling. *Kernel Principal Components Analysis*. Department of Computer Science, University of Toronto.

[96] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44-57, 2002.

[97] J. A. Richards. Analysis of remotely sensed data: the formative decades and the future. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):422-432, 2005.

[98] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113:110-122, 2009.

[99] J. B. Adams, M. O. Smith, and P. E. Johnson. Spectral mixture modeling: a new analysis of rock and soil types at the viking lander 1 site. *Journal of Geophysical Research*, 91(B8):8098-8112, 1986.

[100] Inmaculada García Dópido. *New Techniques for Hyperspectral Image Classification*. January 2014.

[101] N. Keshava and J. Mustard, “Spectral unmixing,” *Signal Processing Magazine, IEEE*, vol. 19, no. 1, pp. 44-57, 2002.

[102] Samuel Rosario Torres. “Iterative Algorithms for Abundance Estimation on Unmixing of Hyperspectral Imagery”. University of Puerto Rico, Mayagüez Campus, 2004.

[103] Gabriel Martín Hernández. *Comparativa de nuevos algoritmos de extracción de endmembers en imágenes hiperespectrales utilizando información espacial*. February, 2009.

[104] C.-I. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, no. 3, pp. 608–619, 2004.

[105] A. Plaza, P. Martinez, R. Perez, and J. Plaza. A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 42(3):650-663, 2004.

[106] J. C. Harsanyi and C. I. Chang. Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):779-785, 1994.

[107] J. M. P. Nascimento and J. M. Bioucas-Dias. Vertex component analysis: a fast algorithm to unmix hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(4):898-910, 2005.

[108] M. E. Winter. N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. *Proceedings of SPIE Image Spectrometry V*, 3753:266-277, 1999.

[109] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 9, pp. 2025–2041, 2002.

[110] J. H. Bowles, P. J. Palmadesso, J. A. Antoniades, M. M. Baumback, and L. J. Rickard, "Use of filter vectors in hyperspectral data analysis," *Proc. SPIE Infrared Spaceborne Remote Sensing III*, vol. 2553, pp. 148–157, 1995.

[111] M. Zortea and A. Plaza, "Spatial preprocessing for endmember extraction," I*EEE Trans. Geosci. Remote Sens.*, under review, 2008.

[112] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 4, pp. 779–785.

[113] Ren, H. y Chang, C.-I, “Automatic spectral target recognition in hyperspectral imagery,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1232-1249, 2003.

[114] Emmett Ientilucci. “Hyperspectral Image Classification Using Orthogonal Subspace Projections: Image Simulation and Noise Analysis”. Center for Imaging Science. April 23, 2001.

[115] M.E. Winter, “N-FINDR: an algorithm for fast autonomous spectral endmember determination in hyperspectral data,” *Imaging Spectrometry V, Proc. SPIE 3753*, pp. 266-277, 1999.

[116] M.E. Winter, “A proof of the N-FINDR algorithm for the automated detection of endmembers in a hyperspectral image,” Algorithms and Technologies for Multispectral, *Hyperspectral and Ultraspectral Imagery X, Proc. SPIE* 5425, pp. 31-31, 2004.

[117] Healey, G., & Slater, D.. “Models and methods for automated material identification in hyperspectral imagery acquired under unknown illumination and atmospheric conditions”, *IEEE Transactions on Geosciencce and Remote Sensing*, 37, 2706–2717, 1999.

[118] D. C. Heinz and C. I. Chang. Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(3):529-545, 2001.

[119] Vélez-Reyes M.; et. al. “Iterative Algorithms for Unmixing of Hyperspectral Imagery”. *Proceeding of SPIE*, April 2003.

[120] Keshava N.; Mustard J. F. “Spectral Unmixing”. *IEEE Signal Processing Magazine*, 19:44 - 57, January issue 1 2002.

[121] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). *An Introduction to Statistical Learning.* Springer. p. 204.

[122] Bermingham, Mairead L.; Pong-Wong, Ricardo; Spiliopoulou, Athina; Hayward, Caroline; Rudan, Igor; Campbell, Harry; Wright, Alan F.; Wilson, James F.; Agakov,

Felix; Navarro, Pau; Haley, Chris S. (2015). "Application of high-dimensional feature selection: evaluation for genomic prediction in man". *Sci. Rep.* 5.

[123] Guyon, Isabelle; Elisseeff, André (2003). "An Introduction to Variable and Feature Selection". *JMLR* 3.

[124] LATORRE-CARMONA, Pedro, et al. Effect of denoising in band selection for regression tasks in hyperspectral datasets. *IEEE Journal of selected topics in applied Earth observations and remote sensing*, 2013, vol. 6, no 2, p. 473 – 481.

[125] Mohsen Ghamary Asl, Mohammad Reza Mobasheri, and Barat Mojaradi, "Unsupervised Feature Selection Using Geometrical Measures in Prototype Space for Hyperspectral Imagery", IEEE transactions on geoscience and remote sensing, vol. 52, no. 7, July 2014.

[126] Vikneswaran. K. S, Dr. N. Kasthuri, V. R. Saraswathy. Enhancement of feature selection technique for network dataset. *International Journal of advanced research tends in engineering and technology (IJARTET),* vol. II, Special Issue VIII, February 2015.

[127] R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing.* New York: Academic, 1997.

[128] C.-I Chang, T.-L. E. Sun, and M. L. G. Althouse, "An unsupervised interference rejection approach to target detection and classification for hyperspectral imagery," *Opt. Eng.*, vol. 37, no. 3, pp. 735–743, Mar. 1998.

[129] J. Harsanyi, W. Farrand, and C.-I. Chang, "Determining the number and identity of spectral endmembers: An integrated approach using neyman-pearson eigenthresholding and iterative constrained rms error minimization," in *Proc. 9th Thematic Conf. Geologic Remote Sensing,* 1993.

[130] H. V. Poor, *An Introduction to Detection and Estimation Theory*, 2$^{nd}$ ed. New York: Springer-Verlag, 1994.

[131] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, 2nd ed. New York: Wiley, 1984.

[132] Benediktsson J.A. and Arnason K., “Classification and feature extraction of aviris data”, *IEEE Trans. on Geoscience and Remote Sensing*, vol.33, pp. 1194:1205, 1995.

[133] Vapnik V. *Statistical learning theory*. Wiley, New York, 1998.

[134] Gualtieri J. A. and Chettri S., “Support vector machines for classification of hyperspectral data”, *Proc. IEEE International Geoscience and Remote Sensing Symposium*, volume 2. IGARSS ’00. Proceedings, July 2000.

[135] Cortes C. and Vapnik V. *“Support vector networks”.* Machine Learning, 20:1- 25, 1995.

[136] Schölkopf B., Smola A. *“Advances in kernel methods: Support vector learning”*. 1999.

[137] Boser B. E., Guyon I. M., and Vapnik V. N.. *“A training algorithm for optimal margin classifiers”*, In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992. ACM Press.

[138] Muller K. R., Mika S., Ratsch G., Tsuda K., Schölkopf B., *“An introduction to kernel-based learning algorithms”*. IEEE Transactions on -eural -etworks, vol 12, 2001.

[139] Mercier G. and Lennon M., *“Support Vector Machines for Hyperspectral Image Classification with Spectral-Based Kernels”*, in IGARSS, 2003.

[140] Foody G. M., *“RVM-based multi-class classification of remotely sensed data”* International Journal of Remote Sensing, vol 29, pp 1817-1823, 2008.

# BUDGET

El presupuesto del presente Proyecto de Fin de Carrera se ha valorado en función de la última lista de Honorarios Orientativos publicada por el Colegio Oficial de Ingenieros de Telecomunicación, denominada *Costes Estimados de Trabajos Profesionales* correspondiente al año 2008. Así, el presupuesto se ha estructurado en siete secciones:

1. Recursos Humanos
2. Recursos Hardware
3. Recursos Software
4. Material Fungible
5. Aplicación de Impuestos
6. Presupuesto total

Hay que indicar que el motivo por el que este presupuesto toma como referencia una lista publicada en 2008 se debe a las modificaciones introducidas en el ordenamiento jurídico y a la actuación de los colegios profesionales en la ley 25/2009 del 22 de diciembre, por la cual se liberalizan los honorarios profesionales y ya no es posible seguir publicando este tipo de listas por parte del COIT.

## Recursos humanos

El coste de los recursos humanos está asociado al tiempo empleado por un ingeniero en la realización de este proyecto. El coste se establece aplicando la fórmula propuesta por el COIT para trabajos por tiempo empleado:

$$Honorarios(€) = (74.88H_n + 96.72H_e)C_t$$

Donde $H_n$ representa el número de horas normales dentro de la jornada laboral, mientras que $H_e$ se considera el número de horas especiales, siendo 260,00 euros el honorario mínimo a cobrar independientemente del número de horas trabajadas. El coeficiente $C_t$ es el factor de corrección que se debe aplicar al número de horas, variando en función del número de horas empleadas según la tabla 8.1:

| Horas trabajadas | $C_t$ |
|---|---|
| 0-36 | 1 |
| 36 - 72 | 0,90 |
| 72 - 108 | 0,80 |

| | |
|---|---|
| 108 - 144 | 0,70 |
| 144 - 180 | 0,65 |
| 180 - 360 | 0,60 |
| 360 - 540 | 0,55 |
| 540 - 720 | 0,50 |
| 720 - 1080 | 0,45 |
| + 1080 | 0,40 |

*Tabla 8.1: Factor de corrección en función de horas trabajadas.*

Teniendo en cuenta una jornada laboral de ocho horas diarias a razón de veinte días laborales cada mes durante 10 meses, el número total de horas empleadas es de 1600 horas normales, siendo cero el número de horas especiales trabajadas. Según la tabla 8.1, el factor de corrección que corresponde al número de horas trabajadas es $C_t = 0{,}40$.

En la tabla 8.2 se muestra el resultado de la aplicación de la ecuación anterior, recogiendo los costes asociados a los recursos humanos libres de impuestos, que ascienden a *cuarenta y siete mil novecientos veintitrés euros con dos céntimos* (47923,2€).

| Concepto | Tiempo trabajado (horas) | Factor de corrección | Importe (€) |
|---|---|---|---|
| Ingeniero | 1600 | 0,40 | 47923,2 |
| Coste Total | | | 47923,2 |

*Tabla 8.2: Coste total de recursos humanos.*

## Recursos hardware

El coste de los recursos hardware vendrá determinado por la instrumentación de medida y los equipos informáticos empleados en la realización del presente Proyecto Fin de Carrera.

- Cámara Hiperespectral Headwall's Hyperspec NIR.
- Cámara Hiperespectral Headwall's Hyperspec VNIR.
- Ordenador portátil HP Pavilion dv6 Intel Core i7 (2,60 GHz), con 8 GB de RAM y 1 TB de disco duro.

Para el cálculo de los costes de los recursos materiales, hardware y software, se utilizara un sistema de amortización lineal o constante, en el que se supone que el inmovilizado material se deprecia de forma constante a lo largo de su vida útil. La cuota de amortización anual se calcula usando la siguiente fórmula:

$$Cuota\ anual = \frac{Valor\ de\ adquisición - Valor\ residual}{Años\ de\ vida\ \text{útil}}$$

El valor residual es el valor teórico que tendrá el elemento analizado después de su vida útil. El periodo de amortización de estos recursos se ha considerado de 36 meses, y su tiempo de uso ha sido de 10 meses. Los costes asociados a los recursos hardware libres de impuestos se recogen en la tabla 8.3 y ascienden a *tres mil cuarenta y siete euros con tres céntimos* (3047,3€).

| Concepto | Coste unitario (€) | Valor residual (€) | Coste mensual (€) | Importe (€) |
|---|---|---|---|---|
| Cámara hiperespectral NIR | 9000 | 4250 | 129,73 | 1297,3 |
| Cámara hiperespectral VNIR | 9500,00 | 4500,00 | 138,89 | 1388,9 |
| Ordenador | 1300,00 | 0,00 | 36,11 | 361,1 |
| Coste Total | | | | 3047,3€ |

*Tabla 8.3: Coste total de recursos hardware.*

## Recursos software

Para los recursos software se utilizan los mismos criterios que en el epígrafe anterior, tomando en este caso un periodo de amortización de 24 meses. Las herramientas software usadas han sido las siguientes:

- Sistema Operativo Microsoft Windows 7 Enterprise Edition 64 bits.
- MATLAB 2010.

En la tabla 8.4 se recogen los costes asociados a los recursos software libres de impuestos. El coste total asociado a las herramientas software empleadas asciende a *novecientos cincuenta y ocho euros con treinta y tres céntimos* (958,33€).

| Concepto | Coste unitario (€) | Valor residual (€) | Coste mensual (€) | Importe (€) |
|---|---|---|---|---|
| Windows 7 | 300,00 | 0,00 | 12,50 | 125,00 |
| MATLAB 2010a | 2000,00 | 0,00 | 83,33 | 833,33 |
| Coste Total | | | | 958,33€ |

*Tabla 8.4: Coste total de recursos software.*

## Material fungible

En este apartado se recopilan los costes relacionados con los materiales utilizados en la realización del proyecto, como son: material de papelería, servicio de impresión del IUMA, discos CD-R y los costes de impresión y encuadernación de la memoria.

Los costes asociados al material fungible libres de impuestos son en este Proyecto Fin de Carrera de *trescientos quince euros* (315€).

| Concepto | Importe (€) |
|---|---|
| Servicio de impresión | 235,36 |
| CD-R | 2,50 |
| Memoria | 77,14 |
| Coste total | 315€ |

*Tabla 8.5: Coste total de material fungible.*

## Redacción del proyecto

De acuerdo a los honorarios orientativos del COIT, el importe de la redacción del presente PFC se calcula mediante la siguiente ecuación:

$$R = 0{,}07 P_t C_r + 0{,}03 P_c C_r$$

Donde $P_t$ es el presupuesto de ejecución material de telecomunicaciones, $P_c$ es el presupuesto de obra civil y $C_r$ es el coeficiente de ponderación por tramos en función del coste del presupuesto. Este Proyecto Fin de Carrera no tiene asociada ninguna obra civil, por lo que $P_c$ es nulo. El presupuesto de ejecución material se corresponde con la suma de los cuatro apartados anteriores:

$$P_t = 47923{,}2 + 3047{,}3 + 958{,}33 + 315 = 52278{,}83€$$

Aplicando la ecuación anterior:

$$R = 0{,}07 \cdot 51644{,}76 = 3615{,}13€$$

Por lo tanto, los costes asociados a la redacción del proyecto libres de impuestos ascienden a *tres mil seiscientos quince euros con trece céntimos* (3615,13€).

## Aplicación de impuestos

En los anteriores apartados se ha recogido cada uno de los costes que se han generado para el desarrollo de este Proyecto Fin de Carrera. La realización del mismo estará gravada con el Impuesto General Indirecto Canario (IGIC), en un siete por ciento (7 %).

| Concepto | Importe (€) |
|---|---|
| Recursos humanos | 47923,2 |
| Recursos hardware | 3047,3€ |
| Recursos software | 958,33€ |
| Material fungible | 315€ |
| Redacción del PFC | 3615,13€ |
| Subtotal | 55858,96 |
| IGIC (7%) | 3910.123 |
| Total | 59769,083 |

*Tabla 8.6: Coste total del Proyecto Fin de Carrera.*

## Presupuesto total

En la tabla 8.7 se recogen todos los costes asociados al Proyecto Fin de Carrera y el importe total del mismo después de impuestos.

Así pues, D. Miguel Ángel Tejedor Hernández declara que el presupuesto para el Proyecto de Fin de Carrera *Titulo* asciende a un total de *cincuenta y nueve mil setecientos sesenta y nueve con cero ochenta y tres céntimos* (59769,083€).

Las Palmas de Gran Canaria, 1 de septiembre de 2015

Fdo. Miguel Ángel Tejedor Hernández

# SPECIFICATIONS

En el pliego de condiciones descrito en este PFC se exponen las condiciones bajo las cuales se ha desarrollado el proyecto. A continuación se describe de forma muy breve el conjunto de los componentes hardware y software empleadas durante la realización del proyecto, así como de las muestras biológicas que se han utilizado.

## Recursos hardware

- Cámara Headwall's Hyperspec VNIR: Cámara de imágenes hiperespectrales destinada tanto a la industria como a la investigación. Es capaz de medir simultáneamente las componentes ópticas del espectro y la localización espacial de objetos sobre una superficie. En conjunto con una cámara monocromática y una lente adecuada, la Headwall's Hyperspec forma un sistema hiperespectral en el rango VNIR (380nm-1000nm), con una resolución espectral de 2-3nm.
- Cámara Headwall's Hyperspec NIR: Cámara de imágenes hiperespectrales destinada tanto a la industria como a la investigación. Es capaz de medir simultáneamente las componentes ópticas del espectro y la localización espacial de objetos sobre una superficie. En conjunto con una cámara monocromática y una lente adecuada, la Headwall's Hyperspec forma un sistema hiperespectral en el rango VNIR (900nm-1700nm), con una resolución espectral de 4-5nm.
- Ordenador personal HP Pavilion dv6: Ordenador portátil en el que se ha instalado el MATLAB 2010a y se ha realizado todo el procesado de las imágenes. Además, se ha redactado la memoria de este Proyecto Final de Carrera.

  Las principales características de este PC son:
  - Procesador Intel Core i7-4510U a 2,60 GHz.
  - 8 GB de memoria RAM.
  - 1 TB de disco duro.

- Disco duro extraíble Western Digital de 1 TB: Utilizado para guardar las imágenes procesadas.
- Impresora HP LaserJet 2430 DTN: Utilizada para la impresión del material necesario para el desarrollo del proyecto y de la presente memoria.

## Recursos software

- MATALB 2010a: Herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio que permite. Facilita el análisis y la visualización de datos.

## Recursos biológicos

Las muestras biológicas que se han empleado en este PFC se corresponden con biopsias cerebrales extraídas durante intervenciones quirúrgicas por el departamento de Neurocirugía del Hospital Universitario de Las Palmas de Gran Canaria Dr. Negrín. Estas biopsias han sido procesadas y diagnosticadas por el departamento de Anatomía Patológica de este mismo hospital. Por otra parte se han utilizado muestras biológicas obtenidas directamente durante intervenciones quirúrgicas mediante el uso de las cámaras hiperespectrales descritas en los recursos hardware y destinadas para tal fin. El empleo de estas muestras en proyectos de investigación está contemplado en los *ethical issues* del proyecto HELICoiD.

Las Palmas de Gran Canaria, 1 de septiembre de 2015

Fdo. Miguel Ángel Tejedor Hernández