Tesis Doctoral

Repositorio digital de imágenes hiperespectrales basado en técnicas de desmezclado con implementación en GPUs

A new digital repository for hyperspectral images based on spectral unmixing and implemented on GPUs

Autor: Jorge Sevilla Cedillo

DEPARTAMENTO DE TECNOLOGÍA DE LOS COMPUTADORES Y DE LAS COMUNICACIONES

Conformidad del Director: Antonio Plaza Miguel

Fdo.:

2014

Acknowledgement

Despite the first lines of this text are in English, I have decided to write this section in Spanish, in order to reach all people to whom it is dedicated.

He decidido escribir esta parte de la tesis en español para que llegue al oído de todas aquellas personas a las que va dedicada. Esas personas que me han acompañado a los largo de los años en mi trayectoria tanto académica como profesional. En los siguientes párrafos he intentado resumir todos los sentimientos que ahora mismo me invaden.

Antes de nada, quiero dar las gracias al Dr. Antonio Plaza, director de esta tesis doctoral. Quien desde que nos conocimos ha sido además de mi tutor un amigo. Que me ha guiado, aconsejado y ha hecho todo lo que estaba en su mano para que este trabajo saliera adelante.

A mis padres, Juan y María, esa pareja inseparable que me ha llevado de la mano desde el momento que llegué a sus vidas y nunca la han soltado. Les doy las gracias por todos los esfuerzos que han hecho para que crezca tanto personal como profesionalmente. Su apoyo y convicción fue esencial para que decidiera estudiar en la universidad.

También agradezco el apoyo incondicional del resto de mi gran familia, de mis hermanos Miguel, Jacinto, David; de mis cuñadas Marisol, Julia, Inma; de mis sobrinos Daniel, Miguel Ángel, Beltrán y Valeria.

Por último, agradecer a las muchas personas que me han acompañado en mi larga trayectoria académica, entre ellos mis amigos, mis compañeros de Hypercomp, mis compañeros de Timisoara, mis compañeros de universidad, mis tíos y primos, y un largo etcétera.

A todos ellos, gracias.

Resumen

La principal contribución del presente trabajo de tesis doctoral viene dada por el diseño e implementación de un nuevo repositorio digital con funcionalidad de recuperación basada en contenido para imágenes hiperespectrales de la superficie terrestre, obtenidas de forma remota mediante sensores aerotransportados o de tipo satélite. En los últimos años, el número de imágenes hiperespectrales se ha incrementado de forma notable, aunque solamente una pequeña parte de las mismas está disponible para uso público y las imágenes se encuentran diseminadas en diferentes localizaciones y formatos. Por tanto, el desarrollo de un repositorio estándar para imágenes hiperespectrales es un objetivo altamente deseado. En particular, en este trabajo de tesis se desarrolla un nuevo repositorio digital compartido para imágenes hiperspectrales obtenidas de forma remota, que permite la carga de nuevas imágenes hiperespectrales junto con sus meta-datos, verdad-terreno y resultados de los análisis (información espectral). Dicho repositorio se presenta como un servicio web para permitir la gestión de las imágenes a través de una interfaz web, que está disponible online en http://www.hypercomp.es/repository. No obstante, la principal contribución del presente trabajo de tesis doctoral es el desarrollo de un sistema eficiente de recuperación de imágenes basada en la información espectral previamente extraída mediante técnicas de desmezclado espectral. Dicho sistema permite la búsqueda de imágenes dentro de una base de datos mediante la comparación con los componentes espectralmente puros de la imagen, lo cual supone una novedad significativa en este campo de estudio. Para la extracción de dicha información espectral se han implementado varias cadenas completas de desmezclado espectral que permiten recuperar y filtrar las imágenes utilizando la semejanza espectral y la abundancia de una verdad-terreno dada. Con vistas a gestionar el coste computacional de extraer la información necesaria para catalogar nuevas imágenes hiperespectrales en el sistema, se recurre a unidades de procesamiento gráfico (GPUs). Las cuales han sido utilizadas de forma satisfactoria para acelerar los cómputos relacionados con los datos hiperespectrales de gran dimensionalidad. Es importante destacar que hay muy pocos trabajos en la literatura que utilicen explícitamente la información espectral para la búsqueda, y ninguno de ellos ha utilizado GPUs para su implementación eficiente. Además, para liberar la carga computacional del servidor web, el presente sistema permite la gestión de computación distribuida utilizando varios clusters de computadores (con arquitecturas CPU y GPU).

Además, esta tesis presenta el desarrollo de un geo-portal para almacenar y devolver de forma eficiente productos de imágenes de la superficie terrestre obtenidas por los sensores MODIS y SEVIRI. Este sistema se presenta como una aplicación web, disponible online en http://www.ceosspain.lpi.uv.es, que permite realizar una búsqueda de imágenes por localización geográfica, además de otros filtros como tipo de sensor, fecha de adquisición, o cobertura nubosa de la imagen. Con con el desarrollo de este sistema se ha abarcado un estudio completo de repositorios digitales para datos de la superficie terrestre, proponiendo una nueva contribución altamente novedosa en este campo. Los productos obtenidos a partir de las imágenes resultan de gran utilidad en muchos estudios, y más si los productos pueden

ser fácilmente filtrados para regiones de interés concretas, tal y como permite la nueva herramienta desarrollada en el presente trabajo.

Abstract

The main contribution of the present thesis work is the design and implementation of a new digital repository with content-based image retrieval functionality for remotely sensed hyperspectral images, collected by airborne or spaceborne Earth observation instruments. Over last years, the amount of hyperspectral images has been significantly increasing, although only a small part of them are available for public use and they are spread among different storage locations and formats. Therefore, the development of a standardized hyperspectral data repository is a highly desired goal. Specifically, in this thesis work we develop a new shared digital repository for remotely sensed hyperspectral data, which allows uploading new hyperspectral data sets along with meta-data, ground-truth and analysis results (spectral information). Such repository is presented as a web service for facilitating the advanced management of images through a web interface, and it is available online from http://www.hypercomp.es/repository. Most importantly, the developed system includes a spectral unmixing-guided content-based image retrieval (CBIR) functionality which allows searching for images from the database using spectrally pure components or endmembers in the scene. Several full spectral unmixing chains are implemented for spectral information extraction, which allows filtering images using the similarity of the spectral signature and its associated abundance in the scene. In order to accelerate the process of obtaining the spectral information for new entries in the system, we resort to efficient implementations of spectral unmixing processing chains on graphics processing units (GPUs), which have been successfully exploited to accelerate hyperspectral-related computations. There are a few works in the literature dealing explicitly with the use of spectral information to perform CBIR from hyperspectral repositories, but none of them are implemented on GPUs. Furthermore, in order to relieve the load of the computational server, our system provides distributed computing management using several clusters (with CPU and GPU architectures).

In addition, another important contribution of this thesis is the development of a new geo-portal for storing and efficiently retrieving MODIS/SEVIRI remote sensing products, since a complete design of a standardized image data repository should also store advanced remote sensing products. In this regard, our newly developed system is presented as a web tool, available online from http://www.ceosspain.lpi.uv.es, which allows for the retrieval of remote sensing data by geographic location, and implements other filters based on the image acquisition date, type of sensor or cloud cover. This development is expected to be quite useful in many studies, in particular, if the products can be easily retrieved for a given area of interest as it is possible with the newly developed tool. In fact, this capability has the potential to impact the activities of many remote sensing users such as farmers, engineers, water managers, as well as the scientific community devoted to Earth observation in general.

Contents

List of Figures

Table Index

Chapter 1

Introduction

1.1 Context and motivation

Content-based image retrieval (CBIR) systems intend to retrieve images from databases using the pixel information contained in those images. The incorporation of CBIR [1] techniques into remote sensing data repositories offers significant advantages from the viewpoint of effectively managing, storing and retrieving large volumes of remotely sensed data [2]. Hyperspectral imaging (also known as imaging spectroscopy [3]) is a fast growing area in remote sensing. Hyperspectral scenes are comprised of hundreds of images (at different wavelength channels) for the same area on the surface of the Earth, thus generating very large data volumes that need to be efficiently stored and managed. The interpretation of remotely sensed hyperspectral scenes is also an increasingly relevant research topic involving many different analysis techniques [4].

In order to provide an idea of available and future missions for Earth observation using hyperspectral instruments, Table 1.1 provides a summary of the main characteristics of nine hyperspectral instruments: three airborne -the hyperspectral data collection experiment (HYDICE) [5], the airborne visible infra-red imaging spectrometer (AVIRIS) [6]) and the reflective optics system imaging spectrometer (ROSIS [7])- and six spaceborne -Hyperion [8], the German spaceborne imaging spectrometer mission (EnMAP) [9], the Italian hyperspectral precursor and application mission (PRISMA) [10], the compact high resolution imaging spectrometer (CHRIS) [11], the hyperspectral infra-red imager (HyspIRI) [12] and infrared atmospheric sounding interferometer (IASI) [13])-. From this list, EnMAP, PRISMA and HyspIRI are still not operational. The spatial resolutions are generally higher for airborne instruments. The spectral coverage of HYDICE, AVIRIS, Hyperion, EnMAP, PRISMA and HyspIRI corresponds to the visible, the near-infrared, and the shortwave infra-red spectral bands (typically, from 0.4 to 2.5 nanometers), whereas CHRIS and ROSIS cover only the visible bands and IASI also covers the mid-infra-red and the long-infrared bands. The number of spectral bands is approximately 200 for HYDICE, AVIRIS, Hyperion, EnMAP, PRISMA and HyspIRI, with a spectral resolution of the order of 10 nanometers. The lowest number of bands is provided by CHRIS, with 63 bands and spectral resolutions of 1.3 and 12 nanometers (depending on the region of the spectrum). Quite opposite, instruments such as IASI provide up to 8461 spectral bands. In all cases, the spectral resolution is very high (offering a huge potential to discriminate materials).

Several factors make the analysis of hyperspectral data a complex and hard task, calling for sophisticated analysis methods. Among these factors, we emphasize the presence of spectral mixing effects that have been generally approached by identifying a set of spectrally pure signatures in the scene

Table 1.1: Main characteristics of several (available and new) hyperspectral imaging instruments.

	Altitude (Km)	Spatial resolution (m)	Spectral resolution (nm)	Coverage (μm)	Number of bands	Data cube size (samples×lines×bands)
HYDICE	1.6	0.75	7-14	0.4-2.5	210	200×320× 210
AVIRIS	20	20	10	0.4-2.5	224	512×614×224
ROSIS	3	1-6	4	0.4-0.9	115	340×610× 115
HYPERION	705	30	10	0.4-2.5	220	660×256×220
EnMAP	653	30	6.5-10	0.4-2.5	228	1000×1000×228
PRISMA	614	5-30	10	0.4-2.5	238	400×880×238
CHRIS	556	36	1.3-12	0.4-1.0	63	748×748×63
HyspIRI	626	60	4-12	0.38-25 & 7.5-12	217	620×512×210
IASI	817	V: 1-2 km H: 25 km	0.5 cm^{-1}	3.62-15.5	8461	765×120×8461

(called *endmembers* in spectral unmixing terminology) and their corresponding abundance fractions in each (mixed) pixel of the scene [14]. An additional issue is the extremely high dimensionality and size of the data, resulting from the very fine spatial, spectral and temporal resolutions currently provided by hyperspectral instruments (see Table 1.1). This demands fast computing solutions that can accelerate the interpretation and efficient exploitation of hyperspectral data sets in various applications [15, 16, 17].

Although the amount and volume of hyperspectral image data has been significantly increased in recent years, with a large number of data sets already collected over different locations over the world and several new missions under development, the data sets which are available for public use are spread among different storage locations and present significant heterogeneity regarding the storage format, associated meta-data (if any), or ground-truth availability. As a result, only a few data sets are recurrently used to validate hyperspectral imaging applications, and available data are highly fragmented. Nowadays, it is estimated that a large fraction of collected hyperspectral data sets are never used but simply stored in different databases. Even if the use of standardized benchmark data is quite interesting from the viewpoint of algorithm comparison, there is a need to increase the pool of benchmark hyperspectral data sets available to the community in order to allow a more appropriate selection of specific test data for different applications. This functionality is already available in other systems which are able to effectively provide remotely sensed data on-demand and with high retrieval performance [18, 19, 20, 21]. At present there is no common repository of hyperspectral data sets which can effectively distribute such data among potential users. Since the amount and volume of hyperspectral data is expected to significantly increase with the new missions described in Table 1.1, a highly desirable objective in the hyperspectral imaging community is to develop new tools to effectively share large amounts of hyperspectral data together with their high-level associated information (e.g., ground-truth, analysis results, pointers to bibliographic references describing previous results on the data, etc.)

On the other hand, the complete design of a standardized image data repository to store advanced remote sensing products could be extremely useful in many studies, in particular, if the products could be easily retrieved for a given area of interest. In fact, this capability has the potential to impact the activities of many remote sensing users such as farmers, engineers, water managers, as well as the scientific community devoted to Earth observation and remote sensing. A large number of remotely sensed multispectral data sets have been collected in recent years by Earth observation instruments such as the moderate resolution imaging spectroradiometer (MODIS) aboard the Terra/Aqua satellite, or the

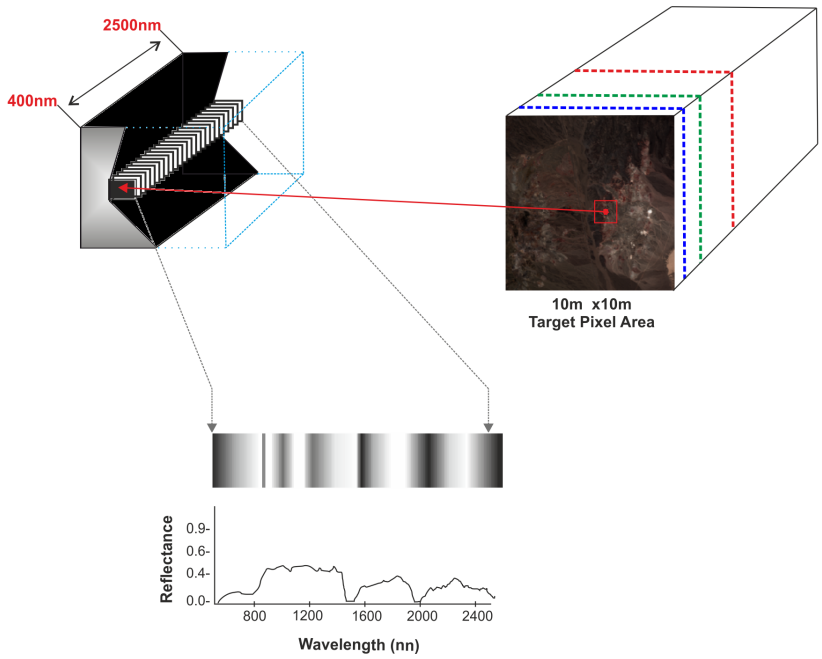


Figure 1.1: Spectral signature definition.

spinning enhanced visible and infrared imager (SEVIRI) aboard the geostationary platform Meteosat Second Generation (MSG). The advanced remote sensing products resulting from the analysis of these data are useful in a wide variety of applications, but require significant resources in terms of storage, retrieval and analysis. Despite the wide availability of these products, the data coming from these instruments are spread among different locations and retrieved from different sources, and there is no common data repository from which the data or the associated products can be retrieved.

1.2 Hyperspectral analysis

Hyperspectral imaging is an emerging and fast growing area in remote sensing. It is concerned with the measurement, analysis, and interpretation of spectra acquired from a given scene (or specific object) at a short, medium or long distance by an airborne or satellite sensor. The main feature of hyperspectral images is the high resolution that they present in the spectral domain, since they are collected by instruments able to measure hundreds of *narrow* spectral bands corresponding to continuous wavelength channels [22]. In contrast, multispectral imaging instruments are only able to provide information in a few spectral bands [23]. In fact, hyperspectral imaging instruments have experienced a significant evolution [24]. The very high spectral resolution of remotely sensed hyperspectral data [25], rooted in technological modelling and processing advances, have fostered a strong interest in this image modality at an unprecedented rate in recent years. Indeed, the very high spectral resolution of hyperspectral data offers very significant potential in the identification of materials and their properties [26].

The wealth of information available from hyperspectral imaging instruments has opened ground-breaking perspectives in several applications, including environmental modelling and assessment for Earth-based and atmospheric studies, risk/hazard prevention a response including with land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination, target

detection for military and defense/security purposes, and urban planning and management studies, among many others [26]. For instance, AVIRIS is now able to record the visible and near-infrared spectrum (wavelength region from 400 to 2500 nanometers) of the reflected light of an area 2 to 12 kilometers wide and several kilometres long, using 224 spectral bands. As Fig. 1.1 shows, the resulting data volume can be seen as a data cube with two spatial and one spectral dimension, in addition of each pixel can be considers as a high-dimensional vector where the values of a pixel comprise its associated spectral signature. The spectral signatures is characteristic of each observed object and can be used as a fingerprint for identification purposes.

Although AVIRIS is a widely used platform, it constitutes only one source of hyperspectral data. Table 1.1 summarizes other international Earth observation missions with hyperspectral sensors already launched or to be launched in the near future. While in this work our focus is on remote sensing applications, hyperspectral sensors have been widely used in many other areas. For instance, hyperspectral cameras are now routinely used for industrial quality control [27], food inspection [28], forensics [29] and medical imaging purposes [30]. Hyperspectral microscopes are also gaining popularity in applications such as nanotoxicology [31], chemometrics [32] and pharmacology [33].

One of the main problems involved in hyperspectral data exploitation is spectral unmixing [4], as many of the pixels collected by imaging spectrometers such as AVIRIS are highly mixed in nature due to spatial resolution and other phenomena. For instance, it is very likely that the pixel labelled as 'vegetation' in Fig. 1.2 is actually composed of several types of vegetation canopies interacting at sub-pixel levels. The same comment applies to the 'soil' pixel, which may comprise different types of geological features. As a result, spectral unmixing is a very important task for hyperspectral data exploitation since the spectral signatures collected in natural environments are invariably a mixture of the pure signatures of the various materials found within the spatial extent of the ground instantaneous field view of the imaging instrument. Among several techniques designed to deal with the inherent complexity of hyperspectral images in supervised fashion [4, 34], linear spectral unmixing follows an unsupervised approach which aims at inferring pure spectral signatures, called *endmembers*, and their material fractions at each pixel of the scene.

Let us assume that a remotely sensed hyperspectral image denoted by $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}n]$ with n pixels and l bands or different spectral channels. Let us also denote a given pixel of the image as $\mathbf{y}_j$. The pixel can be represented as a linear combination of a set of spectral signatures, weighted by their abundances. Under the linear mixture model assumption [35, 36], each pixel vector in the original scene can be modelled using the following expression:

$$\mathbf{y}_j \approx \mathbf{M}\alpha + \mathbf{n} = \sum_{i=1}^{p} \mathbf{m}_i \alpha_i + \mathbf{n}, \tag{1.1}$$

where $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^{p}$ is a matrix containing p *endmember* signatures, $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_p]$ is a p-dimensional vector containing the abundance fractions for each of the p *endmembers* in $\mathbf{M}$, and $\mathbf{n}$ is a noise term.

In order to perform spectral unmixing of hyperspectral scenes, a well-defined spectral unmixing chain consisting of several steps has been traditionally performed [14]. Specifically, the full unmixing chain used in this work takes as input from the original image $\mathbf{Y}$ and produces as a result a set of p spectral signatures or *endmembers* and the abundances of such pixels in the image. Fig. 1.3 shows the different steps involved in the processing chain, which are briefly summarized next.

- **Estimation of the number of endmembers**. This first stage conducts an estimation of the

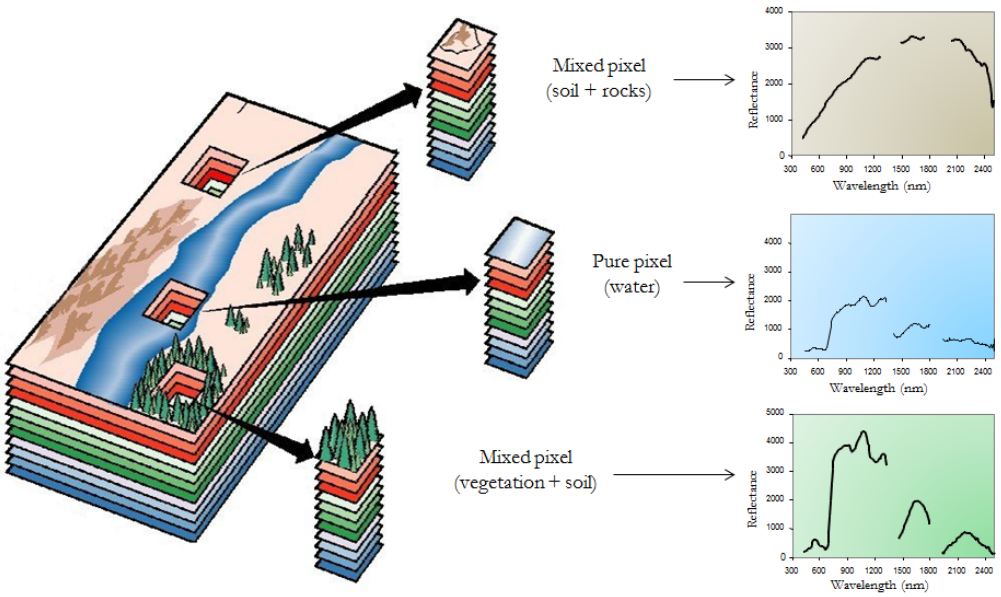


Figure 1.2: Concept of mixed pixel in hyperspectral images.

number of *endmembers* present in the hyperspectral image $\mathbf{Y}$, aimed at finding how many pure spectral signatures are comprised by the scene under analysis.

- **Dimensional reduction**. The number of image bands l is generally higher than the number of *endmembers* p. This allows identifying an appropriate subspace that facilitates dimensionality reduction, improving algorithm performance and reducing computational complexity. Furthermore, if the linear mixture model is accurate, the signal subspace dimension should be one order less than the number of endmembers.

- **Endmember extraction**. This stage consists of identifying the spectral signatures of the endmembers in the scene. This process can be solved from different points of view [37, 38, 39], although this work is focused on *geometric* and *sparse* methods. The *geometric* methods consider that all the mixed pixels are inside a simplex defined by the *endmembers* (see Fig. 1.4). On the other hand, *sparse* methods use spectral libraries to solve the mixed pixel problem [40].

- **Abundance estimation**. Based on the identified endmembers, the abundance estimation stage consists of representing every hyperspectral image pixel as the linear combination of the pure pixels, i.e. to calculate the fractional coverage of such *endmembers* in every pixel of the hyperspectral image. This information allows for the generation of abundance maps comprising the coverage of each *endmember* in each pixel of the scene. Abundance values are normally constrained to be non-negative and to sum to one (they belong to the probability simplex). There are, however, some hyperspectral unmixing approaches in which the *endmember* determination and inversion steps are implemented simultaneously.

As we mentioned above, hyperspectral images are often characterized by extremely high dimensionality and size, which means a high computational cost of executing a full spectral unmixing chain. In recent years, several efforts exploiting high performance computing technology can be found in the hyperspectral unmixing literature, including [41] and references therein, which demonstrates that

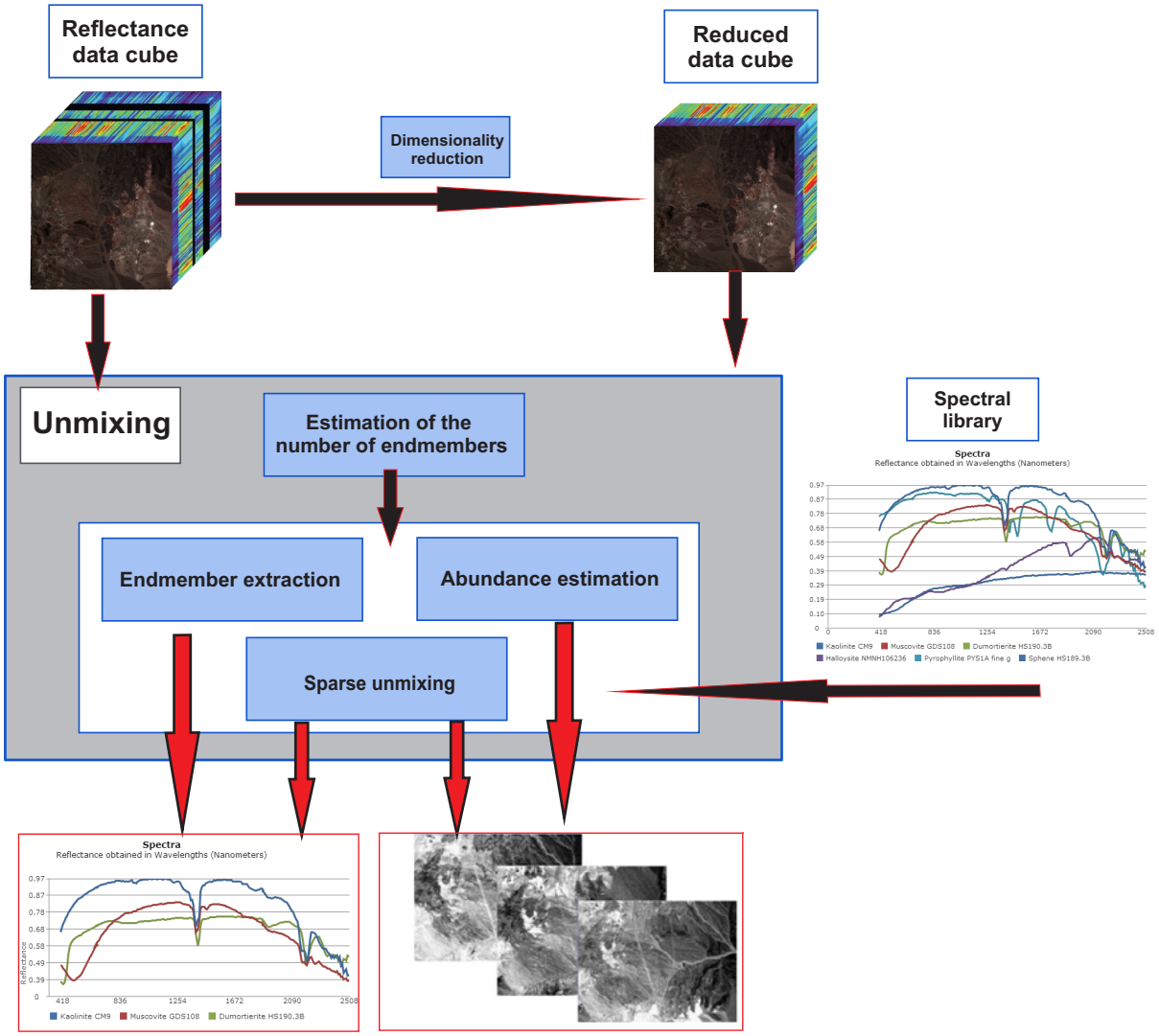


Figure 1.3: Spectral unmixing chain.

the processing of high-dimensional remote sensing scenes can be efficiently performed using specialized hardware accelerators such as FPGAs or GPUs.

1.3 Objectives

The main goal of this thesis work is to develop new data repositories (available online as web services) for sharing and retrieving remotely sensed hyperspectral/multispectral data in an intelligent manner. The developed systems will be able to provide remote sensing data retrieval, using GPU implementations of spectral unmixing techniques (in the case of hyperspectral image repositories) and geolocation algorithms (in the case of multispectral product repositories). This represents an innovative contribution in the remote sensing community as there are currently no data repositories with the functionalities that we are providing. In order to achieve this general objective, we will address a number of specific objectives which are listed below:

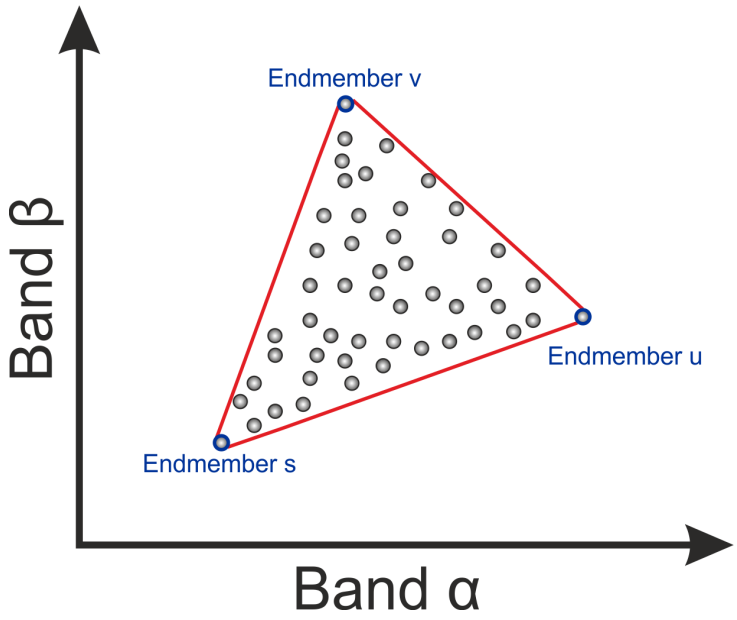


Figure 1.4: Graphic interpretation of the linear mixture model

- To investigate the available spectral unmixing techniques used currently in the hyperspectral imaging community including their GPU implementations, evaluating the advantages and disadvantages presented by them, with the ultimate goal of validating full spectral unmixing chains that will be included in our newly designed systems to extract pure elements and their abundances from high-dimensional hyperspectral images. Such implementations will be carried out in both CPU and GPU architectures from distributed computing resources, so that a remote execution manager for distributed computing will be implemented and validated.

- To analyse, implement and validate advanced matching techniques intended to measure the similarity and abundances between the pure elements of the images and a set of reference spectral signatures.

- To design and develop a standardized database for storing all the different formats of multi/hyperspectral images publicly available, such as images from the widely-used AVIRIS and ROSIS sensors, and by other instruments such as MODIS and SEVIRI. The database schema will include all the main image features, ground-truth classification image, involved publications and results from applying analysis algorithms over the images.

- To study available software resources for web service development in order to manage data repositories for remotely sensed multi/hyperspectral data.

- To design and implement a web tool able to manage services such as new images addition, unmixing algorithm execution or image retrieval based on the results of those algorithms. Mainly, the system will be composed by an user interface which gives access to images through a web service.

- To collect a relevant amount of images and then validate the unmixing-based image retrieval system functionality through a systematic procedure using both synthetic and real images.

- To design and develop a standardized database for storing a wide amount of remote sensing image formats and products.

1.4 Main contributions of the thesis

The digital repository that we present in this work includes some new important features. These innovative contributions can be summarized as follows:

- The presented system uses several full unmixing chains to perform the cataloguing and retrieval of hyperspectral images in the repository, as already discussed in [42] and [43]. However, the proposed implementation allows for more complex search criteria than the ones presented in previous contributions. Specifically, the queries in our system can be defined by the spectral information (provided by *endmember* signatures) and/or the spatial information (provided by abundances), in joint or separate fashion.

- Another important contribution of our system is the possibility to use previously available spectral libraries as the main criteria in order to perform the query. In other words, our tool allows automatically uploading a spectral library and using the spectra in such library in order to perform content-based retrieval. The user may select a few spectra from the library or even the full library, using the selected spectral signatures as input to a query. Hence, the increased availability of open repositories of spectral libraries such as the SPECCHIO project[1] is a good complement to the system that we present in this contribution.

- The processing modules included in our system comprise many well-established techniques in all parts of the full hyperspectral unmixing chain. Specifically, our system currently includes two methods for estimating the number of *endmembers* (VD and HySime), two algorithms for identifying the spectral signatures of the *endmembers* directly from the data (N-FINDR and OSP), and two methods for estimating the fractional abundances, unconstrained (UCLS) and non-negatively constrained (ISRA). GPU implementations for all these methods are included in the system, thus allowing for fast cataloguing and meta-data generation for new hyperspectral image scenes. In addition, we also provide a technique based on sparse unmixing concepts (which are more suitable for large spectral libraries) to perform the retrieval of images from the database.

- Although the contribution [43] already discussed an efficient implementation of an unmixing-based CBIR system for hyperspectral imagery, this implementation was specifically developed for heterogeneous networks of workstations or clusters of computers, without taking advantage of hardware accelerators such as GPUs which are now widely available in modern clusters and supercomputers. In this regard, the proposed system expands the parallel features of the system in [43] and includes the use of GPU accelerators, thus increasing the computational performance significantly.

- In addition, previous developments such as [42] or [43] were not fully available to the community. In this contribution, we present a fully open system, with an advanced user interface, and implemented on two large supercomputing facilities: a first cluster of GPUs is available at the Center of Advanced Technologies in Extremadura (CETA-Ciemat), which is one of the most powerful clusters of GPUs in Spain; and the second GPUs cluster is from West University of Timisoara, which is one of

[1] http://www.specchio.ch

the most powerful clusters of GPUs in Romania. As a result, the CBIR system that we describe in this contribution is completely available for public use from http://hypercomp.es/repository. It contains several synthetic and real hyperspectral data sets that interested readers can use to conduct their own experiments and include additional hyperspectral data sets in the repository.

- Last but not least, despite the fact that web tools like *MRTWeb 2.0* [44] or *LAADS Web* [45] have available products from several multispectral sensors, they are not specifically focused on offering a common data repository in which the data or the associated products can be retrieved. In this thesis, we present a standardized data repository for multispectral data such as images as their products. A freely available web tool (available online from http://ceosspain.lpi.uv.es) has been implemented to manage the repository, which facilitates the remote access (through a web browser) to a massive collection of processed remotely sensed data sets such as validated MODIS and SEVIRI products. The data processing is accomplished in real-time, i.e. as soon as the MODIS/SEVIRI data are collected by the receiving antenna. This makes the final product immediately available to the wide remote sensing community and allows for a detailed exploration of the high-level products obtained after the data processing step.

1.5 Thesis organization

The present thesis has been structured in a series of chapters which can be summarized as follows (see Fig 1.5 for a graphical overview indicating the relationships between the different chapters):

1. **Introduction**. In this chapter, we have described the main motivations and objectives that have led to the development of the thesis. In addition, we introduced the hyperspectral imaging concept and further explained the importance of mixed pixels and their management in remote sensing data interpretation.

2. **System design**. In this chapter, we introduce the web service concepts describing their components, as well as their internal communication and work-flow. Furthermore, we describe the open source framework used to develop the proposed web system. Then, we fully describe the proposed new repository which is implemented as a web service, specifically, we detail the system architecture, database structure and distributed computing resources supported for algorithm execution. At the end of this chapter, we also show how to perform the uploading and searching of images through the web interface.

3. **Unmixing-based image retrieval system**. This chapter describes the content-based image retrieval system implemented over the repository using several full spectral unmixing chains which are used in order to generate suitable meta-data for retrieval purposes. In particular, we emphasize the searching methodology and the GPU implementation of the unmixing chain algorithms adopted by the system. The chapter concludes with a detailed performance analysis of the system by comparing its retrieval accuracy using synthetic and real hyperspectral images.

4. **Sparse unmixing-based image retrieval approach**. This chapter presents a CBIR system using sparse unmixing techniques, in which a sparse algorithm is used to generate the meta-data for retrieval purposes. The algorithm is described and the achieved results from this approach are compared with the results from the unmixing-based approach described in the third chapter of the thesis.

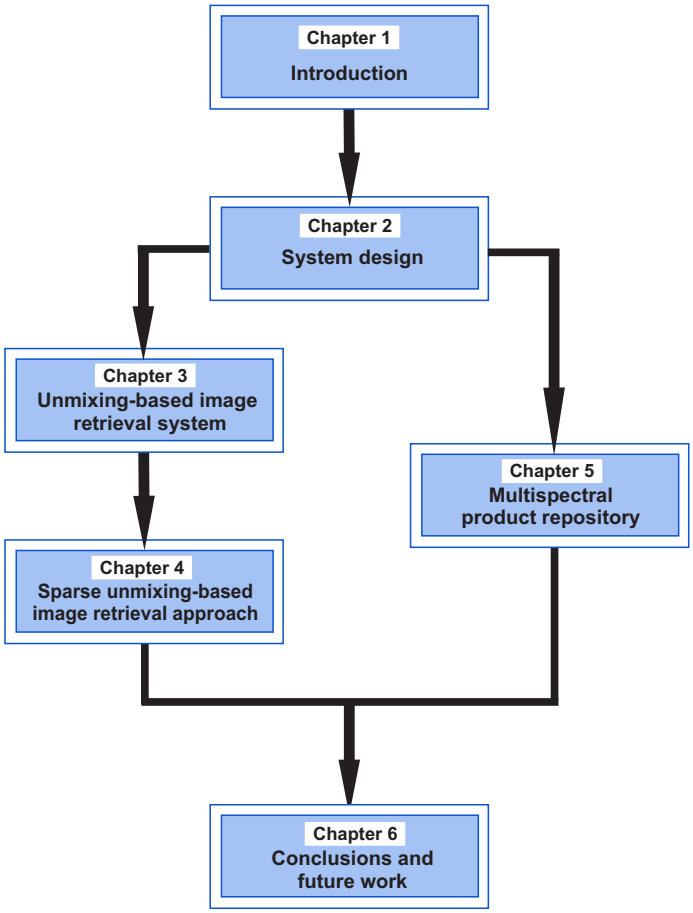


Figure 1.5: Thesis organization.

5. **Multispectral product repository**. In this chapter, we describe a second repository for multispectral imagery products. More specifically, we present a standardized and publicly repository for storing a wide amount of formats of multispectral images and their products. This data repository is also available online providing geo-location retrieval capabilities. In addition, the products from MODIS/SEVIRI included in this repository are also thoroughly described.

6. **Conclusions a future research work**. This chapter concludes by summarizing the main contributions of this thesis, as well as with a presentation of the most plausible research lines that could be explored in future developments of this work.

The thesis concludes with the list of references used during the elaboration of this document. In order to facilitate the reading of the document, Table 1.2 shows a list of the acronyms that will be used throughout the thesis. In an appendix, we include the publications resulting from the present thesis work, together with a statement of the main contributions and relevant aspects which are highlighted for each individual contribution. Specifically, this thesis has resulted in 4 journal citation reports (JCR) papers (2 submitted and the rest already published), and 8 peer-reviewed international conference papers.

Table 1.2: List of acronyms used in this thesis.

Acronyms	
ADMM	Alternating Direction Method of Multipliers [46]
AMEE	Automated Morphological Endmember Extraction [47]
ANC	Abundance Non-negativity Constraint [36]
ASC	Abundance Sum-to-one Constraint [36]
AVIRIS	Airbone Visible Infra-Red Imaging Spectrometer [6]
BRDF	Bidirectional Reflectance Distribution Function [48]
CBIR	Content-Based Image Retrieval
CIFS	Common Internet File System
CSS	Cascade Style Sheet
CSUNSAL	Constrained Sparse Unmixing by variable Splitting and Augmented Lagrangian
CUDA	Compute Unified Device Architecture
EE	Earth Explorer [49]
EOS	Earth observing system
EOSDIS	Earth Observing System Data and Information System
FCLS	Fully Constrained Least Squares [36]
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
FVC	Fraction of Vegetation Cover [50]
GPU	Graphic Processing Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IPL	Image Processing Laboratory
HySime	Hyperspectral Subspace Identification by Minimum Error [51]
ISRA	Image Space Reconstruction Algorithm [52]
JSON	JavaScript Object Notation
MNF	Minimum Noise Fraction [53]
MODIS	Moderate Resolution Imaging Spectroradiometer
MRT	MODIS Retrieval Tool [44]
MSG	Meteosat Second Generation [54]
NAPC	Noise-adjusted Principal Components [55]
NASA	National Aeronautics and Space Administration
NDVI	Normalized Difference Vegetation Index [56]
N-FINDR	[57]
LAADS	Level 1 and Atmosphere Archive and Distribution System [45]
LPDAAC	Land Processes Distributed Active Archive Center
LSM	Land/Sea mask
LST	Land Surface Temperature [58, 59]
ORM	Object-Relational Mapping
OSP	Orthogonal Subspace Projection [60]
OSP-GS	Orthogonal Subspace Projection with Gram-Schmidt orthogonalization [61]
PCA	Principal Component Analysis [62]
PPI	Pixel Purity Index [63]
RMSE	Root Mean Square Error [35]
ROSIS	Reflective Optics Spectrographics Imaging System [7]
SAD	Spectral Angle Distance [35]
SSH	Secure Shell
SCP	Secure Copy Protocol
SEVIRI	Spinning Enhanced Visible and Infra-Red Imager [54]
SMAC	Simplified Method for the Atmospheric Correction [64]
SNR	Signal-to-Noise Ratio [65]
SST	Sea Surface Temperature [66, 67]
SUNSAL	Spectral Unmixing by Splitting and Augmented Lagrangian [68]
SVD	Singular Value Decomposition [69]
UCLS	Unconstrained Least Squares [70]
UGC/IPL	Global Change Unit, Image Processing Laboratory, University of Valencia [71]
USGS	United States Geological Survey [72]
VCA	Vertex Component Analysis [73]
VCI	Vegetation Condition Index [74]
VD	Virtual Dimensionality [75]
VIIRS	Visible Infrared Imaging Radiometer Suit
WSD	Web Service Description

Chapter 2

System design

Our system [76, 77] is implemented as a web service, which integrates a web application and computing services. In this way, the system delivers data to a set of clients (i.e., web interfaces), while it also receives content from the clients. An advantage of this approach is that no additional software has to be installed on the client computer, since a web interface only requires a web browser. The main aim of this thesis is the development of a web application to manage hyperspectral data, including both hyperspectral images and unmixing processing results. Specifically, the system also controls algorithm executions and stores the related data in a database. The remainder of this chapter, devoted to the high-level description of our system, is organized as follows. Section 2.1 introduces the web services definition. Section 2.2 describes `Symfony2`, the developing framework used in our work. Section 2.3 describes the software architecture of the system, which is composed of three main layers: 1) *client layer*, which defines the interactions between the user and the system through a web interface; 2) *server layer*, which manages the requests from end-users; and 3) *processing layer*, in charge of more complex processing tasks such as generation of meta-data or image retrieval. In section 2.4, we describe the structure of the database that stores the hyperspectral data in our system. Finally, in section 2.5, we briefly describe the functionalities provided by the web interface, emphasizing its functionalities in terms of image uploading and queries execution.

2.1 Web service definition

A web service is a software system designed to support interoperable machine-to-machine interactions over a network [78]. Other systems interact, regardless of the programming language in which they are written, using the web service through standard communication protocols. In our case, the communication protocol is `HTTP`[1] protocol and the data objects are exchanged encapsulated in `JSON`[2] format. In this section we describe the implementation of web services in our system from a high-level perspective. The remainder of this section is organized as follows. In subsection 2.1.1, we describe the different roles involved in the web service architecture. Subsection 2.1.2 describes the rules for message exchange among roles. Subsection 2.1.3 shows the communication work-flow among the web service components.

[1] http://www.w3.org/Protocols/rfc2616/rfc2616.html
[2] http://json.org

2.1.1 Roles

A web service is an abstract notion that must be implemented by a concrete computational resource. As we illustrate graphically in Fig. 2.1, the *agent* (resource) is the concrete piece of software or hardware that sends and receives messages, while the *service* is characterized by an abstract set of functionalities that should be provided. Thus, the *agents* could be modified or even replaced as long as they have the same communication interface. The purpose of a web service is to provide some functionality on behalf of its *owner* (a person or organization). The *provider* entity is the person or organization that provides an appropriate agent to implement a particular service. A *requester* entity is a person or organization that wishes to make use of a provider entity's web service.

2.1.2 Service and semantic

The *service* description represents a contract governing the mechanics of the interaction with a particular *service*. The mechanics of the message exchange are documented in a web service description (`WSD`), which is a machine-processable specification of the web service's interface. It defines the message formats, data-types, transport protocols, and transport serialization formats that should be used between the requester agent and the provider agent. It also specifies one or more network locations from which a provider agent can be invoked, and may provide some information about the expected message exchange patterns. In essence, the *service* description represents an agreement governing the mechanics of the interaction with that *service*. The *semantics* represent a contract between the requester entity and the provider entity regarding the purpose and consequences of the interaction.

2.1.3 Work-flow

There are many ways that a requester entity might engage and use a web service. In general, the following broad steps are required: (1) as illustrated in Fig. 2.1 the requester and provider entities become known to each other (or at least one becomes know to the other); (2) the requester and provider entities somehow agree on the service description and semantics that will govern the interaction between the requester and provider agents; (3) the service description and semantics are realized by the requester and provider agents; and (4) the requester and provider agents exchange messages, thus performing some task on behalf of the requester and provider entities (i.e., the exchange of messages with the provider agent represents the concrete manifestation of interacting with the provider entity's web service.).

2.2 Symfony2

A *framework* streamlined application is developed by automating many of the patterns employed for a given purpose, furthermore, it adds structure to the code, prompting the developer to write better, more readable, and more maintainable code. Ultimately, a framework makes programming easier, since it packages complex operations into simple statements. There are several frameworks which provide some common elements to the development of web services (such as security, routing, database access, etc.) and which allow focusing the work on the specific aspects of each development. Several web service development tools have been analysed to implement our system, as such as `Symfony2`[3], `Akelos`[4],

[3] http://symfony.com
[4] http://trac.akelos.org/

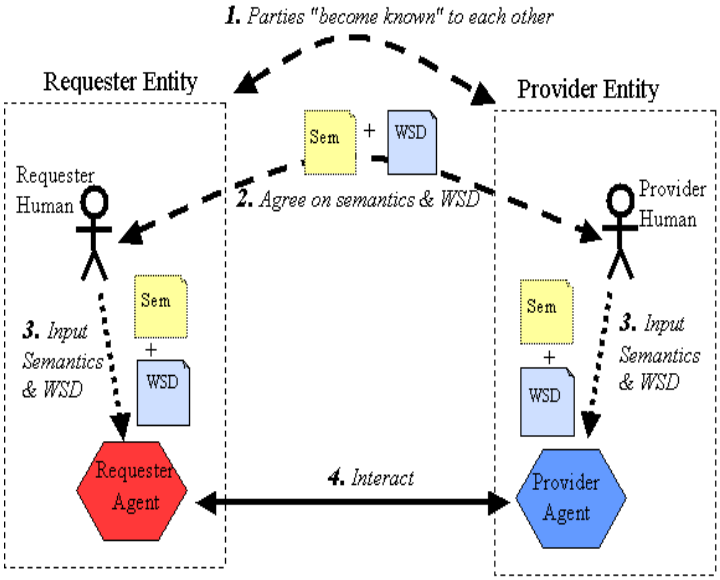


Figure 2.1: Web service architecture.

CakePHP[5], Prado[6], Seagull[7], Yii[8] or Zen[9]. According to our tests, the best candidate has been Symfony2 since it covers all our needs, i.e., it is flexible to include new components. The main developing features and design pattern of this framework are detailed next.

2.2.1 Features

Symfony2 is a complete object-oriented framework written entirely in PHP 5.3[10], which is designed to optimize the development of web applications by way of several key features, so that it separates a web application's business rules, server logic, and presentation views. It contains numerous tools and classes for shortening the development time of complex web applications. Additionally, it automates common tasks which allows the developer to focus entirely on the specifics of an application. It has been thoroughly tested in various real-world projects, and is actually in use for high-demand e-business websites. In the following, we describe the database storing module used in our system, as well as the view module.

2.2.1.1 Storing module

One of the most common and challenging tasks for any web application involves reading information to and from a database. Symfony2 allows to integrate an object/relational abstraction layer with an ORM interface. An important benefit of an object/relational abstraction layer is that it prevents using a syntax that is specific to a given database, since it automatically translates calls to the model objects to

[5]http://cakephp.org/
[6]http://www.pradosoft.com/
[7]http://seagullproject.org/
[8]http://www.yiiframework.com/
[9]http://www.zend.com/
[10]http://php.net/releases/5_3_0.php

SQL queries optimized for the current database at hand. The ORM is defined to access the database in an object-oriented way, and it translates the object logic to the relational logic. In this work, the ORM used is Doctrine[11], which is one of the most important PHP ORM frameworks in the web development community; it communicates with the database regardless the relational database management system [79] used, since the SQL sentences are generated by Doctrine instead of by the programmer. It is compatible with most of the available relational database management engines, including MySQL[12], PostgreSQL[13], Oracle[14], and Microsoft SQL Server[15].

2.2.1.2 View module

Twig[16] is another framework integrated as a module; it is a powerful templates engine which is flexible, efficient and fast. Twig has its own code that is compiled as PHP and produces HTML[17] code. Specially, the inheritance mechanism of templates represents its best feature because it works as an interface for the inheritance from object-oriented programming, i.e., several HTML pages can be produced from base templates.

2.2.2 Design pattern

Symfony2 is based on the model-view-controller pattern which is an application design pattern originally introduced in [80]. The main idea of this pattern is to separate presentation from data and controller from presentation. This kind of separation allows each part of the application focus on exactly one goal. The controller focuses on changing the data from the *model*, and delivering the data to the *view*. Indeed, the *view* is focused on creating representations of the *model*. A step-by-step description of the Symfony2 work-flow follows:

- The user first presents a request for viewing a certain web page.
- The routing system decides which is the appropriate controller for handling such request.
- The controller executes the code associated with the request. The controller is a PHP class designed to attend the user requests.
- Data queries are now demanded by the data manager (*Doctrine* in our case).
- Finally, the retrieved data is rendered the form of view templates and is presented to the user who originated the request.

2.3 System architecture

As shown by Fig. 2.2, the software architecture of the proposed system is formed by different layers, which can be defined by their roles. The system follows a modular design in which the communication between layers is defined using standard data exchange formats and transfer protocols, so that any layer can be modified as long as it can communicate with the rest of the system. Our design has been carried

[11] http://www.doctrineproject.org/
[12] http://www.mysql.com
[13] http://www.postgresql.org.es
[14] http://www.oracle.com/us/products/database/overview/index.html
[15] https://www.microsoft.com/sqlserver/default.aspx
[16] http://twig.sensiolabs.org/
[17] http://www.w3.org/TR/html

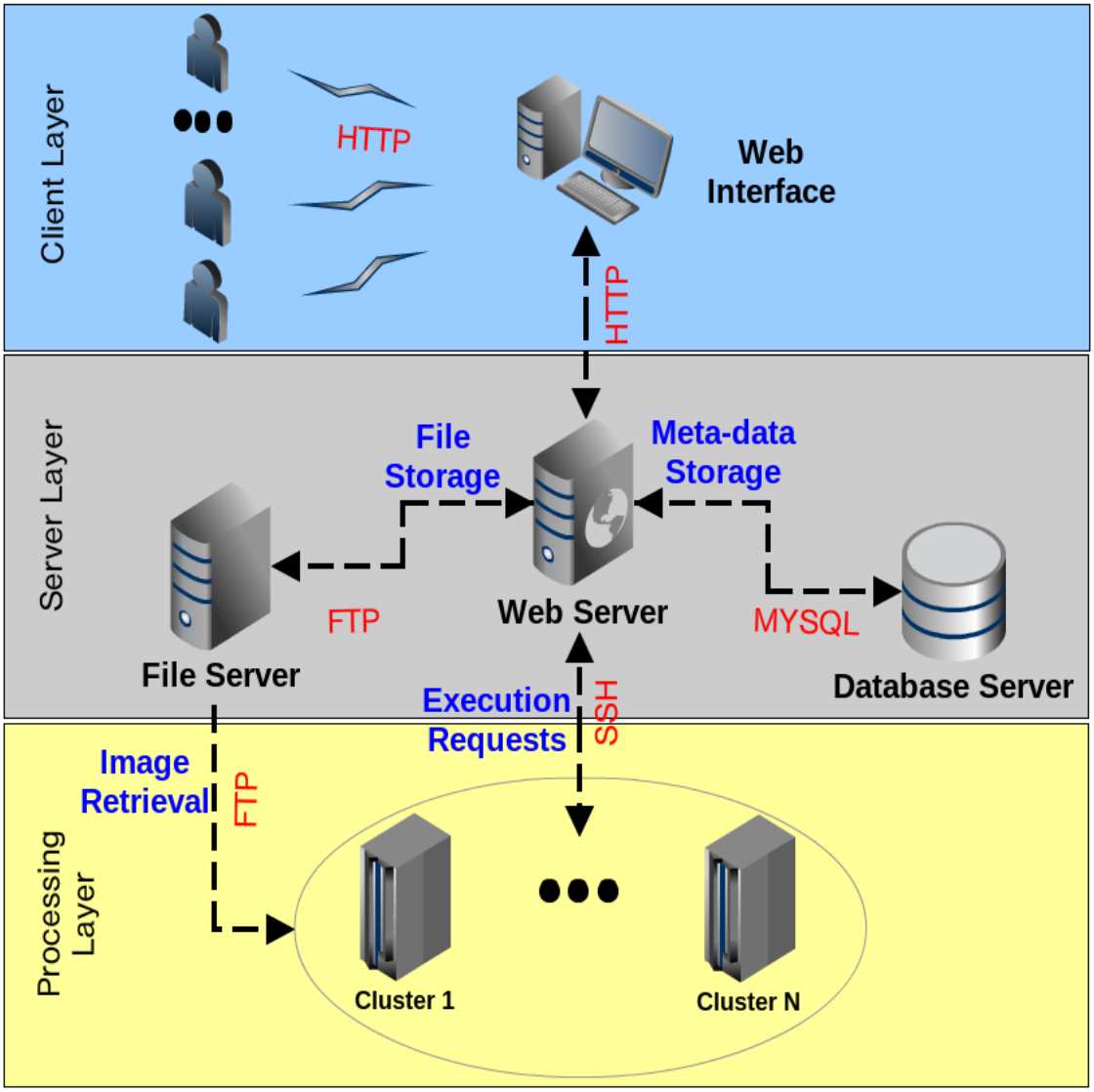


Figure 2.2: Architecture of the proposed hyperspectral image repository.

out using free software tools such as `Symfony2`, described in section 2.2, while the adopted format for data exchange through the layers is `JSON`, an open standard format that uses human-readable text to transmit data objects. In the following, we describe the different software layers that compose the system: 1) *client layer*, which defines the interactions between the user and the system through a web interface; 2) *server layer*, which manages the requests from end-users; and 3) *processing layer*, in charge of more complex processing tasks such as generation of meta-data or image retrieval.

2.3.1 Client layer

This layer defines the interactions between the users (through an internet web browser) and our system, and is responsible for providing users with remote and interactive access to the system. The web

interface has been designed using HTML5[18], which is the last version of HTML, the most widely used web programming language. HTML was primarily designed as a language for semantically describing scientific documents, although its general design and adaptations over the years have enabled it to be also suitable for describing a number of other types of documents, particularly, it has become the main mark-up language for creating web pages. However the HTML5 web appearance is improved in this work using CSS3[19], which is the last version of CSS -a style sheet language used for describing the look and formatting of mark-up documents-. CSS3 allows to control the document view, such as colors, sizes, layouts and visual effects, among others. On the other hand, the interaction between the user and the web interface is captured by the event handlers of jQuery[20], which is a JavaScript library. jQuery provides dynamic manipulation of HTML documents, event handling, animation, and a simpler Ajax[21] use; furthermore, it works across a multitude of browsers since it is versatile and extensible.

The web interface transmits a request to the server layer via HTTP, which is an application-level protocol for distributed, collaborative, hypermedia information systems and the foundation of data communication for the World Wide Web. Thus, HTTP can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through an extension of its request methods, error codes or headers. Since HTTP allows for the typing and negotiation of data representation, the systems can be built independently of the data being transferred.

As a last point, it is worth mentioning that most of the views are actually generated in the server, using the view module Twig of Symfony2, which eases dynamism on the interface since it provides template inheritance, as section 2.2 describes.

2.3.2 Server layer

Many of the services provided by the system are managed and executed on the server layer, which is composed of several elements with different roles. As Fig. 2.2 shows, the web server handles web interface requests (via HTTP) and manages the system resources (i.e., the storage of meta-data and files), in addition of handling algorithm executions. The server layer can be considered as the main engine of the system since it is in charge of managing and connecting the different components of our system.

The server layer is also in charge of storing image meta-data, following a database scheme that is described in the next subsection. In our case, MySQL has been selected as relational database manager, since it is open-source and provides fast queries and low computational cost. This is a popular choice of database for use in web applications, and has the advantage that many programming languages (such as C/C++, used for the development of our system) already include libraries for accessing MySQL databases.

On the other hand, a file storage server is also included in this layer for providing remote file access to any of the layers of the system. This module is also in charge of uploading and downloading data files, such as images and meta-data, via FTP[22], a standard network protocol used to transfer files from one host to another such as, in our case, the internet. Furthermore, since remote file storage allows to include distributed computing resources, the file server provides image data to the processing layer, as described in subsection 2.3.3.

The communication between the web interface and the web server is handled by an application server. Apache2[23] server has been selected for this role, which has become the most popular HTTP server in the

[18] http://www.w3.org/TR/html5
[19] http://www.w3.org/Style/CSS3
[20] http://jquery.com
[21] http://www.w3.org/standards/webdesign/script
[22] http://www.w3.org/Protocols/rfc959/
[23] http://www.apache.org/

last 20 years; further, it was the first web server software to serve more than 100 million websites according to the Netcraft[24] analysis. Apache2, distributed as free software, provides a full range of web server features, including CGI[25], SSH[26], authentication modules, and virtual domains; it also supports plug-in modules for extensibility. Apache2 supports several sever-side programming languages, such as Perl, Python, Tcl or PHP.

In this way, PHP 5.3 is the main programming language used in our system, which is a server-side scripting language designed for web development, but also used as a general-purpose programming language; it is widely used in web development and is now installed on more than millions of websites and web servers. PHP 5.3 is a complete programming language -object-oriented-, that also offers all the features needed to build a complete web server, such as user sessions, parsers for standard data-interchange format, and a full-stack of plug-ins for database managing and transport protocols. As section 2.2 shows, Symfony2 framework is written entirely in PHP 5.3 whose storage layer is defined to access the database in an object-oriented way. Ultimately, in this subsection we show the communication between the server layer and the processing layer (described in subsection 2.3.3). Nowadays, most distributed computing resources are protected by a strong security access; fortunately, the majority can be accessed by the SSH protocol, which is a protocol for secure remote login and other secure network services over an insecure network. Thus, we have implemented a PHP class for remote access to the processing layer, which uses the SSH PHP library. Specially, the communication between the server and the processing layer is carried out by secure remote access using the SSH protocol, and in addition the files produced by the processing layer are retrieved by SCP[27] (which is SSH-based).

2.3.3 Processing layer

The processing layer has been designed to relieve the web server workload of the algorithm executions with high computational cost. High availability and quality of service are provided by this layer, since several algorithms can be executed at the same time without slowing down of the system or introducing time penalizations. The processing layer is in charge of executing algorithms, mostly related to the cataloguing of new hyperspectral images in the system.

The processing layer algorithms are implemented in C/C++ using libraries to access parallel computing facilities, which efficiently execute requests coming from the web server; the complete algorithm implementations are detailed in the next chapter. This layer receives execution requests from the server layer, via secure shell (SSH) protocol, along with execution parameters and FTP links of the required files to execute algorithms. Those links could be the image source and results of previous executions (required for algorithms in which the input is the result of other algorithms).

In order to manage algorithm executions in the processing layer, we have implemented the *execution manager* as a shell script. Shell is a command line language designed for Unix command line interpreters, so that a script is a collection of commands which work serially to resolve a problem. In addition, the *execution manager* uses a special sub-script implemented for monitoring algorithm executions -called *monitoring module*-, thus the only requirement to integrate different *resource managers* (described in subsection 2.3.4.4) is to implement a specific *monitoring module* for them. Below we detail the work-flow of this *execution manager*:

- Creating temporal folder structure which stores the temporal data from the execution.

[24] http://www.netcraft.com/
[25] http://www.w3.org/CGI/
[26] http://www.ietf.org/rfc/rfc4251.txt
[27] http://linux.die.net/man/1/scp

a) CETA-Ciemat Cluster b) Timisoara Cluster

Figure 2.3: Appareance of the facilities of the two clusters supported in the system.

- Checking input data and downloading the files from the storage server.
- Reading configuration of the selected cluster.
- Submitting the execution algorithm request to the *resource manager*.
- Monitoring the execution and retrieving the results.
- Encoding the results as `JSON` format and sending the results back to the server layer.
- Deleting the temporal folder structure.

Currently, our processing layer supports efficient algorithm executions using several clusters with different *resource managers* located in different organizations in the European area, which are provided by two institutions from Spain and Romania. Indeed, any cluster can be supported by our system since a specific *monitoring module* is implemented to control the resource manager of such cluster. The currently supported clusters are described in the next subsection.

2.3.4 Clusters supported in the system

The system has been designed to support access to any cluster which allows `SSH` access; indeed, it allows algorithm execution over clusters with different architectures and resource management systems. At the moment the system has access to several clusters (Fig. 2.3 shows a general view of the cluster facilities) located in different European countries, particularly, CETA-Ciemat[28] from Spain and West University of Timisoara[29] from Romania. Next, we introduce those institutions and provide some features of the available computational resources.

2.3.4.1 CETA-Ciemat resources

The Extremadura Research Centre for Advanced Technologies (CETA-Ciemat), which belongs to the Spanish Ministry of Science and Innovation, participates in our work with two GPU computing platforms. Both systems are managed by `SLURM`, which is briefly described in subsection 2.3.4.4. In the following, we describe the features of these two clusters:

[28] http://www.ceta-ciemat.es
[29] hpc.uvt.ro

1. The *CETA Production* cluster has 32 computing nodes with 2 *NVidia*™ TESLA M2050[30] GPUs, 64 GPUS in total, each GPU features 448 streaming processor cores at 1.15 GHz, with single precision floating point performance of 1.03 TFlops, double precision floating point performance of 515 GFlops, total dedicated memory of 3 GB at 1.5 GHz and memory bandwidth of 144 GB/sec. The GPUs are connected to eight nodes with: two multi-cores of type Quad Core Intel Xeon at 2.93 GHz with 4 physical cores, and 24 GB of DDR3 1333 MHz SRAM memory. The system is mounted on a Bullx R422[31].

2. The *CETA Test* GPU cluster has 34 GPUs split in 17 computing nodes with 2 *NVidia*™ TESLA C1060[32] GPUs; each GPU features 240 streaming processor cores at 1.3 GHz, total memory dedicated of 4 GB at 800 MHz and memory bandwidth of 102 GB/s. The GPUs are connected to eight nodes with: two multi-cores of type Quad Core Intel Xeon at 2.26 GHz with 4 physical cores, and 24 GB of DDR3 1333 MHz SRAM memory. The system is mounted on a Bullx R422[33].

2.3.4.2 West University of Timisoara resources

The second computing resource provider is the HPC center at West University of Timisoara, Romania. It participates in our work with a GPU platform, which is managed by `LoadLeveler` resource manager (described in subsection 2.3.4.4). The INFRAGRID GPU cluster has 7 computing nodes with a *NVidia*™ TESLA M2070Q[34] GPU; each GPU features 448 streaming processor cores at 1.15 GHz, with single precision floating point performance of 1.03 TFlops, double precision floating point performance of 515 GFlops, total dedicated memory of 6 GB at 1.55 GHz and memory bandwidth of 148 GB/sec. Every GPU is installed in a system with two multi-cores of type Quad Core Intel Xeon at 3.46 GHz with 4 physical cores, and 32 GB of DDR3 1333 MHz SRAM memory.

2.3.4.3 GPU cluster architectures

As we have mentioned above, the current version of the system supports two different GPU cluster architectures. Table 2.1 describes the features of both GPU clusters, furthermore Table 2.2 compares the computational differences between the GPU devices installed in those clusters.

2.3.4.4 Resource managers

In this section, we describe the resource managers that have been used for the implementation of our system.

1. `SLURM`[35] is an open-source resource manager designed for Linux clusters of all sizes. It provides three key functions. First, it allocates exclusive and/or non-exclusive access to resources (compute nodes) to users so they can perform works in parallel. Second, it provides a framework for starting, executing, and monitoring works (typically, parallel jobs) on a set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending works. `SLURM` provides high availability and quality of service to our system, since it offers the following features:

[30] http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_M2050_M2070_Apr10_LowRes.pdf
[31] http://www.bull.com/catalogue/details.asp?tmp=bxs-rack-fr&opt=ns-r424e02&dt=ft&cat=bullx
[32] http://www.nvidia.co.uk/object/tesla_c1060_uk.html
[33] http://www.bull.com/catalogue/details.asp?tmp=bxs-rack-fr&opt=ns-r422e02&dt=ft&cat=bullx
[34] http://www.nvidia.com/docs/IO/40049/Tesla_M2070Q_Datasheet.pdf
[35] https://computing.llnl.gov/linux/slurm

Table 2.1: Technical specifications of the GPU clusters used in this study.

Specifications	Cluster		
	INFRAGRID	CETA Test	CETA Production
Scheduler	Loadleveler	SLURM	
Number of GPU devices	7	34	64
Device version	M2070Q	C1060	M2050
Number of streaming processor	444	240	444
Processor clock	1024 MHz	1293 MHz	1024 MHz
Single precision floating point	1.03 TFlops	933 GFlops	1.03 TFlops
Double precision floating point	515 TFlops	78 GFlops	515 TFlops
GPU memory size	6 GB	4 GB	3 GB
GPU memory clock	1546 MHz	800 MHz	1546 MHz
GPU memory bandwidth	148 GB/s	102 GB/s	148 GB/s
Multi-core processor	Quad Core Intel Xeon		
Processor clock	3.46 GHz	2.26 GHz	2.93 GHz
RAM memory size	DDR3 32 GB	DDR3 24 GB	
RAM memory clock	1333 MHz		

Table 2.2: Computational power of our Tesla devices.

Specifications	Version	
	C1060	M2050/M2070Q
Maximum size of z dimension of a grid	65535	
Maximum number of threads per block	512	1024
Maximum size of x and y dimensions of a block	512	1024
Maximum size of z dimension of a block	64	
Warp size	32	
Maximum number of resident threads per MP	768	1536
Maximum number of resident blocks per MP	8	
Maximum number of resident warps per MP	24	48
Number of 32-bit registers per MP	8 K	32 K
Maximum shared memory per MP	16 KB	48 KB
Number of shared memory banks	16	32
Amount of local memory per thread	16 KB	512 KB
Constant memory size	8 KB	

- Scalability: it is designed to operate in a heterogeneous cluster with up to tens of millions of processors.
- Performance: it can accept more than 1000 job submissions per second and fully execute 500 simple jobs per second (depending upon hardware and system configuration).
- Free and open-source: its source code is freely available under the GNU General Public License.
- Fault tolerant: it is highly tolerant of system failures, including failure of the node executing its control functions.
- Status jobs: it provides the status of the running jobs at the level of individual tasks to help identify load imbalances and other anomalies.

`SLURM` provides workload management on many of the most powerful computers in the world. On

the November 2013 Top500[36] list, five of the ten top systems use SLURM including the number one system, which sum over 5.7 million cores. Furthermore, SLURM manages several GPU clusters which are presented in the list, four of them are listed below:

- At the 6th place, **Piz Daint** a Cray XC30 system at the Swiss National Supercomputing Centre with 28 racks and 5272 hybrid compute nodes each with an Intel Xeon E52670 processor and a *NVidia*™ Kepler K20x[37] GPU device, which means 115,984 compute cores, yielding a peak performance of 6.27 PFlops.
- At the 7th place, **Stampede** at the Texas Advanced Computing Center/University of Texas is a Dell with over 80,000 Intel Xeon processors, Intel XeonPhi[38] co-processors, plus 128 *NVidia*™ Kepler K20x GPUs, delivering 5.17 PFlops.
- At the 37th place, **Lomonosov**, a T-Platforms system at Moscow State University Research Computing Center with 52,168 Intel Xeon processors and 8,840 *NVidia*™ Tesla X2070[39] GPU devices, yielding a peak performance of 1.7 PFlops.
- At the 117th place, **LOEWE-CSC** is a combination of CPU-GPU Linux cluster at the Center for Scientific Computing (CSC) of the Goethe University Frankfurt, Germany, with 20,928 AMD Magny-Cours[40] CPU cores plus 778 ATI Radeon 5870[41] GPU devices, delivering 508.499 TFlop/s.

2. LoadLeveler[42] is a commercial workload management and job scheduling system developed by IBM. It is based on the CONDOR[43] system to which many new features were added, making it much more versatile. Current versions of LoadLeveler support load balancing across the workstations in the cluster, multiple resource queues, serial and parallel workloads, and distributed file systems. LoadLeveler also provides a well documented application programming interface for customizations and extensions to the system. Some of its most important features are listed below:

   - Provides facilities for building, submitting and processing serial and parallel batch jobs.
   - Allows to match job requirements with the best available machine resources.
   - Is flexible because each machine (node) can be configured differently if required.

   LoadLeveler provides workload management on many of the most powerful computers in the world. On the November 2013 Top500 list, four of the ten top systems use LoadLeveler including the number three system. Furthermore, it manages several GPU clusters where are presented in the list, three of them are listed below:

   - At the 31st place, a customized IBM iDataPlex DX360M4[44] system at the the Rechenzentrum Garching computing center in Germany. It combines 12 nodes with Intel

[36] http://top500.org
[37] http://www.nvidia.com/content/PDF/kepler/TeslaK20XBD06397001v05.pdf
[38] http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html
[39] http://www.nvidia.com/object/tesla_tech_specs.html
[40] http://www.amd.com/Documents/Magny-coursPerformanceRHEL_BhavnaSarathy.pdf
[41] http://www.amd.com/es/products/desktop/graphics/ati-radeon-hd-5000/hd-5870/Pages/ati-radeon-hd-5870-overview.aspx
[42] http://www-03.ibm.com/systems/software/loadleveler
[43] http://toolkit.globus.org/grid_software/computation/condor.php
[44] http://www03.ibm.com/systems/x/hardware/highdensity/dx360m4/

Xeon Phi co-processors and 338 nodes with two *NVidia*™ Kepler K20x GPU devices, delivering 1.03 PFlops.

- At the 328th place, a customized IBM iDataPlex DX360M4 cluster at private financial institution in United States. It integrates Intel Xeon E5-2680v2 with *NVidia*™ Kepler K20x, yielding a peak performance of 146.2 TFlops.
- At the 336th place, a customized IBM iDataPlex DX360M4 cluster at Arizona University in United States. It integrates Intel Xeon E5-2680v2 with *NVidia*™ Kepler K20x, yielding a peak performance of 144.9 TFlops.

2.4 Database structure

The structure of the database used to store hyperspectral meta-data is illustrated in Fig. 2.4. The database has been carefully designed in order to store relevant information about the hyperspectral images which are stored in our system. In addition to standard information about each scene, such as the number of samples, lines, bands, data type, byte-order, wavelength information or interleave, we also store additional information such as the endmembers and abundances associated to each scene (content meta-data), as well as additional (optional) information such as the results and publications in which a certain hyperspectral scene has been addressed, or the results obtained from the scene by different algorithms. The meta-data related to the image content are automatically generated by the system using unmixing algorithms that are used to catalog each scene, hence the procedure for uploading a new data set to the system only requires basic information about the scene. Furthermore, a RGB quick-look is generated automatically, as Algorithm 1 shows, by performing the three spectral bands red, green and blue. In addition, relevant information about previous analyses and experiments over each scene can also be stored in the database if available, while the unmixing results are used to generate meta-data that can be then used for retrieval purposes using different queries.

Algorithm 1 RGB Encoder

```
1: procedure RGBEncoder(image,RGBFile)
2:     interval = (image.WaveLast−image.WaveFirst)/image.Bands        ▷ Wavelength interval of the image bands
3:     waveRed = 650, waveGreen = 550, waveBlue = 480                 ▷ Standard RGB wavelengths
4:     colors[0] = (waveRed−image.waveFirst)/interval                 ▷ Band according to red wavelength
5:     colors[1] = (waveGreen−image.waveFirst)/interval               ▷ Band according to green wavelength
6:     colors[2] = (waveBlue - waveFirst)/interval                    ▷ Band according to blue wavelength
7:     for j = 0 to 3 do
8:         for i=0 to image.Pixels do
9:             RGBVector[i+(j∗image.Pixels)] = AbsoluteValue(image.Vector[i+(colors[j]∗image.Pixels)])
10:        end for
11:    end for
12:    Paint RGBVector in RGBFile                                     ▷ using any encoder library
13: end procedure
```

The database scheme tables could be classified, according to the above statement, as follows. First the tables which contain features about the image, second the tables related to data execution, and last (but not least) the system authentication/authorization data.

2.4.1 Image features

This subsection describes the features of each of the tables which are related to the image features.

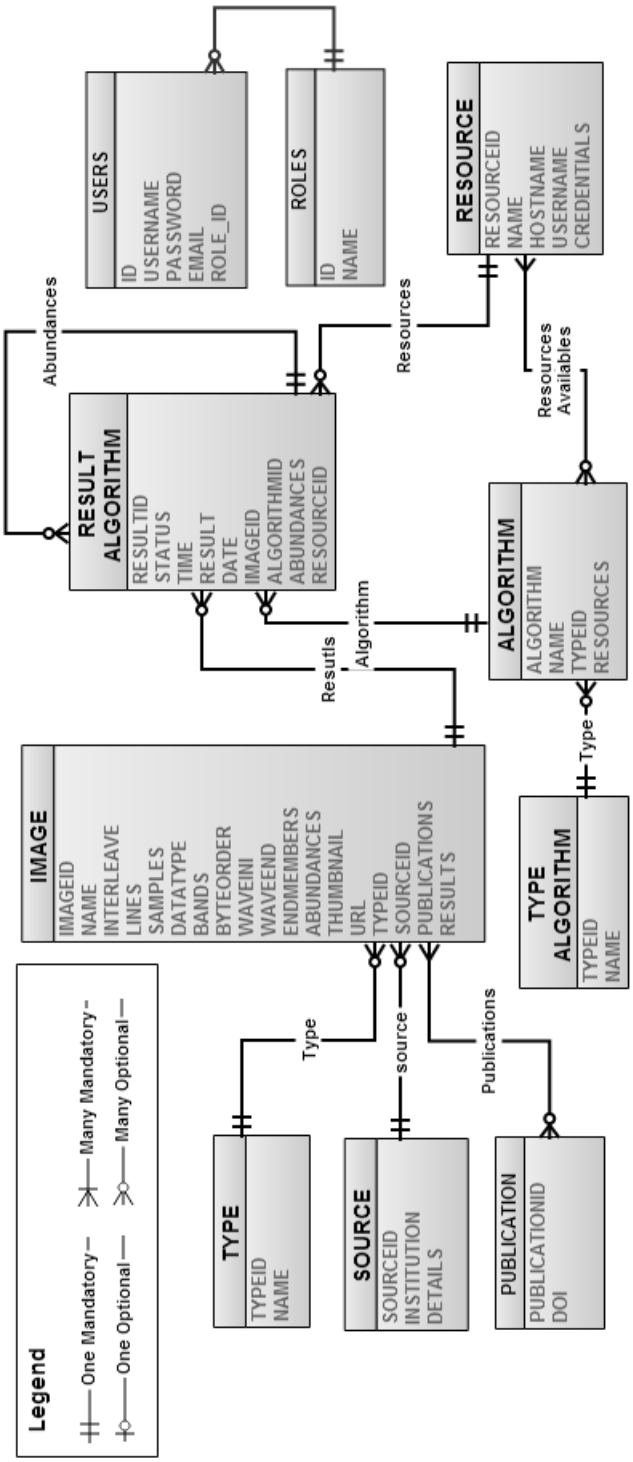


Figure 2.4: Structure of the database used to store hyperspectral data in our system.

2.4.1.1 Image

The image table is the main table of the database, since its attributes have been designed to describe the relevant features for hyperspectral image analysis; furthermore it includes additional information through related tables such as image source, involved publications or type image field. On the other hand, the image table contains the endmembers and abundances which have been selected from the result of algorithm executions on the given image. In the following, the table attributes are listed:

- **Id**. The identifier of the image.
- **Name**. The name assigned by the image owner.
- **Interleave**. Interleave feature of the image, following the `ENVI`[45] standard notation. This attribute refers to the order in which the pixels are stored: band sequential (BSQ), band interleaved by pixel (BIP), or band interleaved by line (BIL).
- **Lines**. Lines feature of the image, following the `ENVI` standard notation. It refers to the the number of lines per image for each band.
- **Samples**. Samples feature of the image, following the `ENVI` standard notation. It refers to the the number of samples (pixels) per image line for each band.
- **Bands**. Bands feature of the image, following the `ENVI` standard notation. It refers to the the number of bands per image file.
- **DataType**. Data type feature of the image, following the `ENVI` standard notation. It is a parameter identifying the type of data representation, where 1=8 bits (byte); 2 = 16 bits signed integer; 3 = 32 bits signed long integer; 4 = 32 bits floating point; 5 = 64 bits double precision floating point; 6 = 2×32 bits complex, real-imaginary pair of double precision; 9 = 2×64 bits double precision complex, real-imaginary pair of double precision; 12 = 16 bits unsigned integer; 13 = 32 bits unsigned long integer; 14 = 64 bits signed long integer; and 15 = 64 bits unsigned long integer.
- **ByteOrder**. A byte order feature of the image, following the `ENVI` standard notation. It describes the order of the bytes in an integer, long integer, 64 bits integer, unsigned 64 bits integer, floating point, double precision or complex data types. The byte order value 0 is Least Significant Byte First (LSF) data (DEC and MS-DOS systems) and byte order value 1 is Most Significant Byte First (MSF) data (all others - SUN, SGI, IBM, HP, DG).
- **Wavelenght**. Lists the centre wavelength values of each band in an image.
- **WavelenghtUnit**. Wavelength values metric.
- **WaveIni**. Indicates the minimum value of the wavelength sensitive range; it depends on the hyperspectral sensor.
- **WaveEnd**. Indicates the maximum value of the wavelength sensitive range; it depends on the hyperspectral sensor.

[45] https://www.exelisvis.com/ProductsServices/ENVIProducts/ENVI.aspx

- **Endmembers**. A special attribute which contains the extracted endmembers from the image; it is automatically assigned after selecting a set of endmembers in the cataloguing process.
- **Abundances**. A special attribute which contains the estimated abundances; it is automatically assigned after selecting a determinate abundance result in the cataloguing process.
- **Thumbnail**. A special attribute which contains a small image quick-look. The RGB image is obtained from Algorithm 1, and then it is encoded on characters and is automatically assigned in the process of uploading image.
- **URL**. A special attribute which contains the image location, in file storage server; it is automatically assigned by the server in the process of uploading image.
- **TypeID**. Foreign key to *Image Type* table.
- **SourceID**. Foreign key to *Source* table.
- **PublicationSet**. Foreign key to *Publication* table. A set of publications in which the image is involved in.
- **ResultSet**. Foreign key to *Result Algorithm* table. A set of results from algorithms executions which have been performed over the image.

2.4.1.2 Image type

Images are classified by fields types as such as hyperspectral satellite images or hyperspectral medical images, among others. In the following, the table attributes are listed:

- **Id**. The identifier of the type.
- **Name**. Descriptive name of the type.

2.4.1.3 Source

It allows cataloguing an image based on its source, such as universities, institutes, national research centers or any institution which has provided such image. In the following, the table attributes are listed:

- **Id**. The identifier of the source.
- **Institution**. Name of the institution.
- **Details**. An additional attribute for commenting.

2.4.1.4 Publication

Publications in which an image is involved, in order to understand the importance of an image and contrast the results of different publications. In the following, the table attributes are listed:

- **Id**. The publication table identifier.
- **DOI**. The identifier of the publication in the research scope, it is the standard `DOI` (digital object identifier).

2.4.2 Data execution

This subsection describes the attributes of the tables related to algorithms execution in our system.

2.4.2.1 Algorithm

The system has been designed to provide an image catalog, which is performed by several algorithms over different compute resources. Thus, different algorithms and computation resources can been selected in each cataloguing execution. In the following, we list the table attributes involved:

- **Id**. The algorithm table identifier.
- **Name**. Descriptive name of the algorithm.
- **TypeID**. Foreign key to *Algorithm Type* table.
- **Resource**. Foreign key to *Resource* table. Set of available computational resources to execute the algorithm.

2.4.2.2 Algorithm type

Algorithms are classified into different types according to their role in the cataloguing process, such as estimation of the number of endmembers, endmember extraction or abundance estimation. In the following, the attributes of the involved table are listed:

- **Id**. The algorithm type table identifier.
- **Name**. Descriptive name of the algorithm type.

2.4.2.3 Result algorithm

The system provides execution monitoring, such that it stores the status of each execution at any time. Therefore, users can check the status of the executions (successful or wrong) and also download the final results. In the following, the table attributes are listed:

- **Id**. The result algorithm table identifier.
- **Status**. Status of the algorithm execution.
- **Time**. Time spent by the algorithm execution.
- **Date**. Date of the algorithm execution.
- **Result**. Result of the algorithm execution. The results could be the number of endmembers or the URL of a file which provides an address to the file results of the endmember extraction or abundance estimation process.
- **Abundances**. Foreign key to *Result Algorithm* table. It links the abundances extracted from a given set of endmembers previously extracted.
- **ImageID**. Foreign key to *Image* table.
- **AlgorithmID**. Foreign key to *Algorithm* table.
- **ResourceID**. Foreign key to *Resource* table.

2.4.2.4 Resource

The algorithm executions are performed in distributed computing resources, so that the system needs access data to connect, execute, monitor and retrieve the execution results to/from remote resources. This table contains the relevant information to authenticate and authorize our system in the distributed computing resources. In the following, the table attributes are listed:

- **Id**. The identifier of the table.
- **Name**. Descriptive name of the resource.
- **HostName**. Host name of the cluster.
- **UserName**. User name with right permissions.
- **Credentials**. User credentials.

2.4.3 Access features

Database tables related to the system authentication/authorization data are described in this subsection.

2.4.3.1 Users

User authentication/authorization data is stored in this table. In the following, the table attributes are listed:

- **Id**. The user table identifier.
- **Name**. User name.
- **Password**. User authentication password.
- **Email**. Email user address.
- **Role**. Foreign key to *Roles* table.

2.4.3.2 Roles

The system authorization is based on roles which apply access restrictions to some services for users according to their role. In the following, the table attributes are listed:

- **Id**. The table identifier.
- **Name**. Descriptive name of the role.

2.5 CBIR interface use

The proposed CBIR system performs a complex process to generate the searching meta-data and retrieve the image, from image analysis in distributed computing resources to matching the spectral/spatial features of these images with an input ground-truth. Fortunately, this process is totally transparent to the user through the web interface. The web interface provides hyperspectral data management, such as images and their associated meta-data, in addition to automatic loading of a spectral library and using the spectra in such library in order to perform CBIR. The user may select a few spectra from the library

or even the full library, using the selected spectral signatures as input to a query. In subsection 2.5.1 we describe the hyperspectral data set uploading process. Subsection 2.5.2 concludes with a description of how the queries are carried out.

2.5.1 Hyperspectral images uploading

All the registered users have permission to add new images to our system; further the users are allowed to modify any data from their images. The process of adding new images follows two steps, first the meta-data of the new image are introduced into the system through a web form as we can see in Fig. 2.5(a); these data are required to perform any algorithm over the image. Second, as Fig. 2.5(b) shows, the binary file, which contains the reflectance data of the image, is uploaded into the system. In this stage, the system generates a compressed file that will be available for downloading. This file includes the binary data and a header file (ENVI format) within the associated meta-data previously added in the first step. Optionally, the ground-truth related to the image can be included in the image data [as Fig. 2.5(c) shows]; such ground-truth is very useful for further analyses.

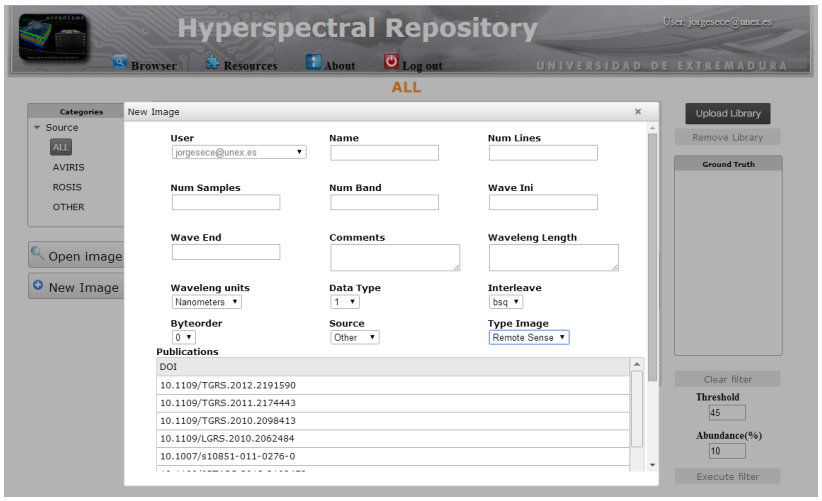
2.5.2 Queries

Our CBIR system allows end-users to perform queries to the hyperspectral image database described in Fig. 2.4. For each new hyperspectral data set, the spectral endmembers and their corresponding abundance maps can be obtained using a set of algorithms implemented in the system. These algorithms, which are described in the next chapters, conform a full spectral unmixing chain made up of three steps: 1) estimation of the number of endmembers, 2) identification of endmember signatures, and 3) abundance estimation. The information provided by endmember identification and abundance estimation is then used as meta-data, to catalog each image stored in the database. This allows for fast CBIR functionality, since the meta-data provides a compact representation of each scene in the database, and the full unmixing chain is implemented efficiently in parallel (exploiting also the GPUs available in the system).

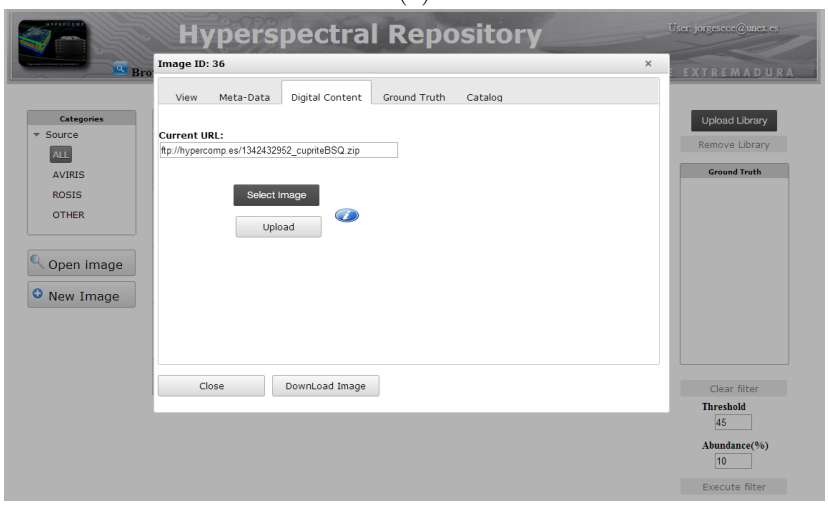
In the following, we provide a simple step-by-step example illustrating how to perform a simple hyperspectral image retrieval task in our system. Fig. 2.6(a) shows a general overview of our system, which allows guest access (this option does not allow uploading new images in the system). Full access to the system is also available upon request, including the image uploading functionality. Although the system is still in beta version, it is fully operational (available online from http://hypercomp.es/repository) and allows any interested user to obtain an account.

Fig. 2.6(b) shows the catalog panel of the system, which allows for automatically extracting meta-data for each new hyperspectral scene that is uploaded in the system. The user can decide between two algorithms for estimating the number of endmembers (VD [75] and HySime [51]), two algorithms for identifying endmember signatures (N-FINDR [57] and OSP-GS [60]), and two algorithms for estimating the abundances (UCLS [70] and ISRA [52]). The user can also decide in which computing resource the algorithms will be performed. As subsection 2.3.4 shows, the algorithms can be executed in both GPUs or multi-core clusters. After the algorithms have been executed, the results are visualized and the user can decide the best combination of algorithms in order to catalog the hyperspectral scene. Once the procedure is completed, the hyperspectral image will be automatically catalogued using the resulting meta-data, and stored in the database structure described in Fig. 2.4.

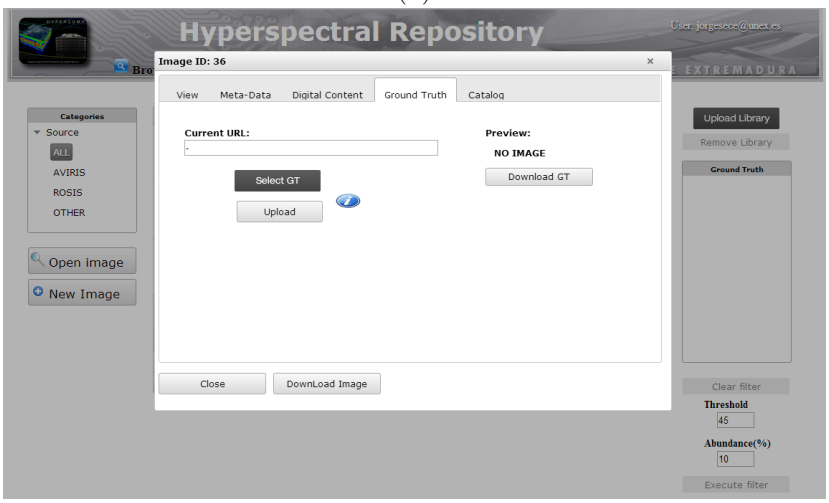
Fig. 2.6(c) shows an example of a query, in which a `USGS` library of minerals is loaded in our system and two specific spectral signatures (muscovite and kaolinite) are used to define the query. The spectral



(a)



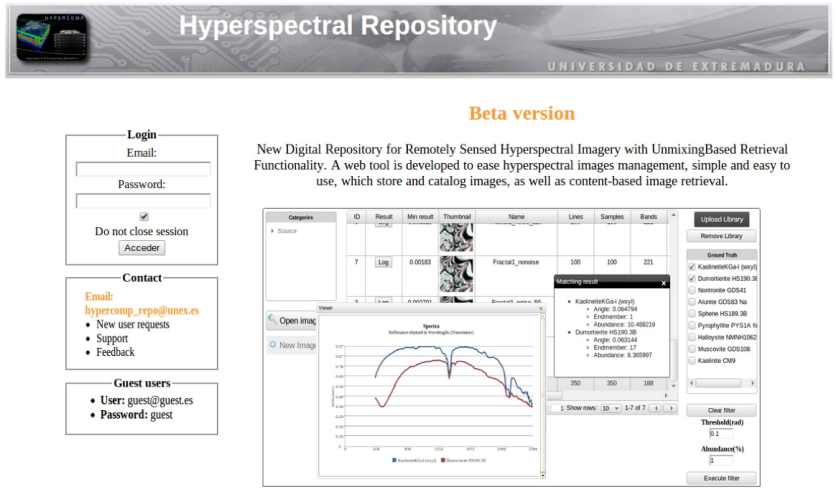
(b)



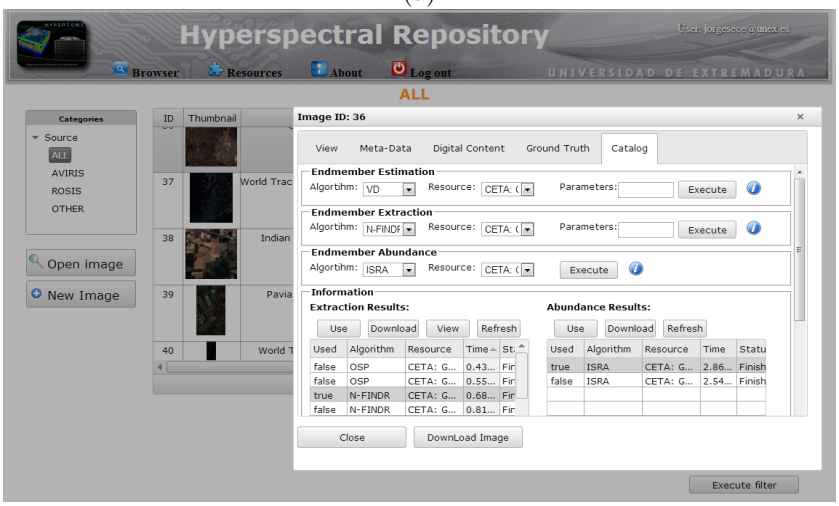
(c)

Figure 2.5: Image uploading work-flow: (a) Introducing the new image meta-data. (b) Uploading a new image binary file. (c) Uploading associated ground-truth file.

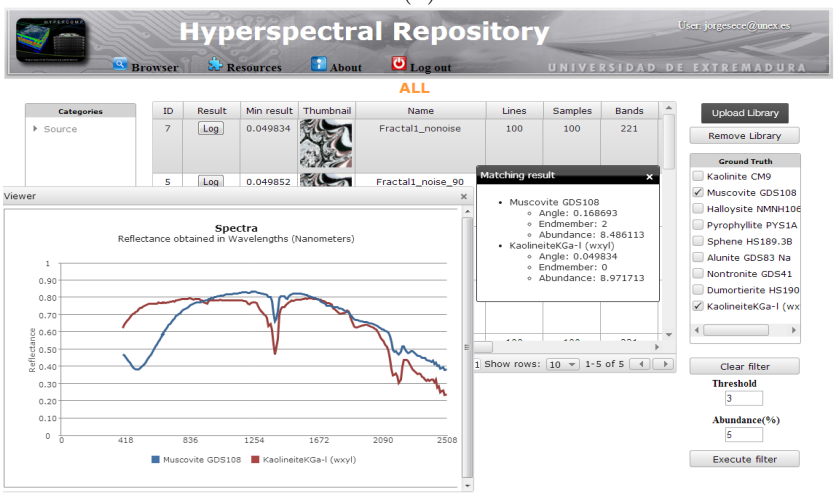
similarity threshold is set to 3 degrees and the minimum abundance is set to 5% (this means that we are looking for hyperspectral scenes containing at least 5% muscovite and 5% kaolinite). The system now provides information the images retrieved. For the first one in the list, the system estimates 8.48% of muscovite and 8.97% of kaolinite. In this case, the spectral similarity scores are very high, with less than one degree in the spectral similarity test for both endmembers. As a result, the end-user can infer that this scene accurately satisfies the search criterion. Since there are other scenes retrieved, the end-user may decide to select any other hyperspectral image retrieved from the query (the images are ordered according to the combined spectral similarity score resulting from the query).



(a)



(b)



(c)

Figure 2.6: (a) General overview of the system. (b) Catalog panel of the system. (c) Example of a query.

Chapter 3

Unmixing-based image retrieval system

A CBIR system is typically intended to retrieve images by means of the information that is contained in the data (images). This is particularly important in large data repositories, such as those available in remotely sensed hyperspectral imaging [65]. For this reason, the development of CBIR systems for hyperspectral data repositories is an emerging need in remote sensing applications, particularly after the development of several new Earth observation missions [15]. In this chapter, we describe the CBIR strategy [76, 77] that we have developed for our system, which is based on an unmixing-based image retrieval methodology.

The remainder of the chapter is structured as follows. Section 3.1 describes developments in the area of CBIR applied to hyperspectral image databases. Section 3.2 describes the CBIR behaviour with particular emphasis on the proposed searching methodology. Section 3.3 describes the unmixing chain algorithms that are used to extract the spectral information. Section 3.4 develops the efficient implementation on GPUs of the unmixing chain algorithms. Section 3.5 describes the comparative metrics used in this work for both algorithm accuracy evaluation and image retrieval. Section 3.6 concludes with the performance of the system by comparing its retrieval accuracy, using real hyperspectral and synthetic images with different noise levels.

3.1 Previous CBIR developments fo hyperspectral imaging

There have been several attempts towards the development of CBIR systems in the area of hyperspectral imaging. One of the most relevant ones was described in [42], which presents a spectral/spatial CBIR system for hyperspectral images. The authors use endmember induction algorithms to extract a set of image spectral features, and then compute spatial features as abundance image statistics. These two sources of information are then combined into a dissimilarity measure that guides the search for answers to database queries [81]. In this context, each hyperspectral image is characterized by a tuple given by the set of induced endmembers and the set of fractional abundance maps resulting from an unmixing process conducted using three stages [82]. For the estimation of the number of endmembers, the authors use the VD [75] method. For the endmember induction step, the authors use three different methods: 1) a fast implementation of the PPI algorithm [63], called fast iterative PPI (FIPPI) [83]; 2) the volume-based N-FINDR method [57]; and 3) the ILSIA algorithm [84]. For the estimation of the fractional abundances of inducted endmembers, the authors use the FCLSU algorithm [36]. The authors validate

their approach using synthetic data and a real hyperspectral data set (with 2878×512 pixels and 125 spectral bands, cut in patches of 64×64 pixels for a total of 360 patches).

A similar strategy is employed in [43], which presents a parallel heterogeneous CBIR system for efficient hyperspectral image retrieval using spectral mixture analysis. Here, the PPI algorithm performs endmember extraction and FCLSU estimates the abundances. A spectral signature matching algorithm guides the queries to the database. This algorithm first considers the SAD [35] in order to retrieve images by means of an endmember-guided similarity criterion, and then the search is refined by analysing the relative difference between the abundance fractions associated with the retrieved image and the example image for the search. Another contribution of [43] is an efficient implementation of the system for heterogeneous networks of computers, possibly distributed among different locations. This idea arose from the naturally distributed format of hyperspectral image databases. The system was tested using a collection of 154 hyperspectral data sets collected by the AVIRIS sensor over the World Trade Center area in New York, only a few days after the terrorist attacks of September 11th, 2001. The implementation used a heterogeneous network of 16 workstations, and also the NASA Thunderhead cluster[1] with 256 CPUs, providing good results in terms of image retrieval accuracy and parallel performance.

3.2 Content-based image retrieval methodology

In this thesis work, we develop a completely new CBIR system which takes advantage of seminal concepts from linear spectral unmixing to perform effective data retrieval. As introduced in chapter 2, a query is defined by the user from a web interface using ground-truth spectral signatures, a similarity threshold and minimum abundance of those ground-truth substances. Then the query is sent to the CBIR engine, which returns the most similar images in the database. However, first of all every new hyperspectral data set needs to be catalogued by extracting the spectral endmembers and their corresponding abundance maps using a set of algorithms implemented in the system. These algorithms conform a full spectral unmixing chain made up of three steps: 1) estimation of the number of endmembers, 2) identification of the spectral signatures of the endmembers, and 3) estimation of fractional abundances. The information provided by endmember identification and abundance estimation is then used as meta-data, to catalog each image stored in the database. This allows for fast CBIR functionality, as the meta-data provides a compact representation of each scene in the database, and the full unmixing chain is implemented efficiently in parallel (exploiting also the GPUs available in the system).

As Fig. 3.1 shows, the CBIR workflow is composed of several stages. First and foremost, we have the *database image catalog* using the algorithms to extract the spectral information, which are detailed in section 3.3, that obtain the spectral endmembers and their abundance maps in every image. The catalog is performed once a new hyperspectral image is incorporated into the repository. This process is commanded by the image provider through the web interface. The following steps are performed in this process:

- *Estimation of the number of endmembers.* First of all, the estimation of the number of endmembers in the hyperspectral image must be performed, since it will be a relevant input parameter for the next algorithm.

- *Endmember extraction.* In a second step, the most spectrally pure spectral signatures or endmembers in the image must be located, which are relevant to a query.

[1] http://science.gsfc.nasa.gov/606.1/docs/Specs.pdf

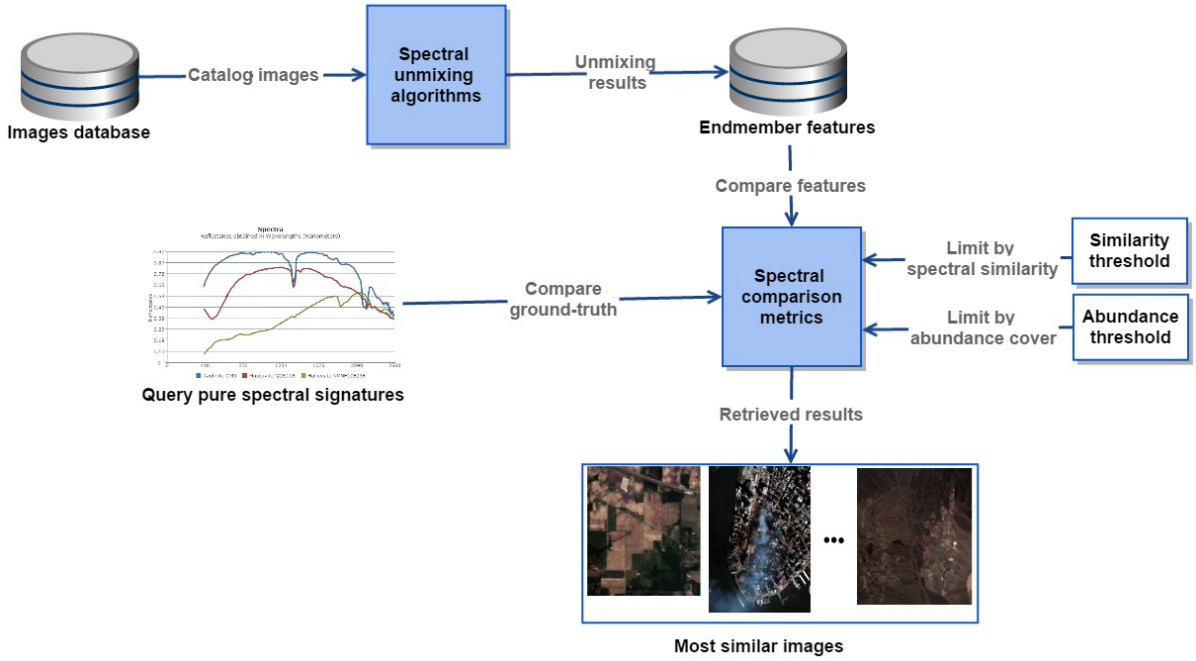


Figure 3.1: Structure of the CBIR procedure based on unmixing concepts.

- *Abundances estimation.* In the last step, the endmember abundances are estimated. This allows us to establish filters by setting a minimum required abundance of a given endmember in the scene.

The catalog process can be performed as many times as desired, until convincing results are obtained. Once the results are good enough they are assigned as associated image meta-data to the database, and they will be used for spectral comparison in future queries.

In a second stage, a query is sent to the server. A standard query defined for our CBIR is composed of spectral signatures, a threshold of similarity between the reference signatures and the image endmembers, in addition to a minimum abundance of such spectral signatures in the scene which limits the results to the images with enough quantity of such substance. The reference spectral signatures used for searching are located in spectral libraries, and our system supports standard library formats such as SLI.

The last stage of the CBIR work-flow is the image *spectral information matching*. The metrics used for this purpose are SAD [35], i.e. the angle between the spectral signatures from the spectral library and the spectral signatures extracted from the images as endmembers, and calculate the abundance percentage of such spectral signatures in the scene. This stage is fully described in subsection 3.2.2.

In the following, we briefly introduce the USGS spectral library which has been used as the main library to guide the searching process in our system. This section concludes with a description of the searching methodology, including aspects such as distance metrics and matching algorithms used for retrieval purposes.

3.2.1 U.S. Geological Survey spectral library

USGS Spectroscopy Lab[2] is studying and applying methods for identifying and mapping materials through spectroscopic remote sensing, on the Earth and throughout the solar system using laboratory, field, airborne and spacecraft spectrometers. One of its most well-known projects is the assembly of the digital reflectance spectral library, which is very useful as a reference for material identification in remotely sensed images. The newest library provided covers the wavelength range from the ultraviolet to far infrared along with sample documentation. This library comprises hundred of spectral signatures of different materials and includes samples of minerals, rocks, soils, plants, vegetation species, microorganisms, and man-made materials. In our work we use a nearlier spectral library [85], which is widely used in the literature. This library comprises the wavelength range from $0.2 - 3.0$ microns, and contains 481 spectra of minerals.

3.2.2 Searching methodology

Two searching options are available in the proposed system. The first one relies on the SAD (described in 3.5.1) in order to retrieve hyperspectral images with endmembers that are similar to those available in a spectral library that can be loaded in the system. A particular issue that may arise in this kind of search is the fact that the wavelengths of the spectral library used as input can be different from the wavelengths of the hyperspectral images stored in the system. For this purpose, we have implemented, as described in Algorithm 2, a spectral convolution strategy that looks for the wavelength values which are present in both the hyperspectral data and the input spectral library (with the possibility to include a tolerance threshold in the wavelength matching procedure). In most cases, the spectral resolution of the input signatures in the spectral library is much higher than the images stored in the database, and it is often possible to retrieve scenes with great accuracy as their associated wavelengths are a subset of those of the signatures in the spectral library.

On the other hand, the system also allows queries based on the specific abundance of a given endmember. For instance, we may not only look for scenes with a specific kind of vegetation (endmember), but also with a significant presence (abundance) of this kind of vegetation in the retrieved scene. For this purpose, we may set a minimum threshold for the abundance of a given endmember (or group of endmembers) in the retrieved scene. As Algorithm 3 shows, this is implemented by calculating the total abundance of each of the endmembers in the scene, summing all the relative contributions in each pixel and obtaining the total abundance coverage of an endmember in the scene. Then, we may impose a minimum abundance threshold that is used in the retrieval process. In this way, we can effectively perform image retrieval based on both endmember and abundance information (i.e., not only retrieving scenes that contain a certain endmember but also scenes that contain a certain amount of a given endmember). Since this information is available as meta-data for each scene, the retrieval process is quite fast and the most computationally expensive part is the generation of the meta-data itself, which is carried out in parallel as will be explained in section 3.4.

In the following, the different stages of a full searching process are detailed:

- *Initialization.* In this step, a spectral library of signatures used for the search is loaded into the system.
- *Spectral convolution.* The system automatically performs a spectral convolution strategy that allows comparing the wavelengths of the input spectral library with the wavelengths of the real

[2] http://speclab.cr.usgs.gov/

Algorithm 2 Bands convolution process

```
1: procedure BandsConvolution(grountruths,endmemders,tolerance,gtOut,endOut)
2:     lastMatched= 0
3:     counter = 0
4:     for i = 0 to grountruths.NumBands do
5:         for j = lastMatched to endmemders.NumBands do
6:             diff=grountruths.WaveLength[i] − endmemders.WaveLength[j]
7:             if diff <= tolerance then
8:                 gtMatchedArray.push(i)
9:                 endMatchedArray.push(j)
10:                lastMatched=j + 1
11:                Break For loop
12:            end if
13:        end for
14:    end for
15:    for line=0 to grountruths.size do
16:        for column=0 to gtMatchedArray.Length do
17:            k=grountruths.NumSamples ∗ line + gtMatchedArray[column];
18:            gtOut.push(grountruths.vector[k])
19:        end for
20:    end for
21:    for line=0 to endmemders.size do
22:        for column=0 to endMatchedArray.Length do
23:            k=endmembers.NumSamples ∗ line + endMatchedArray[column];
24:            endOut.push(endmemders.vector[k])
25:        end for
26:    end for
27: end procedure
```

Algorithm 3 Selection of an abundance percent threshold

```
1: procedure AbuncandesPercent(abundances,percentArray)
2:     total = 0.0
3:     for i = 0 to abundances.Pixels∗abundances.Endembers do
4:         total= AbsoluteValue(abundances.Vector[i]) + total;
5:     end for
6:     for i =0 to abundances.Endmembers do
7:         partial = 0.0
8:         overflow = abundances.Pixels∗i
9:         last = abundances.Pixels∗(i+1)
10:        for j = overflow to last−1 do
11:            partial=AbsoluteValue(abundances.Vector[j])+partial
12:        end for
13:        percentArray[i]= (partial/total)∗100
14:    end for
15: end procedure
```

hyperspectral data stored in the system.

- *Signature comparison.* For a spectral signature (or set of signatures) available in the loaded spectral library, the system calculates the SAD with all the endmembers stored as meta-data for each hyperspectral scene in the system, and retrieves a number of matching scenes satisfying the specified criterion.

- *Abundance filter.* As an optional step, the system allows defining a minimum abundance filter which is used as an additional condition to the signature comparison described in the previous step. In this case, the image is retrieved only if the matched endmember contains a total abundance in the scene that is higher than a minimum predefined abundance threshold.

- *Sorting and visualization.* The retrieved images are shown to the user sorted from higher to lower spectral similarity (lowest to highest spectral angle).

3.3 Unmixing chain

3.3.1 Estimation of the number of endmembers

The estimation of the number of endmembers is an essential stage in the unmixing process because it is highly important in later stages as a crucial parameter in endmember identification and abundance estimation algorithms. Despite its importance in the literature, there are not many automatic proposals to solve this problem. In this subsection, two proposals are described: VD [75] which is based on a detector built on the eigenvalues of the sample correlation and covariance matrices, and HySime [51] which adopts a minimum mean squared error based approach to infer the signal subspace.

3.3.1.1 Virtual Dimensionality (VD) algorithm

Let us denote by $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]$ a hyperspectral image with n pixel vectors, each with l spectral bands. First, VD calculates the eigenvalues of the covariance matrix $\mathbf{K}_{l\times l} = 1/n(\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$ and the correlation matrix $\mathbf{R}_{l\times l} = \mathbf{K}_{l\times l} + \overline{\mathbf{Y}}\overline{\mathbf{Y}}^T$, respectively referred to as covariance-eigenvalues and correlation-eigenvalues, of each of the spectral bands in the original hyperspectral image $\mathbf{Y}$. If a distinct spectral signature makes a contribution to the eigenvalue-represented signal energy in one spectral band, then its associated correlation eigenvalue will be greater than its corresponding covariance-eigenvalue, in which case only noise would be represented in this particular band. By applying this concept, a Neyman-Pearson detector [75] is introduced to formulate the issue of whether a distinct signature is present or not in each of the spectral bands of $\mathbf{Y}$ as a binary hypothesis testing problem where a so-called Neyman-Pearson detector is generated to serve as a decision marker based on a prescribed P_F (i.e. false alarm probability). In light of this interpretation, the issue of determining an appropriate estimation p for the number of endmembers is further simplified and reduced to a specific value of P_F that is present by the Neyman-Pearson detector.

3.3.1.2 Hyperspectral Signal Subspace Identification by Minimum Error (HySime)

Let us denote by $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]$ a hyperspectral image with n pixel vectors, each with l spectral bands. HySime method consists of two parts. First, an algorithm calculates the noise estimation which produces a $n \times l$ matrix $\hat{\epsilon}$ containing an estimation of the noise present in the original hyperspectral image $\mathbf{Y}$ (let us remember that n is the number of pixels and l is the number of bands of $\mathbf{Y}$). This algorithm follows an approach which addresses the high correlation exhibited by close spectral bands. This algorithm stars from calculating $\mathbf{Z} = \mathbf{Y}^T$ and $\mathbf{R}' = (\mathbf{Z}^T\mathbf{Z})^{-1}$, so the regression vector $\hat{\beta}_i$ and noise $\hat{\xi}_i$ for each band $i\epsilon\{1, 2, \cdots, l\}$ can be calculated by:

$$\hat{\beta}_i = ([\mathbf{R}']_{\kappa_i,\kappa_i} - \frac{[\mathbf{R}']_{\kappa_i,i}[\mathbf{R}']_{i,\kappa_i}}{[\mathbf{R}']_{i,i}[\hat{\mathbf{R}}]_{\kappa_i,i}}), \tag{3.1}$$

and

$$\hat{\xi} = z_i - \mathbf{Z}_{\kappa_i}\hat{\beta}_i, \tag{3.2}$$

where $[\mathbf{R}']_{\kappa_i,\kappa_i}$ denotes the matrix $\mathbf{R}'$ in which *i-th* column and *i-th* row has been deleted, $[\mathbf{R}']_{\kappa_i,i}$ denotes *i-th* column of $\mathbf{R}'$ and $[\mathbf{R}']_{\kappa_i,\kappa_i}$ denotes *i-th* row of $\mathbf{R}'$.

The main advantage of the noise estimation algorithm described above is that its computational complexity is substantially lower than other algorithms for noise estimation in hyperspectral data in the

literature. Additional details about this algorithm can be found in [51] and we do not repeat them for space considerations.

On the other hand, the second part of HySime is the signal subspace identification algorithm, which first computes the noise correlation matrix $\hat{\mathbf{R}}_n = 1/n \sum_i (\xi_i \xi_i^T)$ and then computes the signal correlation $\hat{\mathbf{R}}_s = 1/n \sum_i ((\mathbf{y}_i - \xi_i)(\mathbf{y}_i - \xi_i^T))$. The next steps are to calculate the eigenvectors of the signal correlation matrix and the terms $\hat{\delta}_i$ based on quadratic forms, and then to sort the terms in ascending order. Finally, a minimization function is applied to obtain an estimation p in the signal subspace $\hat{\mathbf{S}}$. The main purpose of this algorithm is to select the subset of eigenvectors that best represents the signal subspace in the minimum mean squared error sense, p is the number of terms $\hat{\delta}_i \ll 0$.

3.3.2 Dimensionality reduction

The number of endmembers p present in a given scene is, very often, much smaller than the number of bands l. Therefore, assuming that the linear model is a good approximation, spectral vectors lie in or very close to a low-dimensional linear subspace. The identification of this subspace enables low-dimensional yet accurate representation of spectral vectors. It is usually advantageous and sometimes necessary to operate on data represented in the signal subspace. Thus, a signal subspace identification algorithm is often required as a previous processing step in the spectral unmixing chain before the endmember identification. Although unsupervised subspace identification has been approached in many ways, we have focused on the projection techniques, which seek for the best subspaces to represent data by optimizing objective functions. For example, PCA maximizes the signal variance; SVD [69] maximizes power; MNF [53] and NAPC [55] minimize the ratio of noise power to signal power. NAPC is mathematically equivalent to MNF and can be interpreted as a sequence of two principal component transforms: the first applies to the noise and the second applies to the transformed data set. In this work PCA has been selected as dimensionality reduction method and is described below.

3.3.2.1 Principal component analysis (PCA)

Let us denote by $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]$ a hyperspectral image with n pixel vectors, each with l spectral bands. PCA is a well-known method for dimensionality reduction [62] which can be computed by performing the eigendecomposition of the covariance matrix of the hyperspectral image $\mathbf{Y}$:

$$\mathbf{K} \equiv \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}, \tag{3.3}$$

where $\mathbf{K}$ is the covariance matrix of $\mathbf{Y}$, $\mathbf{V} \equiv [\mathbf{v}_1, \mathbf{v}_2 \cdots \mathbf{v}_l]$ is an orthogonal matrix whose columns are the eigenvectors of $\mathbf{K}$, and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{K}$.

The projection of the data $\mathbf{Y}$ by the eigenvectors $\mathbf{V}$ yields the principal components of $\mathbf{Y}$. The eigenvalues in $\mathbf{\Lambda}$ encase the 'weight' of each principal component of the resulting data. By choosing only the eigenvectors corresponding to the largest p eigenvalues, the data dimensionality has reduced while preserving the maximum information (variance). This fact provides the possibility of working in a reduce data set as the information of the subspace is similar to work with original data information. In this context, for instance, the same endmembers are identified from both the reduced and the original data set.

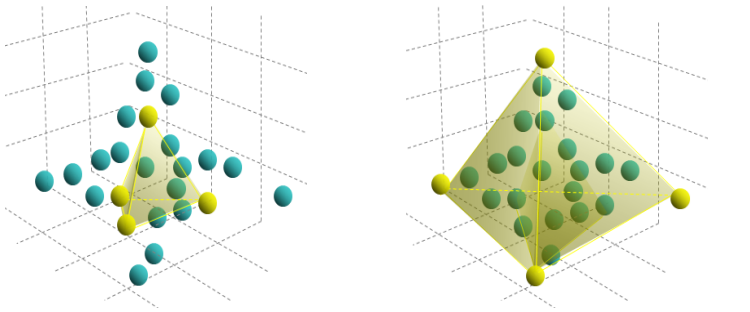
3.3.3 Endmember extraction

As introduced in chapter 1, the endmember identification process can be executed from several points of view. On the one hand, we have methods which exploit a *geometric* approximation by assuming that all the endmembers are inside a simplex. Several *geometric* methods use techniques in which do not assume the presence of pure pixels in the scene [86, 87, 88, 89]. The main problem of this approach is the possibility of obtaining *virtual endmembers*, which may be formed by spectral signatures without any relation with real materials in nature. On the other hand, *sparse* methods use spectral libraries to solve the mixture problem using spectral signatures not necessarily present in the image [40], so the resulting endmembers come from the spectral library, in which the spectral signatures were measured in a laboratory with optimal conditions. Although the endmembers obtained from the image pixels have the advantage of having same conditions as the image, it is also possible that some objects may not be perfectly represented by pure pixels in the scene. In addition, there are other approximations to the endmember identification problem such as *statistical* methods in which spatial information-oriented methods are highlighted. In our implementations, we mainly exploited endmember identification methods based on *geometric* approximations. However, our system also offers the possibility to use sparse unmixing to conduct the queries. Next, two popular *geometric* methods implemented in our system are described: N-FINDR and OSP-GS.

3.3.3.1 N-FINDR

The N-FINDR algorithm [57] is one of the most widely used an successfully applied methods for automatically determining endmembers in hyperspectral image data without using a priori information. This algorithm looks for the set of pixels with the largest possible volume by *inflating* a simplex inside the data. The procedure begins with a random initial selection of pixels (see Fig. 3.2). Every pixel in the image must be evaluated to refine the estimate of endmembers, looking for the set of pixels that maximizes the volume of the simplex defined by the selected endmembers. The mathematical definition of the volume of a simplex formed by a set of endmember candidates is proportional to the determinant of the set augmented by a row of ones. The determinant is only defined in the case where the number of features is $p-1$, p being the number of desired endmembers [90]. Since in hyperspectral data typically the number of bands is bigger than the number of endmembers, $n \gg p$, a transformation that reduces the dimensionality of the input data $\mathbf{Y}$ is required. In this work, we use the PCA for this purpose. The corresponding volume is calculated for every pixel in each endmember position by replacing that endmember and finding the resulting volume. If the replacement results in an increase of volume, the pixel replaces the endmember. This procedure is repeated in iterative fashion until there are no more endmember replacements. The method can be summarized by a step-by-step algorithmic description which is given bellow:

- *Feature reduction.* Apply the PCA to reduce the dimensionality of the data $\mathbf{Y}$ from l to $d = p - 1$, where p is the number of endmembers to be extracted.
- *Initialization.* Let $\{\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \ldots, \mathbf{m}_p^{(0)}\}$ be a set of p endmembers randomly extracted from the input data $\mathbf{Y}$.
- *Volume calculation.* At iteration $k \geq 0$, the volume defined by the current set of endmembers is



a) N-FINDR initialed randomly (p=4) b) Final volume estimation by N-FINDR

Figure 3.2: Graphical interpretation of the N-FINDR algorithm in a three-dimensional space.

calculated in volume calculation as follows.

$$\mathbf{V}(\mathbf{m}_p^{(k)}, \mathbf{m}_p^{(k)}, \ldots, \mathbf{m}_p^{(k)}) = \frac{\left| det \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \mathbf{m}_1^{(k)} & \mathbf{m}_2^{(k)} & \ldots & \mathbf{m}_p^{(k)} \end{bmatrix} \right|}{(p-1)!} \tag{3.4}$$

- *Replacement.* For each pixel vector $\mathbf{y}_i$ in the input hyperspectral data, we recalculate the volume by testing the pixel in all p endmember positions, i.e., first calculate $\mathbf{V}(\mathbf{y}_j, \mathbf{m}_2^{(k)}, \ldots, \mathbf{m}_p^{(k)})$, then calculate $\mathbf{V}(\mathbf{m}_1^{(k)}, \mathbf{y}_i, \ldots, \mathbf{m}_p^{(k)})$, and so on until $V(\mathbf{m}_1^{(k)}, \mathbf{m}_2^{(k)}, \ldots, \mathbf{y}_i)$. If none of the p recalculated volumes is greater than $\mathbf{V}(\mathbf{m}_1^{(k)}, \mathbf{m}_2^{(k)}, \ldots, \mathbf{m}_p^{(k)})$, then no endmember is replaced. Otherwise, the combination with maximum volume is updated and the endmember is replaced by the new one. So that, a new set of endmembers is produced by replacing $\mathbf{m}_j^{(k+1)} = \mathbf{y}_i$ and $\mathbf{m}_i^{(k+1)} = \mathbf{m}_i^{(k)}$ for $i \neq j$, where $\mathbf{m}_j^{(k+1)}$ is the avoided endmember. The replacement step is repeated for all the pixel vectors in the input data until all the pixels have been exhausted. The result of this is a final endmembers set of $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}{}^p$

As a final comment, it has been observed that different random initializations of N-FINDR may produce different final solutions. Since our system intends to work in heterogeneous fashion, the algorithm is not initialized with a common matrix, so that our experiments show high standard deviations in the case of N-FINDR.

3.3.3.2 Orthogonal subspace projection with Gram-Schmidt orthogonalization (OSP-GS)

The OSP algorithm [60] was originally developed to find spectrally distinct signatures using orthogonal projections. For this work, we have used an optimization of this algorithm which allows calculating the OSP without requiring the computation of the inverse of the matrix that contains the endmembers already identified in the image. This operation, which is difficult to implement in parallel, is accomplished using the Gram-Schmidt method for orthogonalization. This process selects a finite set of linearly independent vectors $\mathbf{W} = \{\mathbf{w}_1, \cdots, \mathbf{w}_p\}$ in the inner product space $\mathbf{R}^n$ in which the original hyperspectral image $\mathbf{Y}$ is defined, and generates an orthogonal set of vectors $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_p\}$ which

spans the same p-dimensional subspace of $\mathbf{R}^n$ ($p \leq n$) as $\mathbf{W}$. In particular, $\mathbf{B}$ is obtained as follows:

$$\begin{array}{ll} \mathbf{b}_1=\mathbf{w}_1, & \mathbf{m}_1 = \frac{\mathbf{b}_1}{\|\mathbf{b}_1\|} \\ \mathbf{b}_2=\mathbf{w}_2 - proj_{\,\mathbf{b}_1}(\mathbf{w}_2), & \mathbf{m}_2 = \frac{\mathbf{b}_2}{\|\mathbf{b}_2\|} \\ \mathbf{b}_3=\mathbf{w}_3 - proj_{\,\mathbf{b}_1}(\mathbf{w}_3) - proj_{\,\mathbf{b}_2}(\mathbf{w}_3), & \mathbf{m}_3 = \frac{\mathbf{b}_3}{\|\mathbf{b}_3\|} \\ \mathbf{b}_4=\mathbf{w}_4 - proj_{\,\mathbf{b}_1}(\mathbf{w}_4) - proj_{\,\mathbf{b}_2}(\mathbf{w}_4) - proj_{\,\mathbf{b}_3}(\mathbf{w}_4), & \mathbf{m}_4 = \frac{\mathbf{b}_4}{\|\mathbf{b}_4\|} \\ \vdots & \vdots \\ \mathbf{b}_p=\mathbf{w}_p - \sum_{j=1}^{p-1} proj_{\mathbf{b}_j}(\mathbf{w}_p), & \mathbf{m}_p = \frac{\mathbf{b}_p}{\|\mathbf{b}_p\|}, \end{array} \tag{3.5}$$

where the projection operator is defined in (3.6), in which $< \mathbf{w}, \mathbf{b} >$ denotes the inner product of vectors $\mathbf{w}$ and $\mathbf{b}$.

$$proj_b(\mathbf{w}) = \frac{< \mathbf{w}, \mathbf{b} >}{< \mathbf{b}, \mathbf{b} >}\mathbf{b}. \tag{3.6}$$

The sequence $\mathbf{b}_1, \cdots, \mathbf{b}_p$ in (3.5) represents the set of orthogonal vectors generated by the Gram-Schmidt method, and thus, the normalized vectors $\mathbf{m}_1, \cdots, \mathbf{m}_p$ in (3.5) form an orthonormal set. As far as $\mathbf{B}$ spans the same p-dimensional subspace of $\mathbf{R}^n$ as $\mathbf{W}$, an additional vector $\mathbf{b}_{p+1}$ computed by following the procedure stated at (3.5) is also orthogonal to all the vectors included in $\mathbf{W}$ and $\mathbf{B}$. This algebraic assertion constitutes the cornerstone of the OSP method with Gram-Schmidt orthogonalization, referred to hereinafter as OSP-GS algorithm.

3.3.4 Abundance estimation

The last stage of the unmixing chain of the linear mixture model is the estimation of abundance of the identified endmembers, $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^p$, in every pixel of the hyperspectral image $\mathbf{Y}$. The results are presented as an abundance map associated to each endmember, with as many maps as endmembers, p, being produced. Abundance values can be constrained to be non-negative (ANC), i.e., $\alpha_i \geq 0$ and the abundance sum-to-one constraint (ASC), i.e., $\sum_{i=1}^p \alpha_i = 1$. Since ASC has received some criticisms in the recent literature [40], the most popular solutions estimate the abundances without any restrictions or applying only the ANC. Next, two widely used solutions implemented in our system are presented: UCLS method, which has no restrictions, and ISRA method, which applies the ANC.

3.3.4.1 Unconstrained least-squares (UCLS)

UCLS algorithm [70] works as follows. Once the set of endmembers $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^p$ has been identified, their correspondent abundance fractions $\mathbf{A} = [\alpha_1, \alpha_2, \cdots, \alpha_p]$, in a specific l-dimensional pixel vector $\mathbf{y}_i$ of the scene, can be simply estimated (in least squares sense) by the following unconstrained expression:

$$\alpha_i = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y}_i. \tag{3.7}$$

This method is fast and easy to implement, since non-restrictions are applied. Although it can produce negative abundances which are no nature-related.

3.3.4.2 Image space reconstruction algorithm (ISRA)

ISRA algorithm [52] is an abundance estimation method which applies ANC, so all the values are positive. Once the set of endmembers $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^p$ has been identified (note that $\mathbf{M}$ can also be seen as an $n \times p$

matrix where n is the number of spectral bands and p is the number of endmembers) we apply the following steps. First, a α_i vector abundance is initialized with positive values for each $\mathbf{y}_i$ pixel in the image $\mathbf{Y}$. Then, an iterative process calculates the abundances as follows:

$$\alpha_i^{k+1} = {\alpha_i}^k \left(\frac{\mathbf{M}^T \cdot \mathbf{y}_i}{\mathbf{M}^T\mathbf{M} \cdot {\alpha_i}^k} \right), \tag{3.8}$$

this process is executed until the estimation error is lower than a given error threshold ε, such estimation error is calculated with $\|\mathbf{y}_i - \mathbf{M}{\alpha_i}^{k+1}\|_2^2$. Otherwise, the process keeps executing until:

$$\frac{\|\alpha^{k+1} - \alpha^k\|}{\|\alpha^k\|} \leq \rho, \tag{3.9}$$

where ρ indicates the maximum similarity between abundances. This algorithm has been described in [52]. It is important to emphasize that the calculations of the fractional abundances for each pixel are independent, so they can be calculated simultaneously without data dependencies, thus increasing the possibility of parallelization.

3.4 GPU implementation

In recent years, GPUs have evolved into highly parallel, multi-threaded, many-core coprocessors with tremendous computational power and memory bandwidth [91]. The combined features of general-purpose supercomputing, high parallelism, high memory bandwidth, low cost, compact size, and excellent programmability are now making GPU-based desktop computers an appealing alternative to massively parallel systems made up of commodity CPUs. The exploding GPU capability has attracted more and more scientists and engineers to use it as a cost-effective high-performance computing platform in many applications, including hyperspectral imaging problems. In addition, GPUs can also significantly increase the computational power of cluster-based and distributed systems (*e.g.*, clusters of GPUs are becoming an important architecture for supercomputing purposes.[3]).

Several efforts exploiting GPU technology can already be found in the hyperspectral unmixing literature, including [41] and references therein. Only in the area of spectral unmixing of hyperspectral data there have been many developments already. A GPU-based implementation of the AMEE algorithm for pure spectral signature identification was described in [47]. In this case, speedups on the order of 15x were reported. The well-known PPI algorithm [63] has been implemented in GPUs using different strategies [92, 93]. A GPU-based real time implementation of the VCA algorithm [73] has also been recently reported in [94]. A full spectral unmixing chain [95, 96] comprising the automatic estimation of the number of endmembers using the VD [75] or the HySime [51], the identification of the endmember signatures using the N-FINDR algorithm [57], and quantification of endmember fractional abundances using UCLS [65] has been reported in [97], with speedups superior to 50x. A variation of this chain using the OSP [60] instead N-FINDR for endmember identification [61] was given in [70], achieving similar speedups and real-time unmixing results. Since UCLS provides abundances that are not subject to constraints, a non-negative abundance estimation method called ISRA [52], which was available in the form of an FPGA implementation [98], has been recently implemented in multi-core [99] and GPU [100] systems.

In this section, we describe the GPU implementation of the algorithms described in the previous section. They are carried out using the CUDA architecture developed by NVidia^{TM}. Fig. 3.3 illustrates

[3] http://www.top500.org

Figure 3.3: A GPU device (NVidia^{TM} Tesla).

the appearance of a NVidia^{TM} Tesla GPU device. As Fig. 3.4 shows, the architecture of a GPU can be seen as a set of multiprocessors. Each multiprocessor is characterized by a single instruction multiple data architecture, i.e., in each clock cycle, each processor executes the same instruction but operating on multiple data streams. Each processor access to a local shared memory and also to local cache memories in the multi-processor, while the multiprocessors have access to the global GPU (device) memory. GPUs can be abstracted in terms of a stream model, under which all data sets are represented as streams (i.e. ordered data sets). Algorithms are constructed by chaining so-called kernels which operate on entire streams and which are executed by a multiprocessor, taking one or more streams as inputs and producing one or more streams as outputs. Thereby, data-level parallelism is exposed to hardware, and kernels can be concurrently applied without any sort synchronization. The kernels can perform a kind of batch processing arranged in the form of a grid of blocks, where each block is composed by a group of threads that share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory (see Fig. 3.5). With the above ideas in mind, our GPU implementation of the hyperspectral unmixing chain comprises three stages: a) Estimation of number of endmembers on GPUs; b) Endmember extraction on GPUs; c) Abundance estimation on GPUs.

3.4.1 Estimation of the number of endmembers on GPU

In this subsection we detail the GPU implementation of two methods for estimating the number of endmembers in a hyperspectral scene: VD (described in 3.3.1.1) and HySime (described in 3.3.1.2)

3.4.1.1 GPU implementation of VD

The GPU implementation of VD algorithm, described in subsection 3.3.1.1, can be summarized as follows. Once we load the full hyperspectral image $\mathbf{Y}$ in the GPU memory, the first step is to calculate the covariance matrix $\mathbf{K}_{l\times l}$. For this purpose, we need to calculate the mean value of each band of the image and subtract this mean value to all the pixels in the same band. To perform this calculation in the GPU, we use a kernel called `centerData` configured with as many blocks as the number of bands in the hyperspectral image, l. In each block, all available threads perform a reduction process using shared memory and coalesced memory accesses to add the values of all the pixels from the same band. Once this process is completed, another thread divides the computed value by the number of pixels in the original

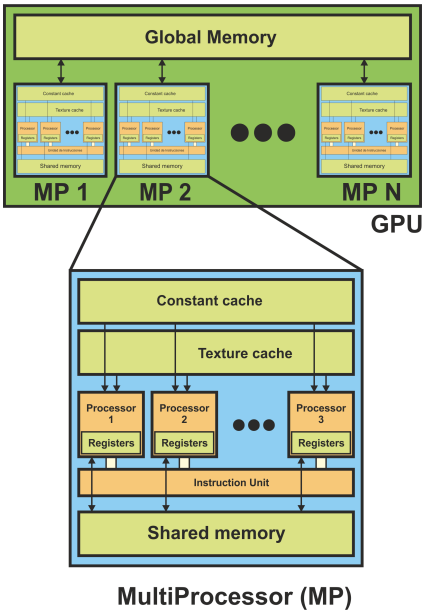


Figure 3.4: Schematic overview of a GPU architecture, which can be seen as a set of multiprocessors.

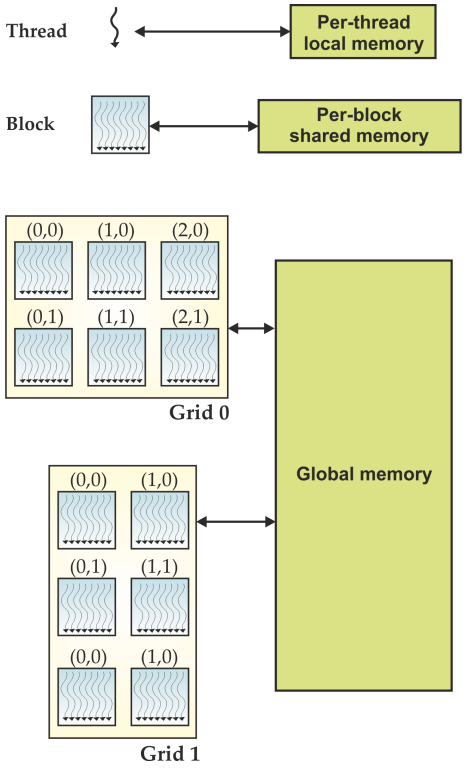


Figure 3.5: Different levels of memory in the GPU for the thread, block and grid concepts.

image, n, and then the mean value is obtained. In the last step, the kernel subtracts the mean value of each band to every pixel of that band, then the result is a matrix $\mathbf{Y} - \overline{\mathbf{Y}}$. This matrix is used to calculate the covariance matrix $\mathbf{K}_{l\times l}$ in the GPU by a matrix multiplication operation $(\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$. This

operation is performed using the cuBLAS[4] library. Specifically, we use the cublasSgemm function that executes efficient matrix multiplications on GPUs. The next step is to calculate the correlation matrix $\mathbf{R}_{l\times l}$ in the GPU. In order to achieve this, we use a kernel called computeCorrelation which launches as many threads as elements in the correlation matrix, so $l \times l$ threads, where each thread computes an element of the resulting matrix as follows: $\mathbf{R}_{ij} = \mathbf{K}_{ij} + \overline{\mathbf{Y}}_i\overline{\mathbf{Y}}_j$. Finally, we have observed that the remaining steps in the VD calculation (i.e., extraction of correlation-eigenvalues, covariance-eigenvalues and Neyman-Pearson test for estimation of the number of endmembers) can be computed very fast in the CPU.

3.4.1.2 GPU implementation of HySime

As subsection 3.3.1.2 shows, HySime is clearly divided in two stages: 1) noise estimation and 2) subspace estimation. In the first stage, once we load the hyperspectral image $\mathbf{Y}$ in GPU memory, we implement the noise estimation algorithm. For this purpose, the first step is to compute $\hat{\mathbf{R}}$ and its inverse, $\mathbf{R}'$. The former is calculated in the GPU by means of standard cuBLAS matrix multiplication (using cublasSgemm) while the latter is implemented in the CPU to avoid the high computational cost of the inverse operation in parallel (we tested the GPU communication time is bigger than inverse CPU execution).

We keep in the CPU the iterative procedure of calculating the noise estimation matrix for each band from the original image, since there are only l iterations. Then, for each *i-th* iteration we compute in the GPU two kernels and a matrix-matrix multiplication using cuBLAS. The first kernel, called computeBeta, performs the computation of all $\hat{\beta}$ values and stores them in a $l \times l$ matrix. This kernel has as many blocks as spectral bands in the original image. We have divided equation 3.1 in three parts described by the equations: 3.10, 3.11 and 3.12. Specifically, Eq. 3.10 multiplies both columns and the computed value is divided by the i-th element, it is optimized by keeping the i-th column in shared memory. Then Eq. 3.11 subtracts $\mathbf{AA}_i$ values to $[\mathbf{R}'_{\kappa_i,\kappa_i}]$ using one thread by row. Finally, Eq. 3.12 multiplies $\mathbf{XX}_i$ and $[\hat{\mathbf{R}}_{\kappa_i,i}]$ matrix.

$$\mathbf{AA}_i = [\mathbf{R}'_{\kappa_i,i}][\mathbf{R}'_{i,\kappa_i}]/[\mathbf{R}'_{i,i}] \tag{3.10}$$

$$\mathbf{XX}_i = [\mathbf{R}'_{\kappa_i,\kappa_i}] - \mathbf{AA}_i \tag{3.11}$$

$$\hat{\beta}_i = \mathbf{XX}_i[\hat{\mathbf{R}}_{\kappa_i,i}] \tag{3.12}$$

Once we have $\hat{\beta}$ matrix, the next step is to compute the operation $\hat{\xi}_i = \mathbf{z}_i - \mathbf{Z}_{\kappa_i}\hat{\beta}_i$. For this purpose, the multiplication $\mathbf{P} = \mathbf{Z}_{\kappa_i}\hat{\beta}_i$ is performed using cuBLAS, followed by the calculation of $\hat{\xi} = \mathbf{Z} - \mathbf{P}$. We can take advantage of this operation to subtract each element of $\mathbf{P}$ from each element of $\mathbf{z}_i$ in parallel. For this goal, we use a simple kernel, called subtractElements, which subtracts two structures of the same size using as many threads as elements in the structures.

On the other hand, the second stage is the subspace estimation algorithm. For this purpose, the first step is to calculate the noise correlation matrix $\hat{\mathbf{R}}_n = [\hat{\xi}_1, \cdots, \hat{\xi}_n]^T[\hat{\xi}_1, \cdots, \hat{\xi}_n]$, which could

[4]https://developer.nvidia.com/cublas

be calculated by means of a cuBLAS multiplication; since only diagonal elements of this matrix are considered, it is just necessary to compute the diagonal values: $\hat{\mathbf{R}}_{n_{i,i}} = [\hat{\xi}_{1_i}, \cdots, \hat{\xi}_{n_i}]^T [\hat{\xi}_{1_i}, \cdots, \hat{\xi}_{n_i}]$. Due to the number of intermediate structures needed in this process, the memory requirements can generate problems in some GPU devices. To address this issue, we implement this operation in the CPU as it is not computationally expensive. The second step of the subspace estimation is to calculate the signal correlation matrix $\hat{\mathbf{R}}_s = 1/n \sum_i ((\mathbf{y}_i - \hat{\xi}_i)(\mathbf{y}_i - \hat{\xi}_i^T))$, so we start by performing the operation $\left(\mathbf{y}_i - \hat{\xi}_i^T\right)$ with the subtractElements kernel. After this, we use again cuBLAS matrix multiplication to obtain $\hat{\mathbf{R}}_s$. Next, the eigenvectors $\mathbf{V}$ are calculated by SVD in the CPU using an optimized function (dsyevr_) of the widely used linear algebra package (LAPACK)[5]. The terms $\hat{\delta}_i$ are obtained based on the quadratic forms given by:

$$\hat{\delta}_{i_j} = \sum_{j=1}^{l} -\mathbf{q}_{i_j} + 2\sigma_{i_j}^2 \tag{3.13}$$

where $\mathbf{v}_i$ corresponds with the i-th eigenvector, $\mathbf{q}_{i_j} = \mathbf{v}_{i_j}^T \hat{\mathbf{R}}_s \mathbf{v}_{i_j}$ and $\sigma_{i_j}^2 = \mathbf{v}_{i_j}^T \hat{\mathbf{R}}_n \mathbf{v}_{i_j}$. We perform these operations using cuBLAS matrix multiplication. Finally, we apply a minimization function to get the estimate of p in the CPU, which is the number of $\hat{\delta}_i$ values smaller than 0. Hence, the signal subspace is defined by the p first eigenvectors of $\hat{\mathbf{R}}_s$.

3.4.2 Dimensionality reduction on GPU

In this subsection we detail the GPU implementation of the dimensionality reduction algorithm used in this work, PCA, which is described in 3.3.2.1.

3.4.2.1 GPU implementation of PCA

As 3.3.2.1 shows, the first step of PCA algorithm is the covariance matrix calculation $\mathbf{K}_{l \times l}$. For this purpose, the original image $\mathbf{Y}$ is copied into GPU memory, and then we use the kernel centerData (described in 3.4.1.2) to perform $\mathbf{Y} - \overline{\mathbf{Y}}$ that produces a matrix of the mean centered image data. This matrix is used to calculate the covariance matrix $\mathbf{K}_{l \times l}$ in the GPU by a matrix multiplication $(\mathbf{Y} - \overline{\mathbf{Y}})^T (\mathbf{Y} - \overline{\mathbf{Y}})$ using the cuBLAS function cublasSgemm.

Later, we apply the SVD to get the eigenvalues and eigenvectors ($\mathbf{V}_{l \times l}$) of the covariance matrix in the CPU. Once the eigenvalues and their associated eigenvectors are sorted in descending order, the eigenvectors are moved to the GPU to perform the projection of $\mathbf{Y}$ over them by the matrix multiplication $\mathbf{B} = \mathbf{YV}$. The resulting $n \times l$ matrix contains the bands sorted in descending order by the variance of their data. Thus, we can create a new image from the first $p - 1$ bands of $\mathbf{B}$, which contains the majority of the original image information. Furthermore, we can accelerate the process by using in the last multiplication the first $p - 1$ eigenvectors (or columns) instead of the complete $\mathbf{V}$ matrix.

3.4.3 Endmember extraction on GPU

This subsection details the GPU implementation of the endmember extraction algorithms: N-FINDR (described in 3.3.3.1) and OSP-GS (described in 3.3.3.2).

[5] http:www.netlib.orglapack

3.4.3.1 GPU implementation of N-FINDR

The GPU implementation of the N-FINDR algorithm, described in subsection 3.3.3.1, can be summarized as follows. Prior to the implementation on the GPU, a series of optimizations were performed in the original method, which have even improved the serial implementation. Since the most time-consuming computation in the N-FINDR algorithm is the calculation of the determinants, we have optimized this step. The determinant of a non-singular matrix $\mathbf{V}$ is usually obtained from the factorization $\mathbf{PV} = \mathbf{LU}$ (where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is a unit lower triangular matrix, and $\mathbf{U}$ is an upper triangular matrix) as the product of the diagonal elements of $\mathbf{U}$. This decomposition is known as *Gaussian elimination* or $\mathbf{LU}$ factorization (with partial row pivoting). The repeated volume calculations of the N-FINDR algorithm can be reduced by exploiting some basic properties of the $\mathbf{LU}$ factorization and matrix determinants. Consider, i.e., the $p \times p$ and $p \times p-1$ matrices:

$$\begin{aligned} V_{\mathbf{B}}^{(1)} &= \begin{bmatrix} 1 & \dots & 1 & 1 \\ \mathbf{m}_2^{(0)} & \dots & \mathbf{m}_p^{(0)} & \mathbf{y}_j \end{bmatrix} \textit{and} \\ \bar{V}_{\mathbf{B}}^{(1)} &= \begin{bmatrix} 1 & \dots & 1 \\ \mathbf{m}_2^{(0)} & \dots & \mathbf{m}_p^{(0)} \end{bmatrix}, \end{aligned} \tag{3.14}$$

where $\mathbf{B}$ is the reduced version of the hyperspectral image $\mathbf{Y}$ with p components, obtained as result of the PCA transform which is also performed in the GPU as described in the previous subsection.

Let assume that we have computed the $\mathbf{LU}$ factorization (with partial pivoting) $\mathbf{P}_B\overline{\mathbf{V}}_B^{(1)} = \mathbf{L}_B\mathbf{U}_B$. Then, the LU factorization (with partial pivoting) of $\mathbf{V}_B^{(1)}$ is simply given by $\mathbf{P}_B\mathbf{V}_B^{(1)} = \left[\mathbf{U}_B(\mathbf{L}_B^{-1}\mathbf{P}_B^T\mathbf{y}_j)\right]$. Therefore, the $\mathbf{LU}$ required factorizations for the N-FINDR volume calculations can be all computed by simply forming the $p \times m$ matrix $\hat{\mathbf{B}} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ & & \mathbf{B}^T & \end{bmatrix}$, where $\mathbf{B}$ is the reduced hyperspectral image.

Then, we need to compute ${\mathbf{L}_B}^{-1}{\mathbf{P}_B}^T\hat{\mathbf{B}}$. This is one of the parts that we accomplish in the GPU by means of a *Volume-Calculation* kernel which obtains the pixel volume in one iteration. The n volumes required for the first iteration of the N-FINDR algorithm are calculated from the product of the determinant of $\mathbf{U}_B$ and the last row of $\mathbf{L}_B^{-1}\mathbf{P}_B^T\hat{\mathbf{B}}$. By means of a `maxVolume` kernel, we get the value of the maximum volume and the location of the pixel that produces such volume. In the following, we describe the process in step-by-step fashion:

1. *Initialization.* First, we form a $p \times p$ matrix $\overline{\mathbf{V}}_B^{(1)}$ by initializing the first row to ones and setting in each row (from the second one) a random endmember. The determinant of the resulting matrix is calculated and the result is sorted in the `currentVolume` variable. Since the matrix dimension is small, the determinant is performed in the CPU. On the other hand, we create the n-elements vector `VolVector` which is used to store in the k-th position the volume resulting from an endmember replacement for the i-th pixel. Therefore, the reduced image $\mathbf{B}$ is modified by inserting a new first band of ones and getting $\hat{\mathbf{B}}$.

2. *Volume calculation.* In each k-iteration we replace in $\overline{\mathbf{V}}_B^{(1)}$ the current endmember of the k-th position for the endmember from p-th position, in addition to replacing the p-th column by a column with this format: $\begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}$. Then the $\mathbf{LU}$ factorization is applied to this matrix and we obtain $\mathbf{L_B}, \mathbf{U_B}$ and $\mathbf{P_B}$. After that, we compute the determinant of $\mathbf{U}_B$ and the inverse of $\mathbf{L}_B$. Since those matrices are triangular and small, we can execute the determinant and the inverse in the

CPU without any time penalty. At this point, we have the necessary elements to calculate the volume in just one iteration. Due to the fact that these elements are calculated by multiplying the determinant of $\mathbf{U}_B$ by the last row of $\mathbf{L}_B^{-1}\mathbf{P}_B^T\hat{\mathbf{B}}$, we divide this process in two phases: first, we perform the matrix multiplication $\mathbf{S} = \mathbf{L}_B^{-1}\mathbf{P}_B^T$ in the CPU. The second phase, due to its high computational cost, it is performed in the GPU using the kernel `VolumeCalculation`. This last kernel executes a multiplication of $\mathbf{U}_B$ by the last row of the $p \times m$ matrix $\mathbf{S}\hat{\mathbf{B}}$, therefore, we have optimized the $\mathbf{S}\hat{\mathbf{B}}$ operation by multiplying only the last row of $\mathbf{S}$ by $\hat{\mathbf{B}}$, so we save the calculations of obtaining the first $p-1$ rows.

3. *Replacement.* Once we have computed the volumes of one iteration, the next step is to find the pixel which generates the biggest volume and check if this volume is bigger than *currentVolume*. For this purpose, we use the `ReductionVol` kernel, which performs a reduction process in which j blocks work on different `VolVector` sections, so that each block calculates the local maximum and identifies its position. At the end of the execution, due to the fact that there are as many produced values as blocks (each block has its local maximum), it is necessary to keep those local maximums and their positions in global memory and later copy them to the CPU memory. Thus, the last reduction step focused on comparing the j elements is executed in the CPU, since $j << n$. At the end of each k-iteration, the new volume is compared with the previous one $(k-1)$. If the new one is the biggest, then the k-th endmember is replaced for the testing pixel and the *currentVolume* is updated.

The volume calculation and replacement steps are repeated for all the pixel vectors in the input data until all the pixels have been exhausted.

3.4.3.2 GPU implementation of OSP-GS

The GPU implementation of the OSP-GS algorithm, described in subsection 3.3.3.2, is given below. The algorithm is based on two main steps.

1. *Initialization.* The first step is related to the proper arrangement of the hyperspectral data in the local GPU memory. In order to optimize accesses, bearing in mind that the OSP-GS algorithm uses the pixel vector as minimum unit of computation, we store the pixel vectors of the hyperspectral image $\mathbf{Y}$ by columns. This arrangement is intended to access consecutive wavelength values in parallel by the processing kernels, so the i-th thread of a block will access the i-th wavelength component of a pixel vector of $\mathbf{Y}$. This technique is used to maximize global memory bandwidth and minimize the number of bus transactions. Once the image is stored on GPU memory, a structure is created in which there are as many blocks as pixel vectors are in the hyperspectral image divided by the number of threads per block, where the maximum number of supported threads depends on the considered GPU architecture. A kernel called `pixelMaxBright` is now used to calculate the brightest pixel $\mathbf{m}_1$ in $\mathbf{Y}$. This kernel computes the dot product between each pixel vector and its own transposed version, retaining the pixel that results in the maximum projection value.

2. *Orthogonal vector calculation.* Once the brightest pixel in $\mathbf{Y}$ has been identified, the pixel is allocated as the first column in matrix $\mathbf{M}$. The algorithm now calculates the orthogonal vectors through the Gram-Schmidt method as detailed in Eq. (3.5). Since the data structure dimension is small, the orthogonal vectors are performed in the CPU. A new kernel called `pixelProjection`

is created to project the orthogonal vector onto each pixel in the image, in which there are as many blocks as pixel vectors are in the hyperspectral image divided by the number of supported threads which depends on the considered GPU architecture. An important optimization applied at this point involves the effective use of the shared memories, which are used to store the most orthogonal vectors obtained at each iteration of OSP-GS (this is because these vectors will be accessed every time that the projection onto each pixel of the image is performed). The maximum of all projected pixels, is calculated using a separate reduction kernel `reductionProjection` which also uses the shared memory to store each of the projections and obtains the new endmember $\mathbf{m}_2$. The algorithm now extends the endmember matrix as $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2\}$. The second step process is repeated until the desired number of endmembers p has been extracted, $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_p\}$

3.4.4 Abundance estimation on GPU

This subsection describes the GPU implementation of the abundance endmember estimation algorithms UCLS (described in 3.3.4.1) and ISRA (described in 3.3.4.2).

3.4.4.1 GPU implementation of UCLS

As subsection 3.3.4.1 shows, Eq. 3.7 resolves the problem of estimating the abundance of the endmembers in a hyperspectral image $\mathbf{Y}$ without any restriction applied. In order to simplify the problem, we transform this equation into a matrix form:

$$\mathbf{A} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Y}^T, \tag{3.15}$$

where $\mathbf{A}$ is the abundance matrix, $\mathbf{M}$ is the endmember matrix and $\mathbf{Y}$ is the original hyperspectral image matrix. Due to its matricial nature, `cuBLAS` multiplications seems the best way to calculate the last equation, but we have tested that the performance is higher if the first two multiplications and the inverse are executed in the CPU, and then the last and biggest multiplication is performed in the GPU, thus, the algorithm is divided into two stages. In the first stage, a $p \times l$ matrix is calculated in the CPU by multiplying $\mathbf{C} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$, which is invariant for the abundance calculations of all the image pixels.

On the other hand, the second stage is performed in the GPU by a kernel called `UCLS`. This kernel uses the maximum number of threads available in the GPU architecture, at least the number of pixels, n. The main picture of this kernel is to multiply every row of $\mathbf{C}$ by every image pixel $\mathbf{y}_i$, which produces a $p \times n$ abundance matrix $\mathbf{A} = [\alpha_1, \alpha_2, \cdots, \alpha_p]$. For this purpose, $\mathbf{C}$ matrix is stored in shared memory, then each i-th thread load the i-th pixel, $\mathbf{y}_i$.

3.4.4.2 GPU implementation of ISRA

As subsection 3.3.4.2 shows, ISRA follows an iterative process to calculate the non-negative abundances of previously identified p endmembers in the image $\mathbf{Y}$ using Eq. 3.8. This expression is transformed to matrix form:

$$\mathbf{A}^{k+1} = \mathbf{A}^k \otimes \frac{\mathbf{YM}}{\mathbf{A}^k\mathbf{M}^T\mathbf{M}} \tag{3.16}$$

where $\mathbf{M} = \{\mathbf{m}_i\}^p{}_{i=1}$ is the endmember matrix, $\mathbf{A}$ denotes the p-dimensional abundance maps estimated for every pixel of $\mathbf{Y}$, the abundance matrix is iteratively updated. The symbol $\otimes$ indicates the multiplication -element by element- of $\mathbf{A}$ by the matrix obtained from $\frac{\mathbf{YM}}{\mathbf{A}^k\mathbf{M}^T\mathbf{M}}$. The GPU

implementation of Eq. 3.16 is performed with two kernels and three `cuBLAS` multiplications. At the beginning $\mathbf{A}$ is initialized with positive values, in order to keep the non-negative restriction, so a kernel called `ONES` initializes the abundance matrix with ones using as many threads as matrix elements, $n \times p$.

In the iterative process, Eq. 3.16 has an invariant numerator $\mathbf{YM}$ for all the iterations, so before starting to iterate, we perform $\mathbf{Num} = \mathbf{YM}$ by a `cublasSgemm` multiplication. During the iterative process, we perform first $\mathbf{Aux} = \mathbf{A}^k\mathbf{M}^T$ and later $\mathbf{Den} = \mathbf{AuxM}$ using `cublasSgemm` multiplication. At the end, a second kernel `UPDATE`, which uses as many threads as elements in $\mathbf{A}$, calculates $\mathbf{A}^{k+1} = \mathbf{A}^k \otimes \frac{\mathbf{Num}}{\mathbf{Den}}$. In this last kernel, the i-th thread executes three steps: first, it multiplies the i-th element of $\mathbf{A}$ by the i-th element of $\mathbf{Num}$; second, it divides the multiplication result by the i-th element of matrix $\mathbf{Dem}$; and third, it updates the i-th element of the abundance matrix $\mathbf{A}$ with the result of the previous division. The iterative process continues until a specific iteration threshold, which is considered enough to perform the abundances, is reached.

3.5 Quantitative metrics

In this section, we describe the comparative metrics used in this work for both algorithm accuracy evaluation and image retrieval. Two metrics have been used, the RMSE -for accuracy evaluation- and SAD -for both for accuracy evaluation and image retrieval-. Since SAD calculates the spectral similarity between to spectral signatures, it provides a perfect measure for content-based image retrieving. These two metrics are widely used in the hyperspectral unmixing literature [35] and constitute the main evaluation metrics included in our system.

3.5.1 Spectral angle distance (SAD)

The spectral angle measures the spectral similarity between two spectral signatures, where the best case is 0 degrees and the worst case is 90 degrees. For illustrative purposes, Fig. 3.6 shows the graphical representation of SAD. Let us assume that two spectral signatures are denoted by $\mathbf{y}_i$ and $\mathbf{y}_j$. These signatures are associated with any pair of hyperspectral image $\mathbf{Y}$ pixels and can be represented by two vectors $\mathbf{y}_i = [\mathbf{y}_i{}^{(1)}, \mathbf{y}_i{}^{(2)}, \cdots, \mathbf{y}_i{}^{(l)}]$ and $\mathbf{y}_j = [\mathbf{y}_j{}^{(1)}, \mathbf{y}_j{}^{(2)}, \cdots, \mathbf{y}_j{}^{(l)}]$, where $\mathbf{y}_i{}^{(k)}$ refers to the k-th spectral value of the $\mathbf{y}_i$ pixel, and $\mathbf{y}_j{}^{(k)}$ refers to the k-th spectral value of the $\mathbf{y}_j$ pixel, $k \in \{1, 2, \cdots, l\}$. The SAD value between $\mathbf{y}_i$ and $\mathbf{y}_j$ is the arc cosine of the spectral angle performed by such pixels in the l-dimension:

$$SAD(\mathbf{y}_i, \mathbf{y}_j) = arccos \frac{\mathbf{y}_i \cdot \mathbf{y}_j}{\| \mathbf{y}_i \| \cdot \| \mathbf{y}_j \|} = arccos \frac{\sum_{k=1}^{l} \mathbf{y}_i{}^{(k)} \cdot \mathbf{y}_j{}^{(k)}}{\sqrt{\sum_{k=1}^{l} \mathbf{y}_i{}^{(k)^2}} \cdot \sqrt{\sum_{k=1}^{l} \mathbf{y}_j{}^{(k)^2}}} \tag{3.17}$$

The SAD metric is invariant to multiplicative scalings that may arise due to different illumination conditions and sensor observation angle. Thus, this metric is commonly used to measure the spectral quality of the endmembers extracted with regards to some (possibly available) reference signatures. However, we use it to compare endmembers resulting from image results with reference signatures, and then retrieve them.

Due to the fact that reference signatures are mostly obtained in optimal conditions (e.g. in the laboratory) by a ground spectroradiometer, these signatures are not affected by atmospheric interferers,

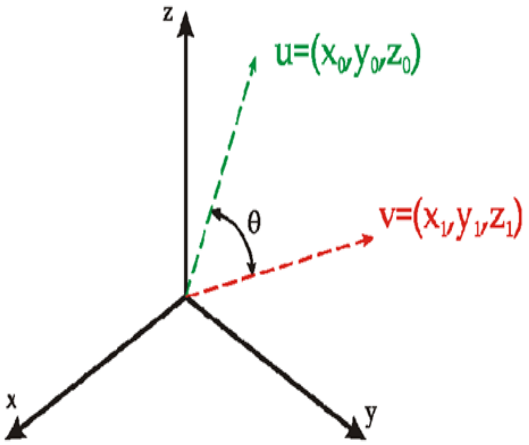


Figure 3.6: Graphic representation of SAD.

non-linear mixtures, geometric corrections, etc. In this regard, it is quite important to use a metric, like SAD, that is invariant to these effects.

3.5.2 Root mean square error (RMSE)

RMSE is a frequently used measure which evaluates the difference between the reconstructed image from the performed image spectral information (extracted endmembers and the estimated abundances) and the original data set. In our case, the reconstructed image $\hat{\mathbf{Y}}$ is calculated by multiplying the endmember matrix, $\mathbf{M}$, by the abundance matrix, $\mathbf{A}$. Therefore, the reconstructed error can be simply calculated as follows:

$$RMSE(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=1}^{l} [\mathbf{y}_i^{(k)} - \hat{\mathbf{y}}_i^{(k)}]^2 \right)^{1/2}, \tag{3.18}$$

where n is the number of pixels, l is the number of bands, $\mathbf{y}_i^{(k)}$ is the k-th band in the $\mathbf{y}_i$ pixel of the original hyperspectral image, and $\hat{\mathbf{y}}_i^{(k)}$ is the k-th band in the $\hat{\mathbf{y}}_i$ pixel of the reconstructed image, $k \in \{1, 2, \cdots, l\}$. This definition allows us to obtain the RMSE for each pixel and the complete image. The best case is zero, which means that pixels with error close to zero are better represented in the reconstructed image.

3.6 Experimental results

The performance of the proposed unmixing-based CBIR system has been evaluated from two different perspectives: its ability to retrieve hyperspectral images of interest from the set of catalogued ones available in the system, and its efficiency in cataloguing and retrieving hyperspectral images. Experiments have been conducted using both synthetic images (in a fully controlled environment) and also representative real hyperspectral images. The remainder of this section is organized as follows. First, we describe the synthetic and real hyperspectral data sets used in our experiments. Then, we perform an evaluation of the system from the viewpoint of retrieval accuracy. Finally, we perform an evaluation from the viewpoint of computational efficiency.

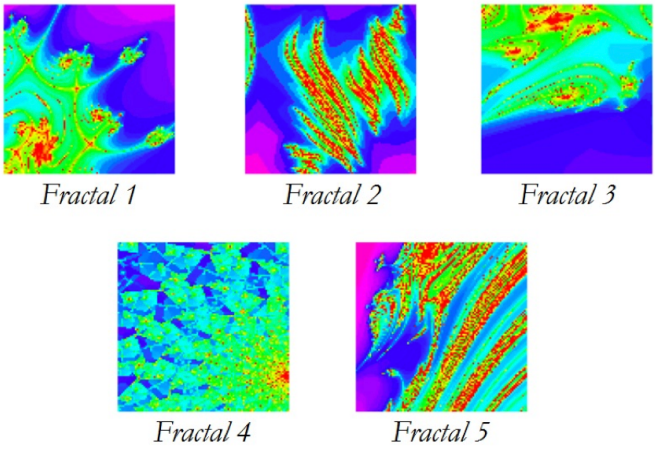


Figure 3.7: Fractal images used in our simulations.

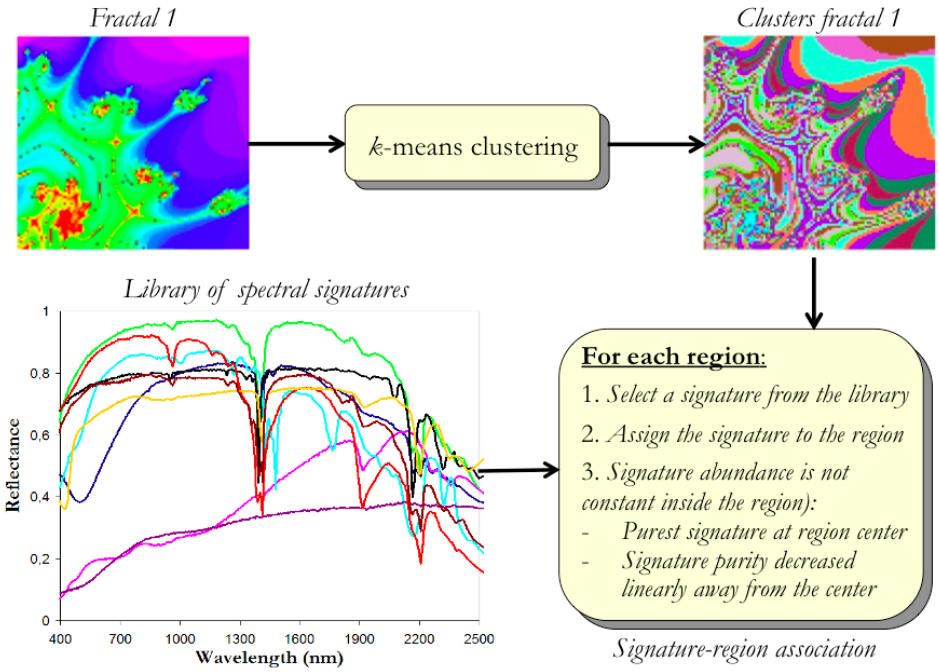


Figure 3.8: Procedure used for generating a synthetic hyperspectral image from a fractal image.

3.6.1 Hyperspectral data

3.6.1.1 Synthetic data

The main reason for using synthetic data in our evaluation of retrieval accuracy is that these kind of images can be generated in a fully controlled environment. As a result, algorithm accuracy can be effectively validated and tested. In this work, we have used a set of synthetically generated images using fractals. We have selected fractals because they can simulate naturally occurring patterns in nature. For illustrative purposes, Fig. 3.7 displays the five fractal images used in our simulations. The procedure for generating one of our simulated images is depicted in Fig. 3.8, in which a fractal image is used to simulate spatial patterns. The k-means clustering algorithm is adopted to select a set of clusters

from the fractal image. Then, a procedure starts, which assigns a set of spectral signatures from a spectral library to each region resulting from the clustering step mentioned before. A crucial step in the simulation procedure is how to assign a spectral signature to each cluster. For this purpose, we have implemented an automatic procedure that follows a simple strategy, in which $p = 9$ signatures are first assigned to spatially disjoint regions belonging to different clusters. The remaining regions are then assigned spectral signatures in an automatic way, ensuring that: (1) spatially adjacent clusters always have different signatures associated with them, and (2) there is a balance among the overall number of pixels in the image which are associated to each spectral signature. Inside each region, the abundance proportions of spectral signatures have been generated following a procedure that tries to imitate reality as much as possible, i.e. those pixels closer to the borders of the regions are more heavily mixed, while the pixels located at the center of the regions are more spectrally pure in nature. This is accomplished by linearly mixing the signature associated to each cluster with those associated with neighbouring clusters, making sure that the most spectrally pure signature remains at the center of the region while signature purity decreases linearly away from the center to the borders of the regions. With the aforementioned procedure, the simulated regions exhibit the following properties:

- All the simulated pixels inside a region are mixed, and the simulated image does not contain completely pure pixels. This increases the complexity of the unmixing problem and simulates the situation often encountered in real-world analysis scenarios, in which completely pure pixels are rarely found.
- Pixels close to the borders of the region are more heavily mixed than those in the center of the region.
- If the simulated region is sufficiently large, the pixels located at the center can exhibit a degree of purity of 99% of a certain endmember. However, if the size of the simulated region is small, the degree of purity of pixels at the center of the region can decrease until 95% of a certain endmember, while pixels located in the region borders are generally more heavily mixed.

To conclude the simulation process, zero-mean Gaussian noise was added to the scenes in different signal to noise ratios (SNRs) of 10:1, 30:1, 50:1, 70:1, 90:1 and 110:1 to simulate contributions from ambient and instrumental sources, following the procedure described in [60]. For illustrative purposes, Fig. 3.8 shows the spectra of the USGS library used in the simulation of one of the synthetic scenes (labeled as "Fractal 1"). In total, we considered five different fractal images (first simulated without noise, i.e. with SNR=∞) and then seven different versions of each scene corrupted with different noise levels, which gives a total of 35 synthetic images, all of them available in our system for public use.

3.6.1.2 Real data

In addition to the 35 synthetic images described in the previous section, our repository has currently 7 additional real hyperspectral images for a total of 42 hyperspectral images and total space of about 1 GB. It is worth noting that the repository is ready to receive additional scenes from end-users so that the database can grow. For the experiments that will be described in the following section, we have considered three well-known hyperspectral images with reference information (and which have been widely used in recent hyperspectral imaging literature) in order to substantiate both the retrieval accuracy and parallel performance of the proposed CBIR system. Since the images comprise different analysis scenarios, sizes and properties, our selection is expected to be sufficiently heterogeneous to provide an evaluation of the system from different perspectives.

Figure 3.9: AVIRIS hyperspectral image collected by NASA Jet Propulsion Laboratory over Indian Pines region.

- The first scene used in our experiments is the AVIRIS Indian Pines data set (see Fig. 3.9), which comprises 145 lines, 145 samples and 220 spectral channels between 0.4μm and 2.5μm with a nominal spectral resolution of 10 mm, and a total size of around 9 MB. This scene has been widely used as a benchmark in classification applications, and contains detailed ground-truth in the form of a ground-truth map with 16 mutually exclusive classes.

- The second scene used in our experiments was acquired by the AVIRIS sensor in 1995 over the mining Cuprite region, Nevada, U.S.A. Fig. 3.11 shows the image location over an aerial photograph of the region, in which the different mineral zones can be distinguished. It comprises 224 bands between 0.4μm and 2.5μm, with a nominal spectral resolution of 10 mm. Prior to the analysis, bands 1-2, 105-115, 150-170 and 223-224 were removed due to water abortion and low SNR in those bands, leaving 188 bands. A portion of 350×350 the original image has been selected that has a total size of around 50 MB. This scene has been widely used as a benchmark in spectral unmixing applications [25], since a complete reference information for this image is provided online by USGS. Among this information we have a spectral library with the spectral signatures of the minerals comprised by this scene and the classification map 3.12 of the main minerals in Cuprite region.

- The third scene used in our experiments is the AVIRIS World Trade Center data set, which was acquired by the AVIRIS sensor on 16th September in 2001 (five days later of the terrorist attack) over World Trade Center (WTC), New York. Fig. 3.10 shows the false color composition of the scene. It comprises 512 lines, 614 samples and 224 spectral channels between 0.4μm and 2.5μm with a nominal spectral resolution of 10 mm, and a total size of around 140 MB. It is important to emphasize the high spatial resolution of this image, which is 1.7 meter/pixel while the normal AVIRIS spatial resolution is 20 meters/pixel. This scene, which has been widely used as a benchmark in target and anomaly detection applications, comprises reference information available in several forms[6].

[6] http://www.speclab.cr.usgs.gov/wtc

Figure 3.10: AVIRIS hyperspectral image collected by NASA Jet Propulsion Laboratory over the World Trade Center in New York City on Sept. 16, 2001.

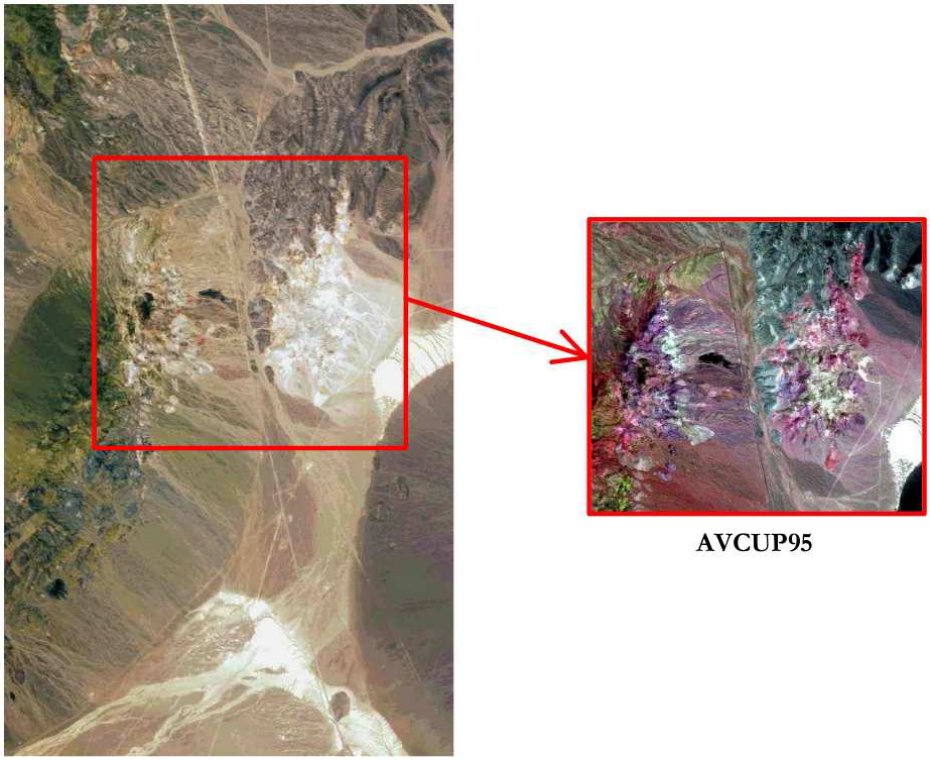


Figure 3.11: Location of AVIRIS hyperspectral Cuprite image over an aerial photograph of Cuprite mining region, Nevada, 1995.

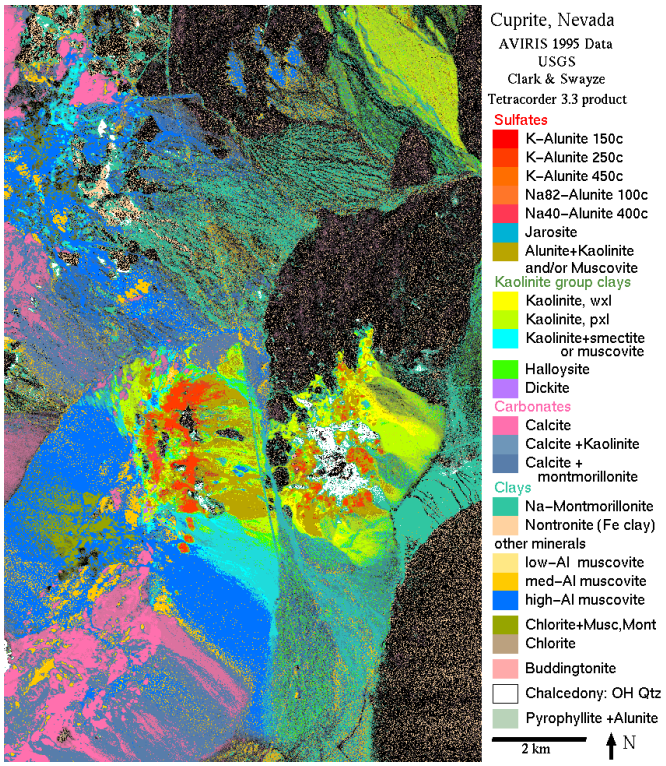


Figure 3.12: Classification map of AVIRIS Cuprite image (acquired using the USGS Tetracorder algorithm).

3.6.2 Evaluation of image retrieval accuracy

In order to illustrate the performance of our CBIR system, we specifically address a case study in which the synthetic images described in the previous subsection are used to substantiate the retrieval accuracy using different noise conditions. For illustrative purposes, we also use the AVIRIS Cuprite scene to evaluate the retrieval accuracy with real hyperspectral data. The two metrics that we have used in our experiments (which are described in section 3.5) are the SAD in endmember comparison and the RMSE in the estimated abundance fractions.

In our experiments we use synthetic and real hyperspectral data sets. First, as synthetic image, we use six versions of a fractal image with six different signal to noise ratios (10:1, 30:1, 50:1, 70:1, 110:1 and no-noise version). Such fractal image was constructed using 9 spectral signatures from the USGS digital spectral library: KaolineiteKGA-l(wxyl), Dumortierite HS190.3B, Nontronite GDS41, Alunite GDS83 Na, Sphene HS189.3B, Pyrophyllite PYS1A, Halloysite NMNH10623, Muscovite GDS108 and Kaolinite CM9. On the other hand, we use the real image AVIRIS Cuprite since its well-known features provide adequate results for evaluating the system accuracy over real hyperspectral data sets.

The SAD and RMSE results are obtained after using a query based on the nine USGS spectral signatures that were used to construct the fractal synthetic images. In the following, we analyse the results achieved from cataloguing images with different unmixing chain combinations. In order to show the real accuracy of our system, the scores show the mean and the standard deviation of ten query executions with different catalogs.

First, Table 3.1 compares the SAD results obtained from both endmember extraction algorithms

implemented in this system: N-FINDR and OSP-GS. As shown by this table, the SAD scores for the synthetic scenes generally decrease as the amount of simulated noise is lower, and most of the scores for SNR levels of 30:1 or lower are below 10 degrees. In both algorithm results the accuracy is very good, although the best results are from OSP-GS, due to the fact that N-FINDR algorithm starts from a random endmember matrix and this causes variability in the results. However, since the worst case of the SAD is 90 degrees, these are considered to be good similarity scores. In other words, the SAD metric reveals that our system can effectively retrieve the scenes containing endmembers which are highly similar, spectrally, to those used in the set of input signatures used to launch the query.

The second evaluation is performed by using RMSE metric, which evaluates the abundance maps accuracy. The system provides two abundance estimation algorithms: UCLS and ISRA. In order to make a complete evaluation of the abundance maps, we analyse the results performed from the abundance estimation using the endmembers previously identified by both algorithms N-FINDR and OSP-GS. The RMSE scores are grouped according to the extraction algorithm in two tables: Table 3.2 shows the RMSE scores of cataloguing using N-FINDR combined with UCLS and ISRA; and the results of cataloguing using the endmembers identified by OSP-GS are displayed on Table 3.3. Both tables show RMSE scores for the synthetic scenes, which generally decrease as the amount of simulated noise is lower. The scores of Table 3.3 are the better, since the abundances have been estimated using endmembers from OSP-GS, which have shown better results in Table 3.1. The analysis of the abundance accuracy scores in Table 3.3 reveals lower RMSE results for UCLS but some of them are negative, while ISRA scores are higher and all of them are positive.

These RMSE results could not be obtained for the AVIRIS Cuprite scene since the ground-truth abundances are difficult to obtain in real scenarios, but in this case we used the RMSE between the original and the reconstructed scene using the endmembers and abundances derived by the considered unmixing chains. As Table 3.4 shows, the reconstruction errors obtained in this case were very low [39] in all the cases, indicating that the considered unmixing chains provide accurate abundance maps for real hyperspectral data sets.

In addition, it is important to analyze the abundance coverage achieved by a query performed over the datasets considered in these experiments using the 9 USGS spectral signatures. As subsection 3.2.2 describes, the abundance coverage is performed as the percentage of an endmember in the scene. Table 3.5 shows the abundance coverage obtained after cataloguing the images using N-FINDR+UCLS and N-FINDR+ISRA. In the same way, Table 3.6 shows the abundance coverage obtained after cataloguing the images using OSP-GS + UCLS and OSP-GS + ISRA. In order to show the accuracy of this technique, we provide the scores achieved by applying the abundance coverage algorithm over the ground-truth of the synthetic dataset (see Table 3.7). In our experiments, we have observed that this technique is very sensitive to endmember identification variations, thus the scores achieved by cataloguing the images are a bit different with regards to the ground-truth scores. However, the abundance percentages can be seen as the numerical values of each abundance map, and the abundance estimation accuracy has been demonstrated by using RMSE, so we consider that the abundance coverage scores are sufficient for image retrieval purposes.

3.6.3 Evaluation of parallel performance

The computational performance of the proposed CBIR system has been evaluated using CPU and GPU architectures available in the two CPU-GPU clusters of CETA-Ciemat (see section 2.3.4). The parallel algorithms were performed in the two clusters, the *CETA Production* cluster (*GPU 1* hereinafter) with

Table 3.1: Spectral angle distance (degrees) obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The similarity results obtained for the six synthetic scenes (on average) with different noise ratios, and for the AVIRIS Cuprite scene are reported. The scenes were catalogued using both N-FINDR and OSP-GS.

USGS	**Extraction**	**Signal to noise ratio**						**AVIRIS**
signature	**Algorithm**	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**	**Cuprite**
	N-FINDR	15.013	3.137	2.393	1.555	2.283	2.266	5.444
Kaolinite		±1.788	±1.406	±1.321	±1.513	±0.914	±2.592	±1.254
CM9	**OSP-GS**	12.345	1.293	0.158	0.104	0.104	0.104	3.731
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	15.459	1.687	0.192	0.116	0.194	0.115	3.597
Muscovite		±0.388	±0.006	±0.001	±0.001	±0.157	±0.001	±0.001
GDS108	**OSP-GS**	18.843	1.685	0.192	0.117	0.115	0.115	4.082
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	26.261	2.742	0.573	0.274	0.361	0.331	17.161
Halloysite		±2.257	±0.184	±0.405	±0.001	±0.059	±0.068	±1.147
NMNH106236	**OSP-GS**	24.203	2.842	0.602	0.568	0.571	0.570	18.625
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	14.656	1.682	0.205	4.762	2.702	3.334	9.651
Pyrophyllite		±0.708	±3.064	±0.082	±4.127	±5.404	±6.669	±1.225
PYS1A	**OSP-GS**	15.568	1.682	0.352	0.015	0.000	0.000	9.154
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	25.781	4.438	10.768	11.324	10.828	6.667	6.900
Sphene		±1.821	±1.798	±12.137	±10.968	±7.185	±7.384	±0.625
HS189.3B	**OSP-GS**	26.091	3.544	1.322	0.313	0.309	0.308	9.867
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	23.822	1.605	0.180	0.096	0.095	0.079	8.722
Alunite		±10.882	±0.229	±0.028	±0.034	±0.034	±0.029	±1.812
GDS83 Na	**OSP-GS**	14.744	1.941	0.234	0.144	0.126	0.124	8.751
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	15.483	1.583	0.268	5.746	0.092	2.882	11.504
Nontronite		±0.803	±0.450	±0.151	±7.308	±0.001	±5.039	±1.892
GDS41	**OSP-GS**	38.733	1.600	0.201	0.096	0.092	0.092	15.243
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	24.635	2.487	0.513	0.501	0.502	0.502	3.081
Dumortierite		±8.451	±1.327	±0.001	±0.001	±0.001	±0.001	±0.001
HS190.3B	**OSP-GS**	15.732	4.467	1.994	0.501	0.502	0.502	3.334
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001
	N-FINDR	22.221	8.192	0.182	0.018	0.000	0.000	9.696
KaolineiteKGa-l		±7.198	±13.181	±0.001	±0.001	±0.001	±0.001	±0.105
(wxyl)	**OSP-GS**	17.434	27.964	0.182	0.0185	0.000	0.000	10.522
		±0.001	±0.001	±0.001	±0.001	±0.001	±0.001	±0.001

NVidia™ TESLA M2050 GPU devices, and the *CETA Test* cluster (*GPU 2* hereinafter) with *NVidia*™ TESLA C1060 GPU devices. In both clusters, each GPU is connected to a Quad Core Intel Xeon at 2.93 GHz with 4 physical cores, and 24 GB of DDR3 1333 MHz SRAM memory. The cluster specifications are described in Table 2.1, and the computational specifications of both GPU devices are detailed in Table 2.2. The serial algorithms were executed in one of the available cores of the *CETA Production* cluster.

In this subsection, we analyze the system performance over three real images, with different sized. They are AVIRIS Indian Pine (145 lines, 145 samples and 220 spectral channels for a total size of 9 MBytes), AVIRIS Cuprite (350 lines, 350 samples and 188 spectral channels for a total size of 50 MBytes) and AVIRIS World Trade Center (512 lines, 614 samples and 224 spectral channels for a total size of 140 MBytes), all of them widely described in subsection 3.6.1.

As mentioned in section 3.3, our system includes six algorithms -two for each stage of the unmixing chain- which can be combined to produced eight different unmixing chains, although in our experiments we only consider two completely different chains because they cover the performance times of all the algorithms. The first unmixing chain, based on VD for identification of the number of endmembers, N-FINDR for endmember signature finding, and ISRA for non-negative abundance estimation. In the case of N-FINDR, a dimensionality reduction of the original scene using PCA is conducted, hence we report the times for the PCA and for the N-FINDR algorithm in this case. The second unmixing chain is made up of HySime for finding the number of endmembers, OSP-GS for extracting the endmember signatures, and ULS for unconstrained abundance estimation.

Before describing our results, it is important to emphasize that our GPU versions provide exactly the same results as the serial versions of the algorithms, implemented using the `gcc` (gnu compiler default) with optimization flag -O3. Since the hyperspectral images contained in our repository can all fit the video memory of a single GPU (between 3 and 4 GB), we report the processing results obtained in a single GPU, although our system is ready to use a cluster of GPUs in parallel if needed. This is mainly because the current volume of stored hyperspectral data makes computations manageable with a single GPU. Since the computational requirements of the system can be currently managed with one GPU unit, we have decided to provide results based on the utilization of a single GPU. In the future, if the system grows as we expect with the addition of new data sets from external users, we may need to resort to a multi-GPU implementation. This is perfectly feasible, since the system has been designed with this configuration in mind, although currently we only need to use one GPU device. Another important consideration is that working in a single GPU reduces communication time, particularly if the scenes can be allocated into a single GPU memory. However, the communication overheads for a full multi-GPU implementation could be significant and it would be necessary to fully test our implementation in this scenario. In any event, we expect the searching part to scale properly with the GPU number since our search strategy is based on comparing the spectral endmembers and the abundance fractions using standard distance metrics. The most challenging part would be the cataloguing of a new scene using different endmember identification techniques, as it will depend on the considered endmember identification strategy, but in this case the cataloguing will take place only once (at the beginning) and the results would then be immediately available for searching purposes.

In the following we analyze the processing times (in seconds) used by our system in the process of cataloguing three real hyperspectral scenes using two different unmixing chains. In each experiment, ten runs were performed and the mean values were reported. We include the initialization times in our experiments since, as mentioned before, these times are generally higher in a GPU cluster than in single GPU devices, which is due to the fact that the communications between the nodes of the cluster

introduce a slight delay in the process. The analysis is manly focused on the total time of each algorithm (initialization and processing time).

As shown by the following results, using only one GPU of the cluster can already significantly accelerate the catalog process for the three considered scenes, which takes only a few seconds in all cases and with significant speedups. The initialization times (retained in our results in order to give an idea of the performance of the system in a GPU cluster, even if only one GPU is really used for the calculations) are not significant. This means that, once a new hyperspectral scene has been uploaded into our system, its associated meta-data can be efficiently generated in automatic fashion for subsequent retrieval. It is also worth noting that the results of the cataloguing can be dynamically selected, i.e. the end-user may decide to use any algorithm combination to catalog a scene.

3.6.3.1 AVIRIS Indian Pines scene

Here, we analyze the results achieved using both chains and the AVIRIS Indian Pines. Table 3.8 shows the processing times of the first unmixing chain, based on VD, N-FINDR and ISRA and Table 3.9 shows the processing times of the second unmixing chain, based on HySime, OSP-GS and UCLSC. We consider 28 as the number of endmembers in all the cases since it is the mean of several executions using both VD and HySime. In both tables we describe the CPU and GPU times (including the initialization required in the GPU clusters) and the speedups achieved by the parallel implementation in both GPU clusters (*GPU 1* and *GPU 2*).

All the results are better in *GPU 1* cluster since its GPU devices are the most powerful available in these experiments. In any case we analyse the results considering the scores of both GPU architectures. Table 3.8 displays the time measured for the first unmixing chain, which achieved speedups between 3x and 50x. On the other hand, the second unmixing chain times are displayed on Table 3.9, which shows speedups between 1x and 20x. In the case of UCLS less significant results are achieved due to the small image size, which does not balance the time consumed by the GPU initialization.

3.6.3.2 AVIRIS Cuprite scene

Tables 3.10 and 3.11 show the processing times and speedups achieved for the GPU implementations of both chains mentioned above tested on the AVIRIS Cuprite scene. We consider 19 as the number of endmembers in all the cases since it is the mean of several estimation executions using both VD and HySime. In both tables we describe the CPU and GPU times (including the initialization required in the GPU clusters) and the speedups achieved by the parallel implementation in both GPU clusters (*GPU 1* and *GPU 2*).

Table 3.10 shows the processing times of the first unmixing chain, based on VD, N-FINDR and ISRA. GPU implementations achieved good speedups which are between 13x and 46x. If we focus on the difference of times between both clusters, we can see that some of the speedups are better in the less powerful GPU architecture, this fact can take place in cases in which all the threads provided by the GPU architecture are not used and the stream processor cores are less but more powerful. In this regard, *GPU 2* GPU devices provide 240 cores at 1.3 GHz and *GPU 1* GPU devices provide 444 cores at 1.14 GHz. In our case, the AVIRIS Cuprite image has $p = 19$ endmembers, so some of the calculations are not enough to cover all the available threads. On the other hand, the times obtained by the second chain based on Hysme, OSP-GS and UCLS, are shown in Table 3.11. As we can see, all the results are better in *GPU 1* device, and the speedups are between 2x and 43x.

3.6.3.3 AVIRIS World Trace Center scene

Tables 3.12 and 3.13 show the processing times and speedups achieved for the GPU implementations of both chains mentioned above using the AVIRIS World Trade Center image. We consider 30 as the number of endmembers in all the cases since it is the mean of several estimation executions using both VD and HySime. In both tables we describe the CPU and GPU times (including the initialization required in the GPU clusters) and the speedups achieved by the parallel implementation in both GPU clusters (*GPU 1* and *GPU 2*). As shown by these tables, the best speedups achieved over this scene range from 17x to 172x in total times, this fact demonstrates that our parallel implementations perform better over large hyperspectral images.

Table 3.2: Root mean reconstruction error (RMSE) in abundance estimation obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The accuracy of the abundance maps estimated from the six synthetic scenes (on average) with different noise ratios are reported. The scenes are catalogued using two spectral unmixing chains: N-FINDR + UCLS and N-FINDR + ISRA.

USGS	**Abundance**	**Signal to noise ratio**					
signature	**algorithm**	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**
Kaolinite CM9	**UCLS**	0.196 ±0.001	0.334 ±0.001	0.246 ±0.001	0.059 ±0.001	-0.001 ±0.001	0.383 ±0.001
	ISRA	0.228 ±0.159	0.139 ±0.101	0.114 ±0.007	0.103 ±0.114	0.107 ±0.009	0.078 ±0.014
Muscovite GDS108	**UCLS**	0.223 ±0.001	0.760 ±0.001	0.843 ±0.001	0.944 ±0.001	0.953 ±0.001	0.953 ±0.001
	ISRA	0.263 ±0.149	0.390 ±0.070	0.389 ±0.096	0.799 ±0.823	0.418 ±0.087	0.419 ±0.106
Halloysite NMNH106236	**UCLS**	-0.029 ±0.001	0.035 ±0.001	-0.002 ±0.001	0.002 ±0.001	-0.064 ±0.001	-0.0001 ±0.001
	ISRA	0.129 ±0.263	0.116 ±0.046	0.094 ±0.023	0.137 ±0.112	0.068 ±0.017	0.094 ±0.026
Pyrophyllite PYS1A	**UCLS**	0.194 ±0.001	0.042 ±0.001	-0.087 ±0.001	0.001 ±0.001	0.232 ±0.001	-0.028 ±0.001
	ISRA	0.238 ±0.177	0.646 ±0.365	0.069 ±0.014	0.130 ±0.120	0.083 ±0.013	0.074 ±0.014
Sphene HS189.3B	**UCLS**	0.0266 ±0.001	0.006 ±0.001	0.009 ±0.001	-0.005 ±0.001	-0.058 ±0.001	-0.076 ±0.001
	ISRA	0.147 ±0.231	0.136 ±0.121	0.174 ±0.005	0.128 ±0.083	0.154 ±0.004	0.184 ±0.006
Alunite GDS83 Na	**UCLS**	0.221 ±0.001	-0.085 ±0.001	-0.031 ±0.001	0.003 ±0.001	-0.028 ±0.001	-0.107 ±0.001
	ISRA	0.275 ±0.153	0.0.066 ±0.021	0.061 ±0.016	0.858 ±1.630	0.063 ±0.015	0.063 ±0.0171
Nontronite GDS41	**UCLS**	-0.109 ±0.001	-0.003 ±0.001	-0.002 ±0.001	0.001 ±0.001	0.035 ±0.001	-0.096 ±0.001
	ISRA	0.102 ±0.234	0.632 ±0.466	0.069 ±0.011	0.104 ±0.093	0.080 ±0.011	0.071 ±0.011
Dumortierite HS190.3B	**UCLS**	0.150 ±0.001	-0.054 ±0.001	-0.020 ±0.001	0.003 ±0.001	0.001 ±0.001	-0.030 ±0.001
	ISRA	0.219 ±0.186	0.677 ±0.438	0.075 ±0.016	0.094 ±0.043	0.074 ±0.014	0.077 ±0.017
KaolineiteKGa-l (wxyl)	**UCLS**	0.152 ±0.001	-0.055 ±0.001	0.024 ±0.001	-0.002 ±0.001	-0.000 ±0.001	0.001 ±0.001
	ISRA	0.240 ±0.235	0.077 ±0.027	0.078 ±0.012	0.346 ±0.614	0.085 ±0.012	0.081 ±0.012

Table 3.3: Root mean reconstruction error (RMSE) in abundance estimation obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The accuracy of the abundance maps estimated from the six synthetic scenes (on average) with different noise ratios are reported. The scenes are catalogued using two spectral unmixing chains: OSP-GS + UCLS and OSP-GS + ISRA.

USGS signature	**Abundance algorithm**	**Signal to noise ratio**					
		10:1	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**
Kaolinite CM9	**UCLS**	0.155 ±0.001	0.156 ±0.001	0.071 ±0.001	0.049 ±0.001	0.048 ±0.001	0.048 ±0.001
	ISRA	0.147 ±0.001	0.089 ±0.001	0.082 ±0.012	0.083 ±0.012	0.083 ±0.012	0.076 ±0.001
Muscovite GDS108	**UCLS**	0.041 ±0.001	0.841 ±0.001	0.948 ±0.001	0.952 ±0.001	0.952 ±0.001	0.952 ±0.001
	ISRA	0.570 ±0.001	0.484 ±0.001	0.409 ±0.104	0.406 ±0.104	0.406 ±0.001	0.406 ±0.001
Halloysite NMNH106236	**UCLS**	0.074 ±0.001	0.082 ±0.001	0.004 ±0.001	0.006 ±0.001	-0.001 ±0.001	-0.001 ±0.001
	ISRA	0.063 ±0.001	0.057 ±0.001	0.099 ±0.026	0.095 ±0.026	0.094 ±0.026	0.080 ±0.001
Pyrophyllite PYS1A	**UCLS**	0.084 ±0.001	0.029 ±0.001	0.002 ±0.001	0.001 ±0.001	-0.000 ±0.001	-0.000 ±0.001
	ISRA	0.086 ±0.001	0.080 ±0.001	0.071 ±0.013	0.071 ±0.013	0.071 ±0.013	0.063 ±0.001
Sphene HS189.3B	**UCLS**	0.026 ±0.001	0.060 ±0.001	-0.011 ±0.001	-0.010 ±0.001	-0.001 ±0.001	-0.002 ±0.001
	ISRA	0.035 ±0.001	0.118 ±0.001	0.179 ±0.006	0.184 ±0.005	0.184 ±0.005	0.181 ±0.001
Alunite GDS83 Na	**UCLS**	0.094 ±0.001	-0.060 ±0.001	-0.003 ±0.001	0.001 ±0.001	-0.000 ±0.001	-0.000 ±0.001
	ISRA	0.094 ±0.001	0.058 ±0.001	0.060 ±0.017	0.061 ±0.017	0.061 ±0.016	0.052 ±0.001
Nontronite GDS41	**UCLS**	0.028 ±0.001	-0.027 ±0.001	-0.002 ±0.001	0.001 ±0.001	-0.001 ±0.001	-0.001 ±0.001
	ISRA	0.035 ±0.001	0.089 ±0.001	0.065 ±0.010	0.068 ±0.010	0.068 ±0.010	0.062 ±0.001
Dumortierite HS190.3B	**UCLS**	0.199 ±0.001	0.002 ±0.001	-0.002 ±0.001	-0.001 ±0.001	0.001 ±0.001	0.001 ±0.001
	ISRA	0.193 ±0.001	0.064 ±0.001	0.075 ±0.016	0.075 ±0.016	0.075 ±0.016	0.065 ±0.001
KaolineiteKGa-l (wxyl)	**UCLS**	0.261 ±0.001	0.090 ±0.001	0.016 ±0.001	0.001 ±0.001	0.000 ±0.001	0.001 ±0.001
	ISRA	0.260 ±0.001	0.063 ±0.001	0.074 ±0.012	0.076 ±0.011	0.076 ±0.011	0.067 ±0.001

Table 3.4: Root mean reconstruction error (RMSE) in the reconstructed scene using endmember and abundances estimated over the AVIRIS Cuprite scene. The results obtained for the Cuprite AVIRIS scene are reported. The scene was catalogued using four spectral unmixing chains: OSP-GS + UCLS, OSP-GS + ISRA, N-FINDR + UCLS and N-FINDR + ISRA.

N-FINDR		**OSP-GS**	
UCLS	**ISRA**	**UCLS**	**ISRA**
0.217	0.189	0.219	0.190
±0.006	±0.005	±0.001	±0.001

Table 3.5: Percentage of the estimated abundances in the image obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The similarity results obtained for the five synthetic scenes (on average) with different noise ratios, and for the AVIRIS Cuprite scene are reported. The scenes are catalogued using two spectral unmixing chains: N-FINDR + UCLS and N-FINDR + ISRA.

USGS	**Abundance**	**Signal to noise ratio**						**AVIRIS**
signature	**algorithm**	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**	**Cuprite**
Kaolinite CM9	**UCLS**	14.473 ±0.001	7.803 ±0.001	7.031 ±0.001	10.634 ±0.001	14.814 ±0.001	8.700 ±0.001	10.937 ±0.001
	ISRA	11.185 ±0.503	8.630 ±0.456	8.565 ±0.023	13.946 ±0.001	12.711 ±0.001	9.480 ±0.001	10.854 ±0.001
Muscovite GDS108	**UCLS**	14.473 ±0.004	14.298 ±0.003	8.345 ±0.001	19.556 ±0.001	9.126 ±0.001	11.378 ±0.001	4.782 ±0.001
	ISRA	11.185 ±0.522	9.546 ±0.448	9.121 ±0.081	22.546 ±0.001	14.223 ±0.001	8.486 ±0.001	5.534 ±0.001
Halloysite NMNH106236	**UCLS**	9.275 ±0.001	11.230 ±0.001	13.201 ±0.001	3.057 ±0.001	5.349 ±0.001	13.711 ±0.001	6.074 ±0.001
	ISRA	11.034 ±0.308	10.761 ±0.221	9.356 ±0.091	9.579 ±0.001	8.620 ±0.001	9.383 ±0.001	3.702 ±0.001
Pyrophyllite PYS1A	**UCLS**	8.520 ± 0.004	8.530 ± 0.002	8.746 ± 0.001	3.057 ±0.001	6.753 ±0.001	5.523 ±0.001	6.181 ±0.001
	ISRA	11.188 ±0.736	8.592 ±0.608	12.039 ±0.131	9.743 ±0.001	8.937 ±0.001	9.383 ±0.001	2.925 ±0.001
Sphene HS189.3B	**UCLS**	8.794 ±0.010	11.041 ±0.006	18.032 ±0.001	4.399 ±0.001	8.407 ±0.001	11.923 ±0.001	6.856 ±0.001
	ISRA	11.147 ±0.653	8.591 ±0.364	8.623 ±0.116	8.876 ±0.001	9.171 ±0.001	11.560 ±0.001	5.261 ±0.001
Alunite GDS83 Na	**UCLS**	11.630 ±0.001	11.659 ±0.001	6.409 ±0.001	4.457 ±0.001	15.142 ±0.001	3.149 ±0.001	1.941 ±0.001
	ISRA	11.100 ±0.201	9.298 ±0.138	8.585 ±0.006	8.924 ±0.001	18.231 ±0.001	8.972 ±0.001	3.180 ±0.001
Nontronite GDS41	**UCLS**	15.646 ±0.001	9.542 ±0.001	14.762 ±0.001	6.458 ±0.001	22.560 ±0.001	30.932 ±0.001	4.116 ±0.001
	ISRA	11.130 ±0.191	13.048 ±0.191	13.842 ±0.001	8.762 ±0.001	8.436 ±0.001	8.574 ±0.001	7.785 ±0.001
Dumortierite HS190.3B	**UCLS**	11.791 ±0.001	9.542 ±0.001	9.338 ±0.001	23.623 ±0.001	3.698 ±0.001	9.265 ±0.001	2.975 ±0.001
	ISRA	11.067 ±0.223	13.048 ±0.189	8.978 ±0.051	8.840 ±0.001	10.023 ±0.001	20.646 ±0.001	8.465 ±0.001
KaolineiteKGa-l (wxyl)	**UCLS**	10.253 ±0.001	8.887 ±0.001	8.345 ±0.001	4.457 ±0.001	3.698 ±0.001	3.149 ±0.001	3.036 ±0.001
	ISRA	11.058 ±0.533	8.952 ±0.475	9.121 ±0.068	8.924 ±0.001	10.023 ±0.001	8.972 ±0.001	4.262 ±0.001

Table 3.6: Percentage of the estimated abundances in the image obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The similarity results obtained for the five synthetic scenes (on average) with different noise ratios, and for the AVIRIS Cuprite scene are reported. The scenes are catalogued using two spectral unmixing chains: OSP-GS + UCLS and OSP-GS + ISRA.

USGS	**Abundance**	**Signal to noise ratio**						**AVIRIS**
signature	**algorithm**	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**	**Cuprite**
Kaolinite CM9	**UCLS**	13.112 ±0.001	15.340 ±0.001	8.306 ±0.001	7.607 ±0.001	7.521 ±0.001	7.522 ±0.001	4.808 ±0.001
	ISRA	13.754 ±0.533	9.510 ±0.356	8.340 ±0.121	8.369 ±0.001	8.369 ±0.001	7.853 ±0.001	5.147 ±0.001
Muscovite GDS108	**UCLS**	11.761 ±0.003	11.241 ±0.002	18.431 ±0.001	19.130 ±0.001	19.170 ±0.001	19.170 ±0.001	4.390 ±0.001
	ISRA	12.046 ±0.301	11.979 ±0.098	22.057 ±0.010	21.400 ±0.001	21.396 ±0.001	20.862 ±0.001	5.483 ±0.001
Halloysite NMNH106236	**UCLS**	12.676 ±0.002	10.982 ±0.001	8.306 ±0.001	7.607 ±0.001	7.521 ±0.001	7.523 ±0.001	2.350 ±0.001
	ISRA	13.591 ±0.525	16.500 ±0.430	8.340 ±0.159	8.369 ±0.001	8.369 ±0.001	7.853 ±0.001	6.365 ±0.001
Pyrophyllite PYS1A	**UCLS**	9.951 ±0.002	10.158 ±0.001	10.200 ±0.	10.565 ±0.001	10.587 ±0.001	10.587 ±0.001	4.811 ±0.001
	ISRA	9.093 ±0.763	10.302 ±0.620	9.209 ±0.211	9.183 ±0.001	9.183 ±0.001	9.779 ±0.001	5.209 ±0.001
Sphene HS189.3B	**UCLS**	10.193 ±0.008	6.741 ±0.006	13.570 ±0.001	13.692 ±0.001	13.725 ±0.001	13.726 ±0.001	5.097 ±0.001
	ISRA	10.740 ±0.598	8.487 ±0.418	9.354 ±0.078	9.4096 ±0.001	9.409 ±0.001	9.909 ±0.001	3.721 ±0.001
Alunite GDS83 Na	**UCLS**	12.961 ±0.001	13.580 ±0.001	16.536 ±0.001	17.128 ±0.001	17.184 ±0.001	17.184 ±0.001	7.667 ±0.001
	ISRA	14.371 ±0.430	11.807 ±0.334	14.211 ±0.015	14.531 ±0.001	14.531 ±0.001	15.115 ±0.001	3.973 ±0.001
Nontronite GDS41	**UCLS**	10.748 ±0.001	14.186 ±0.001	7.640 ±0.001	7.764 ±0.001	7.775 ±0.001	7.774 ±0.001	8.615 ±0.001
	ISRA	11.181 ±0.400	10.502 ±0.341	8.364 ±0.010	8.374 ±0.001	8.374 ±0.001	7.976 ±0.001	8.850 ±0.001
Dumortierite HS190.3B	**UCLS**	10.988 ±0.001	12.029 ±0.001	10.499 ±0.001	10.869 ±0.001	10.894 ±0.001	10.894 ±0.001	6.994 ±0.001
	ISRA	7.909 ±0.490	12.677 ±0.295	11.422 ±0.021	11.544 ±0.001	11.544 ±0.001	12.240 ±0.001	4.269 ±0.001
KaolineiteKGa-l (wxyl)	**UCLS**	13.112 ±0.002	15.341 ±0.001	8.462 ±0.001	7.260 ±0.001	7.254 ±0.001	7.254 ±0.001	4.205 ±0.001
	ISRA	13.7548 ±0.654	9.510 ±0.345	8.417 ±0.161	8.462 ±0.001	8.461 ±0.001	7.965 ±0.001	7.675 ±0.001

Table 3.7: Abundance percentage of the 9 spectral signatures in the ground-truth of the synthetic image used in our experiments.

Kaolinite CM9	**Muscovite GDS108**	**Halloysite NMNH106236**	**Pyrophyllite PYS1A**	**Sphene HS189.3B**
7.607	10.805	18.194	14.086	17.182
Alunite GDS83 Na	**Nontronite GDS41**	**Dumortierite HS190.3B**	**KaolineiteKGa-l (wxyl)**	
7.886	11.125	7.066	6.047	

Table 3.8: Processing times (in seconds) and speedups achieved for the GPU implementation of VD, N-FINDR and ISRA algorithms, used to catalog the AVIRIS Indian Pines scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	VD			**N-FINDR**				**ISRA**		
	Initialize	**VD**	**Total**	**Initialize**	**PCA**	**N-FINDR**	**Total**	**Initialize**	**ISRA**	**Total**
CPU	0.063 ±0.057	3.248 ±0.252	3.311 ±0.252	0.021 ±0.005	1.742 ±0.021	0.937 ±0.260	2.700 ±0.256	0.025 ±0.003	44.253 ±0.513	44.278 ±0.514
GPU 1	0.498 ±0.084	0.027 ±0.001	0.525 ±0.084	0.512 ±0.0136	0.021 ±0.008	0.245 ±0.140	0.778 ±0.140	0.447 ±0.089	0.432 ±0.001	0.879 ±0.089
Speedup	-	120.296	6.307	-	82.952	3.824	3.470	-	102.437	50.373
GPU 2	0.586 ±0.118	0.053 ±0.001	0.639 ±0.128	0.539 ±0.200	0.029 ±0.001	0.225 ±0.087	0.793 ±0.150	0.560 ±0.098	0.460 ±0.001	1.020 ±0.098
Speedup	-	63.283	5.182	-	60.068	4.164	3.404	-	96.202	43.410

Table 3.9: Processing times (in seconds) and speedups achieved for the GPU implementation of HySime, OSP-GS and UCLS algorithms, used to catalog the AVIRIS Indian Pines scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	HYSIME			**OSP-GS**			**UCLS**		
	Initialize	**HYSIME**	**Total**	**Initialize**	**OSP-GS**	**Total**	**Initialize**	**UCLS**	**Total**
CPU	0.300 ±0.025	13.124 ±3.164	13.154 ±3.164	0.039 ±0.002	0.552 ±0.064	0.591 ±0.062	0.035 ±0.003	0.290 ±0.050	0.325 ±0.001
GPU 1	0.450 ±0.368	0.194 ±0.013	0.643 ±0.375	0.523 ±0.400	0.009 ±0.001	0.532 ±0.400	0.479 ±0.342	0.017 ±0.001	0.496 ±0.542
Speedup	-	63.610	20.457	-	61.333	1.111	-	17.059	0.655
GPU 2	0.613 ±0.214	0.316 ±0.001	0.929 ±0.214	0.576 ±0.197	0.014 ±0.006	0.590 ±0.198	0.619 ±0.116	0.022 ±0.001	0.641 ±0.242
Speedup	-	41.532	14.159	-	39.429	1.001	-	13.181	0.507

Table 3.10: Processing times (in seconds) and speedups achieved for the GPU implementation of VD, N-FINDR and ISRA algorithms, used to catalog the AVIRIS Cuprite scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	VD			N-FINDR				ISRA		
	Initialize	**VD**	**Total**	**Initialize**	**PCA**	**N-FINDR**	**Total**	**Initialize**	**ISRA**	**Total**
CPU	0.340	14.476	14.816	0.125	9.069	2.434	11.628	0.149	120.650	120.799
	±0.158	±2.134	±2.221	±0.020	±0.064	±0.440	±0.480	±0.021	±0.355	±0.355
GPU 1	0.579	0.127	0.706	0.583	0.075	0.134	0.792	0.535	2.083	2.618
	±0.147	±0.001	±0.148	±0.009	±0.002	±0.032	±0.032	±0.098	±0.001	±0.098
Speedup	-	113.094	20.986	-	120.920	18.164	14.682	-	57.921	46.142
GPU 2	0.736	0.217	0.953	0.608	0.122	0.120	0.850	0.570	2.019	2.589
	±0.169	±0.001	±0.171	±0.169	±0.001	±0.016	±0.132	±0.073	±0.001	±0.098
Speedup	-	66.709	15.547	-	74.336	20.283	13.680	-	59.757	46.658

Table 3.11: Processing times (in seconds) and speedups achieved for the GPU implementation of HySime, OSP-GS and UCLS algorithms, used to catalog the AVIRIS Cuprite scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	HYSIME			OSP-GS			UCLS		
	Initialize	**HYSIME**	**Total**	**Initialize**	**OSP-GS**	**Total**	**Initialize**	**UCLS**	**Total**
CPU	0.512	63.610	63.122	0.198	2.225	2.423	0.160	1.500	1.660
	±0.134	±0.231	±0.274	±0.015	±0.016	±0.022	±0.019	±0.012	±0.031
GPU 1	0.567	0.907	1.464	0.564	0.024	0.588	0.493	0.054	0.547
	±0.385	±0.008	±0.389	±0.202	±0.001	±0.202	±0.306	±0.001	±0.507
Speedup	-	70.132	43.116	-	92.708	4.121	-	27.778	3.035
GPU 2	0.736	1.558	2.294	0.619	0.034	0.653	0.763	0.065	0.828
	±0.215	±0.001	±0.221	±0.195	±0.001	±0.212	±0.421	±0.001	±0.476
Speedup	-	40.828	27.516	-	65.441	3.711	-	23.077	2.005

Table 3.12: Processing times (in seconds) and speedwells achieved for the GPU implementation of VD, N-FINDR and ISRA algorithms, used to catalog the AVIRIS World Trade Center scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	VD			N-FINDR				ISRA		
	Initialize	**VD**	**Total**	**Initialize**	**PCA**	**N-FINDR**	**Total**	**Initialize**	**ISRA**	**Total**
CPU	0.934	57.629	58.563	0.381	35.210	16.422	52.013	0.521	885.336	885.857
	±0.573	±8.678	±8.913	±0.496	±0.199	±5.925	±5.919	±0.066	±45.971	±46.006
GPU 1	0.683	0.279	0.962	0.710	0.216	0.837	1,763	0.679	4.460	5.139
	±0.084	±0.001	±0.169	±0.010	±0.006	±0.907	±0.908	±0.126	±0.014	±0.131
Speedup	-	202.919	60.876	-	163.009	19.620	29.502	-	198.506	172.379
GPU 2	0.621	0.617	1.340	0.720	0.243	0.927	1.699	0.595	6.595	7.190
	±0.084	±0.001	±0.169	±0.010	±0.006	±0.907	±0.911	±0.113	±0.006	±0.122
Speedup	-	93.402	43.70	-	144.897	22.312	30.613	-	134.244	123.207

Table 3.13: Processing times (in seconds) and speedups achieved for the GPU implementation of HySime, OSP-GS and UCLS algorithms, used to catalog the AVIRIS World Trade Center scene available in our CBIR system. The table reports the mean values and the standard deviations measured across ten algorithm executions.

	HYSIME			**OSP-GS**			**ULS**		
	Initialize	**HYSIME**	**Total**	**Initialize**	**OSP-GS**	**Total**	**Initialize**	**ULS**	**Total**
CPU	0.349 ±0.336	240.027 ±64.367	240.376 ±64.337	0.495 ±0.040	18.620 ±0.019	19.115 ±0.055	0.445 ±0.056	16.455 ±0.069	16.900 ±0.103
GPU 1	0.678 ±0.312	3.245 ±0.011	3.923 ±0.325	0.612 ±0.136	0.106 ±0.001	0.718 ±0.136	0.651 ±0.385	0.232 ±0.001	0.883 ±0.385
Speedup	-	73.968	61.274	-	175.660	26.622	-	70.927	19.139
GPU 2	0.657 ±0.184	4.690 ±0.132	5.357 ±0.342	0.451 ±0.225	0.103 ±0.001	0.554 ±0.229	0.711 ±0.161	0.261 ±0.001	0.972 ±0.201
Speedup	-	51.178	44.871	-	180.777	34.504	-	63.046	17.387

Chapter 4

Sparse unmixing-based image retrieval approach

As mentioned in previous chapters, the *geometric* techniques for endmember identification present some difficulties in practical application due to two main reasons. First, if the spatial resolution of the sensor is not high enough to separate different pure signatures classes at a macroscopic level, the resulting spectral measurement can be a composite of individual pure spectra which correspond to materials that jointly occupy a single pixel. In this case, the use of image-derived endmembers may not result in accurate fractional abundance estimations since in this case it is likely that such endmembers may not be completely pure in nature. Second, mixed pixels can also result when distinct materials are combined into a microscopic (intimate) mixture, which is independent from the spatial resolution of the sensor. Since the mixtures in this situation happen at the particle level, the use of image derived spectral endmember cannot accurately characterize intimate spectral mixtures. For these reasons we resort in this chapter to a solution based on sparse unmixing techniques [40], which take advantage of the increasing availability of the spectral libraries made from materials measured on the ground (i.e. using advanced field spectrometers), thus the results are expressed as linear combinations of several pure spectral signatures known in advance and available in a library.

More specifically, in this chapter we propose a new CBIR system [101] that improves the unmixing-based image retrieval system described in the previous chapters (*unmixing-based CBIR* hereinafter), and is also available online from http://hypercomp.es/repository. The new system uses linear sparse unmixing methods in order to extract the features from images stored in the database. In this way, the abundance maps of the spectral signatures of a library (collected in ideal laboratory conditions) are estimated for each image in the database. This results in a collection of spectral signatures automatically selected from the library and their associated abundance maps. In addition, our improved system provides a *sparsity* map for each image which shows the subset of signatures that can best model each mixed pixel in the scene. This approach has several advantages, on one hand, the endmembers are extracted from the spectral library instead from the original image, which generally enhances the searching process since the endmembers and the ground-truth spectral signatures are measured in the same conditions. On the other hand, the catalog process is performed in just one execution while the previous *unmixing-based CBIR* system requires executing the full unmixing chain (three stages in total), hence the cataloguing efforts are now reduced for the user. The sparse-based unmixing method developed in this chapter has been implemented by resorting to a constrained sparse unmixing by variable splitting and augmented Lagrangian (CSUNSAL) algorithm, which has been recently proposed [68] to cope with the abundance

fraction estimation from a set of spectral signatures (or spectral library). In this chapter, we exploit both serial and GPU implementations of the CSUNSAL method.

The remainder of the chapter is structured as follows. Section 4.1 describes some related work including the fundamentals of the CSUNSAL algorithm used to extract the spectral information. Section 4.2 describes how the CBIR performs queries to the database through the web interface in a step by step fashion. Section 4.3 makes a comparison of both CBIR approaches in terms of retrieval accuracy and parallel performance using real hyperspectral and synthetic scenes with different noise levels.

4.1 Related work

Linear sparse regression [14, 102] is a direction recently explored in spectral unmixing analysis that can be solved using constrained sparse regression methods such as ADMM [46]. The SUNSAL method introduced in [68] is an instance on ADMM, which decomposes a difficult problem into a sequence of simpler ones, resulting in a very fast method. In addition, the results of the method can be improved [40] by using physical constraints usually imposed on the unmixing problem such as the ASC. In this way, the CSUNSAL algorithm used in this work combines SUNSAL with the fully constrained least squares (FCLS) [36] method to include the ASC constraint.

In order to properly identify the image endmembers, spectral libraries with a large number of spectral signatures are generally fed to CSUNSAL which makes the algorithm quite slow. Indeed the executions on very large hyperspectral images can reach very long execution times. Hence we resort to an efficient GPU implementation of this algorithm. A parallel implementation of CSUNSAL has already been published in [103], in which the most time-consuming operations are performed in the GPU by exploiting of the architecture characteristics. On the other hand, the simplest operations, i.e. small matrix inversion, are executed in the CPU.

4.2 Sparse-based CBIR

In this chapter we develop a new cataloguing system based on sparse regression methods, which extracts the spectral features from every image in the database and stores the results as meta-data associated to each image. In most of the cases, the spectral resolution of the input signatures in the spectral library is much higher than the images stored in the database. In this regard, in order to allow to catalog images using spectral libraries with any resolution, we have implemented a spectral convolution strategy (described in Algorithm 2) that looks for wavelength values which are present in both the hyperspectral data and the input spectral library (with the possibility to include a tolerance threshold in the wavelength matching procedure). So, the resulting endmembers are adapted to the analyzed image, which means that the dimension of the spectral signatures of those endmembers has as many spectral bands as the image bands have been matched with the spectral library bands. In addition, we filter the best results by reducing to the most significant ones by applying a constraint based on a minimum abundance percentage (described in 3.2.2), since CSUNSAL produces as many endmembers as spectral signatures are present in the spectral library and most of them have hardly any useful information.

Regarding the searching procedure, the *sparse-based* CBIR uses the same methodology as the *unmixing-based* CBIR. So, for each new hyperspectral data set, the spectral endmembers and their corresponding abundance maps are obtained using the CSUNSAL algorithm, then the information provided by the algorithm execution is used as meta-data in the searching procedure. As Fig. 4.1 shows, the system allows an end-user to perform queries to the hyperspectral image database following

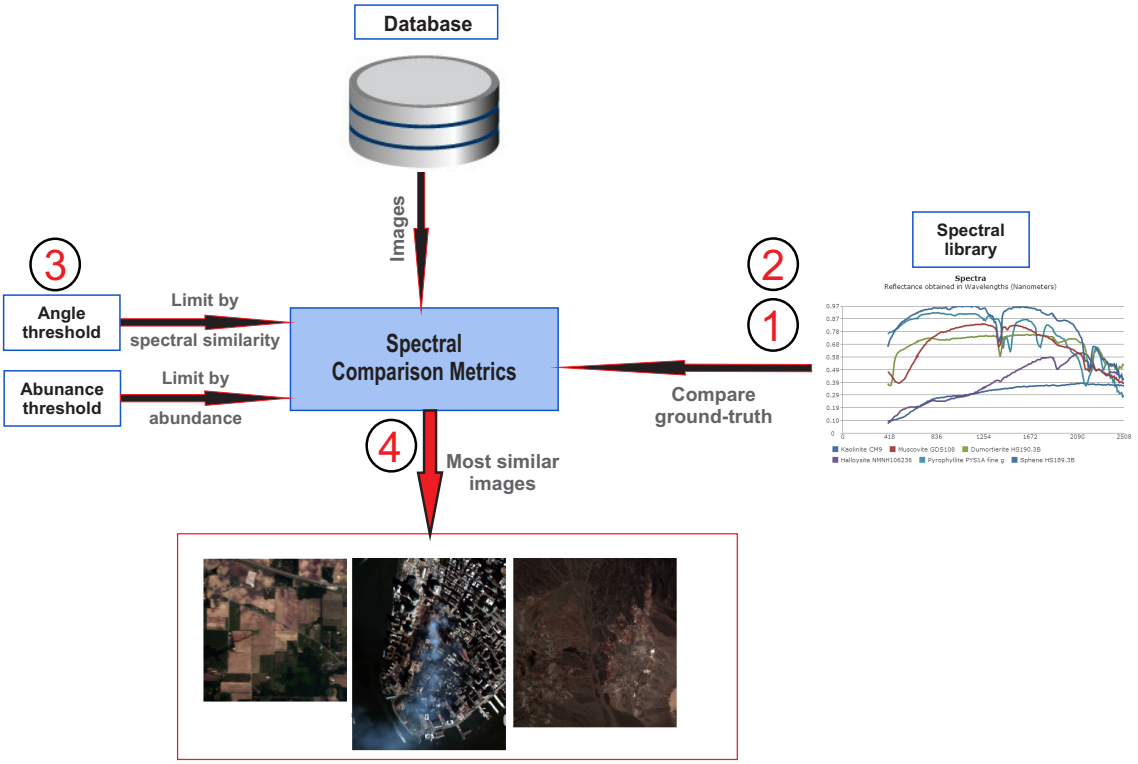


Figure 4.1: Search process work-flow for the sparse unmixing-based CBIR system.

the next work-flow: 1) uploading spectral library, 2) selecting ground-truth, 3) introducing minimum abundance and spectral angle threshold, 4) executing the query.

On the other hand, the *sparse-based* CBIR shows new features in the cataloguing process. Fig. 4.2 shows the different steps involved in the cataloguing procedure, which are enumerated from 1 to 5.

1. *Uploading spectral library.* The catalog procedure is fed by a spectral library, thus the system allows for uploading standard spectral libraries in the system.

2. *Selecting compute cluster.* Several computing resources are supported in the system, so we select the computing resources to perform the algorithm.

3. *Introducing the abundance threshold.* A minimum abundance limits the catalog results to the most relevant endmembers.

4. *Executing.* The algorithm execution returns a set of endmembers and their abundances.

5. *Endmember features storage.* Ultimately, the system stores the results in the database as meta-data associated to the processed image.

As we mentioned above, the catalog procedure is fed by a spectral library and the system allows to upload standard spectral libraries in the system using the same process adopted when using spectral libraries for searching [see Fig. 4.3 (a)]. Furthermore, as Fig. 4.3 (b) shows, the cataloguing interface consists of one algorithm (CSUNSAL) which extracts the endmembers and their abundance maps. Ultimately, Fig. 4.3(c) displays the sparsity map of an execution result, which was performed using

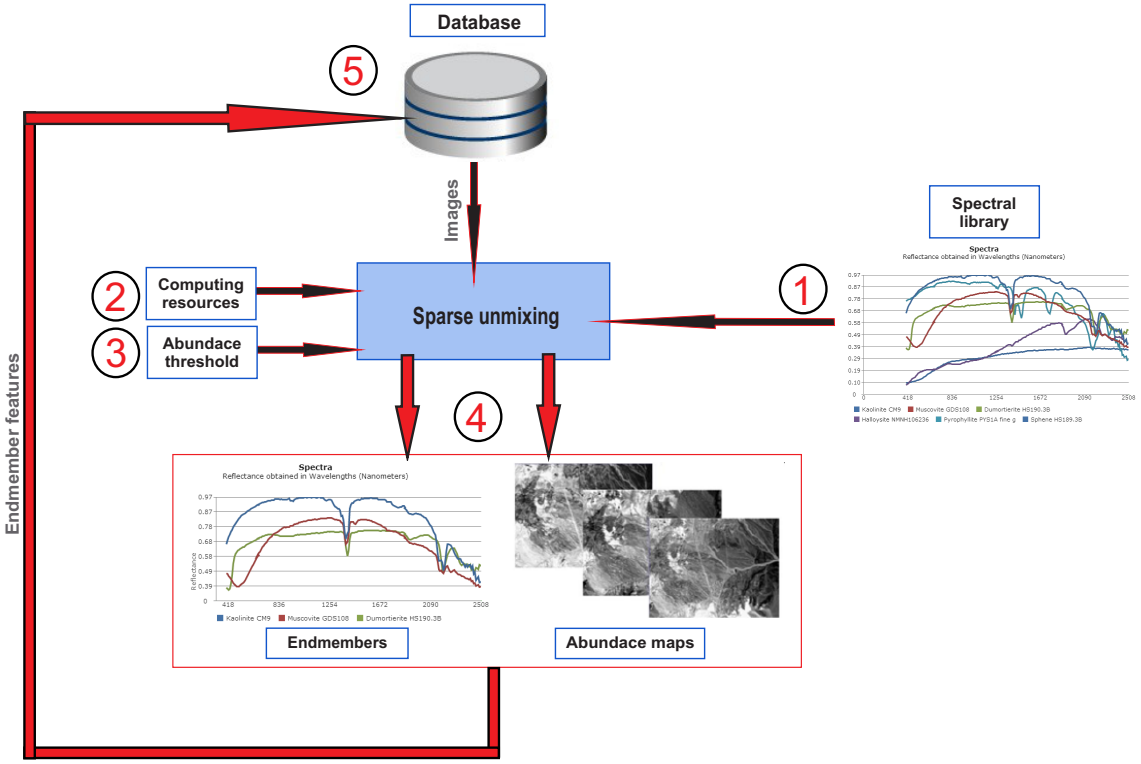


Figure 4.2: Catalog procedure work-flow.

the USGS spectral library over Cuprite image requiring 1 as the minimum abundance (threshold value). This sparsity map shows the 60 endmembers that satisfy such minimum abundance value.

4.3 Experiments and results

The performance of the proposed *sparse-based* CBIR system has been evaluated from two different perspectives: its ability to retrieve hyperspectral images of interest from the set of catalogued ones available in the system, and the efficiency in cataloguing new hyperspectral images that are stored in the repository. In addition, we compare the results of the new proposed CBIR with the *unmixing-based* CBIR approach. For this purpose, we use the full unmixing chain given by VD [75], OSP-GS [60] and ISRA [100] algorithms (described in the previous chapter) because we consider that this chain provides very accurate retrieval results in quite efficient processing times.

Experiments have been conducted using both synthetic images (described in section 3.6.1.1) and also the AVIRIS Cuprite real hyperspectral image (described in subsection 3.6.1.2). The AVIRIS Cuprite scene has several exposed minerals of interest, which are included in the USGS library considered in our experiments. The full spectral library of minerals from USGS (described in subsection 3.2.1) has been used to catalog every image considered in our experiments.

The remainder of the section is organized as follows. First, subsection 4.3.1 makes an evaluation of the system from the viewpoint of retrieval accuracy. Then, subsection 4.3.2 performs an evaluation from the viewpoint of computational efficiency.

Table 4.1: Spectral angle distance (degrees) obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The similarity results obtained for the six synthetic scenes (on average) with different noise ratios, and for the AVIRIS Cuprite scene are reported. The scene was catalogued using a spectral unmixing chain given by VD + OSP-GS + ISRA for the *unmixing-based CBIR*.

USGS	**Signal to noise ratio**						**AVIRIS**
signature	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**	**Cuprite**
Kaolinite CM9	12.345 ±0.001	1.293 ±0.001	0.158 ±0.001	0.104 ±0.001	0.104 ±0.001	0.104 ±0.001	3.731 ±0.001
Muscovite GDS108	18.843 ±0.001	1.685 ±0.001	0.192 ±0.001	0.117 ±0.001	0.115 ±0.001	0.115 ±0.001	4.082 ±0.001
Halloysite NMNH106236	24.203 ±0.001	2.842 ±0.001	0.602 ±0.001	0.568 ±0.001	0.571 ±0.001	0.570 ±0.001	18.625 ±0.001
Pyrophyllite PYS1A	15.568 ±0.001	1.682 ±0.001	0.352 ±0.001	0.015 ±0.001	0.000 ±0.001	0.000 ±0.001	9.154 ±0.001
Sphene HS189.3B	26.091 ±0.001	3.544 ±0.001	1.322 ±0.001	0.313 ±0.001	0.309 ±0.001	0.308 ±0.001	9.867 ±0.001
Alunite GDS83 Na	14.744 ±0.001	1.941 ±0.001	0.234 ±0.001	0.144 ±0.001	0.126 ±0.001	0.124 ±0.001	8.751 ±0.001
Nontronite GDS41	38.733 ±0.001	1.600 ±0.001	0.201 ±0.001	0.096 ±0.001	0.092 ±0.001	0.092 ±0.001	15.243 ±0.001
Dumortierite HS190.3B	15.732 ±0.001	4.467 ±0.001	1.994 ±0.001	0.501 ±0.001	0.502 ±0.001	0.502 ±0.001	3.334 ±0.001
KaolineiteKGa-l (wxyl)	17.434 ±0.001	27.964 ±0.001	0.182 ±0.001	0.0185 ±0.001	0.000 ±0.001	0.000 ±0.001	10.522 ±0.001

4.3.1 Evaluation of image retrieval accuracy

In order to illustrate the performance of the retrieval system, we specifically address a case study in which synthetic images are used to substantiate retrieval accuracy using different noise conditions. For illustrative purposes, we also use the AVIRIS Cuprite scene to evaluate the retrieval accuracy with real hyperspectral data. The spectral information has been extracted using both *sparse-based* and *unminxing-based* CBIR approaches. The two metrics that we have used in our experiments are the SAD in endmember comparison and the RMSE in the estimated abundance fractions [35], both of them are described in section 3.5.

Table 4.1 shows the SAD results over the endmembers achieved using the full unmixing chain (VD + OSP-GS + ISRA). On the other hand, all the SAD results obtained by the *sparse-based* CBIR approach using CSUNSAL with the USGS spectral library are really close to zero or zero, thus it is not necessary to show them in a table. The main reason for this behaviour is that the endmembers come from the same spectral library and the spectral bands are adapted to the common bands of both the image and the spectral library pixels by using Algorithm 2. Since the best case for SAD is 0 degrees, this is considered to be an optimal similarity score. Clearly, these results mean that the new approach has clearly better accuracy than the previous one using USGS spectral libraries.

On the other hand, we computed the RMSE between the ground-truth abundances and the estimated abundances by the considered unmixing chain and by the CSUNSAL algorithm. As we expected, Table 4.3 shows worse RMSE scores in the case of using sparse unmixing over the six synthetic images, because the endmembers come from spectral signatures that are taken from a laboratory measured spectral library

Table 4.2: Root mean reconstruction error (RMSE) in the reconstructed scene using endmember and abundances estimated over the AVIRIS Cuprite scene. The scene was catalogued using a spectral unmixing chain given by VD+OSP-GS+ISRA for *unmixing-based* approach and the CSUNSAL algorithm for the *sparse-based* approach with the full USGS spectral library

	Unmixing-Based	**Sparse-Based**
AVIRIS CUPRITE	0.190	0.332
	±0.001	±0.001

and then the achieved abundances are different to the ground-truth abundances of the synthetic scenes. The RMSE could not be obtained for the AVIRIS Cuprite scene since the ground-truth abundances are difficult to obtain in real scenarios, but in this case we used the RMSE between the original and the reconstructed scene using the endmembers and the abundances derived by the considered unmixing chain and by the CSUNSAL algorithm. As Table 4.2 shows, the RMSE results from the abundances obtained using CSUNSAL are worse than those obtained using the full unmixing chain because, as aforementioned, the endmember spectral signatures are taken from a laboratory measured spectral library, and then the reconstructed image is quite different to the original one. However, the reconstruction errors obtained in the AVIRIS Cuprite scene were very low [39] in all the cases, indicating that this strategy can also be used as a retrieval criterion.

On the other hand, the fractional abundance estimations which were obtained for all the signatures of the spectral library (481 minerals) by the proposed CSUNSAL algorithm over both the synthetic and the real AVIRIS Cuprite scenes are respectively displayed in Fig. 4.4 and Fig. 4.5. The abundance maps achieved over the six synthetic scenes are shown in Fig. 4.4, in which the maps show remarkable abundances in the spectral signatures present in the image. Although the results over the noisier scenes reveal as more inaccurate than in the rest, all the abundance maps clearly highlight the materials that compose the synthetic scenes. In addition, Fig. 4.5 shows the abundance map achieved over the AVIRIS Cuprite scene which shows clearly the predominant minerals in the image that are well-known from other studies such as [40].

4.3.2 Evaluation of parallel performance

The computational performance of the proposed CBIR system has been evaluated using both CPU and GPU architectures available in two different clusters. The serial algorithms were executed in one of the available cores of the *CETA Production* cluster with multi-cores Quad Core Intel Xeon at 2.93 GHz. On the other hand, the parallel algorithms were performed in the *INFRAGRID GPU* cluster with *NVidia*™ TESLA M2070Q GPU devices mounted in a system with two multi-cores Quad Core Intel Xeon at 3.46 GHz with 4 physical cores, and 32 GB of DDR3 SRAM memory. These clusters were widely described in section 2.3.4. Before describing our results, it is important to emphasize that our GPU implementations provide exactly the same results as the serial versions of the algorithms, implemented using the `gcc` (gnu compiler default) with optimization flag -O3.

In the following, we analyze the processing times (in seconds) used by our system in the process of cataloguing the synthetic and AVIRIS Cuprite scenes using the two approaches (based on the full unmixing chain and CSUNSAL algorithms). In each experiment, ten runs were performed and the mean values were reported. Table 4.4 shows the processing times of the unmixing chain based on VD for identification of the number of endmembers, OSP-GS for endmember signature finding, and ISRA for

Table 4.3: Root mean reconstruction error (RMSE) in abundance estimation obtained using 9 USGS mineral spectra (those used to construct the fractal synthetic scenes used in our experiments) as input to a query on our database of hyperspectral scenes. The accuracy of the abundance maps estimated from the six synthetic scenes (on average) with different noise ratios are reported. The scene was catalogued using a spectral unmixing chain given by VD + OSP-GS + ISRA for the *unmixing-based* approach and the CSUNSAL algorithm for the *sparse-based* approach

USGS	**CBIR**	**Signal to noise ratio**					
signature	**Approach**	**10:1**	**30:1**	**50:1**	**70:1**	**110:1**	**No noise**
Kaolinite CM9	**Sparse-based**	0.203 ±0.001	0.201 ±0.001	0.201 ±0.001	0.201 ±0.001	0.201 ±0.001	0.201 ±0.001
	Unmixing-based	0.147 ±0.001	0.089 ±0.001	0.082 ±0.012	0.083 ±0.012	0.083 ±0.012	0.076 ±0.001
Muscovite GDS108	**Sparse-based**	0.245 ±0.001	0.245 ±0.001	0.245 ±0.001	0.245 ±0.001	0.245 ±0.001	0.245 ±0.001
	Unmixing-based	0.570 ±0.001	0.484 ±0.001	0.409 ±0.104	0.406 ±0.104	0.406 ±0.001	0.406 ±0.001
Halloysite NMNH106236	**Sparse-based**	0.342 ±0.001	0.365 ±0.001	0.368 ±0.001	0.369 ±0.001	0.369 ±0.001	0.369 ±0.001
	Unmixing-based	0.063 ±0.001	0.057 ±0.001	0.099 ±0.026	0.095 ±0.026	0.094 ±0.026	0.080 ±0.001
Pyrophyllite PYS1A	**Sparse-based**	0.289 ±0.001	0.293 ±0.001	0.293 ±0.001	0.293 ±0.001	0.293 ±0.001	0.293 ±0.001
	Unmixing-based	0.086 ±0.001	0.080 ±0.001	0.071 ±0.013	0.071 ±0.013	0.071 ±0.013	0.063 ±0.001
Sphene HS189.3B	**Sparse-based**	0.313 ±0.001	0.310 ±0.001	0.242 ±0.001	0.282 ±0.001	0,275 ±0.001	0,275 ±0.001
	Unmixing-based	0.035 ±0.001	0.118 ±0.001	0.179 ±0.006	0.184 ±0.005	0.184 ±0.005	0.181 ±0.001
Alunite GDS83 Na	**Sparse-based**	0.230 ±0.001	0.264 ±0.001	0.269 ±0.001	0.268 ±0.001	0.268 ±0.001	0.269 ±0.001
	Unmixing-based	0.094 ±0.001	0.058 ±0.001	0.060 ±0.017	0.061 ±0.017	0.061 ±0.016	0.052 ±0.001
Nontronite GDS41	**Sparse-based**	0.232 ±0.001	0.249 ±0.001	0.274 ±0.001	0.272 ±0.001	0.264 ±0.001	0.272 ±0.001
	Unmixing-based	0.035 ±0.001	0.089 ±0.001	0.065 ±0.010	0.068 ±0.010	0.068 ±0.010	0.062 ±0.001
Dumortierite HS190.3B	**Sparse-based**	0.177 ±0.001	0.198 ±0.001	0.182 ±0.001	0.183 ±0.001	0.183 ±0.001	0.183 ±0.001
	Unmixing-based	0.193 ±0.001	0.064 ±0.001	0.075 ±0.016	0.075 ±0.016	0.075 ±0.016	0.065 ±0.001
KaolineiteKGa-l (wxyl)	**Sparse-based**	0.186 ±0.001	0.186 ±0.001	0.185 ±0.001	0.185 ±0.001	0.185 ±0.001	0.185 ±0.001
	Unmixing-based	0.260 ±0.001	0.063 ±0.001	0.074 ±0.012	0.076 ±0.011	0.076 ±0.011	0.067 ±0.001

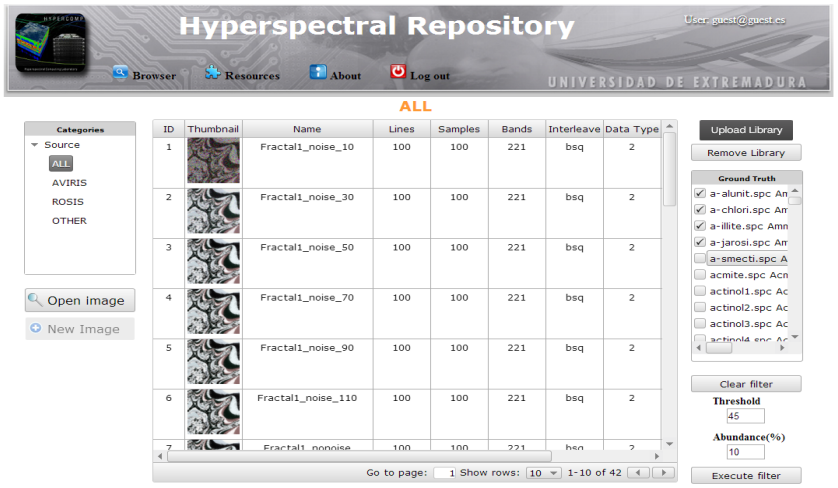
non-negative abundance estimation. Furthermore, Table 4.5 shows the timing results obtained by the CSUNSAL algorithm using the full USGS spectral library of minerals comprising 481 signatures. In both Table 4.4 and Table 4.5 we display the CPU and GPU times (including the initialization time required in the GPU cluster) and the speedup of the GPU version over the CPU one, furthermore the total times required to perform the full catalog process are included. The GPU version shows significant speedups, which mainly increase with the hyperspectral image size. Furthermore, since CSUNSAL is iteratively executed until reaching the best set of abundances, the times increase lightly with the noisier synthetic images.

Due to the very high speedups performed by the GPU version, we study the algorithm behaviour by using different numbers of spectral signatures, n. Table 4.6 shows the timing and speedups obtained by the CSUNSAL algorithm using $n = 10$, $n = 95$, $n = 200$, $n = 300$ and $n = 481$. In this way, Fig. 4.6 shows the speedup evolution in terms of the number of signatures used by the algorithm. In spite of the fact that we have observed that there are some peaks in the GPU times (related to the diversity of the spectral signatures), the times and speedups mainly grow with the number of spectral signatures. The high increase of the speedups with the number of signatures has been already analyzed in other studies such as [103].

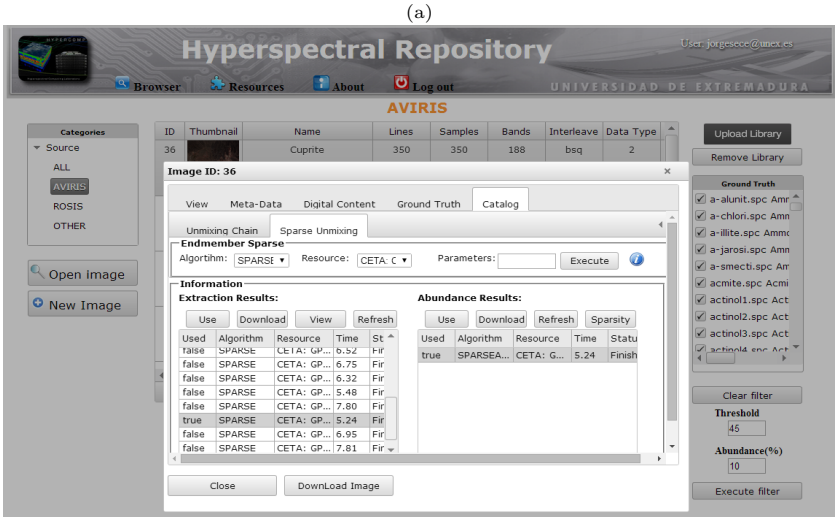
Although Table 4.4 and Table 4.5 demonstrate that CSUNSAL algorithm is slower than the full unmixing chain, we consider that the *sparse-based* approach is faster for cataloguing images since it can obtain all the needed information relevant to a query in just one execution, while the *unmixing-based* approach needs to execute several algorithms (which implies several downloading/uploading transactions).

Table 4.4: Processing times (in seconds) and speedups achieved for the GPU implementation of VD, OSP-GS and ISRA algorithms. The results obtained for the six synthetic and the real AVIRIS Cuprite scenes (on average) are reported. The table reports the mean values and the standard deviations measured across ten algorithm executions.

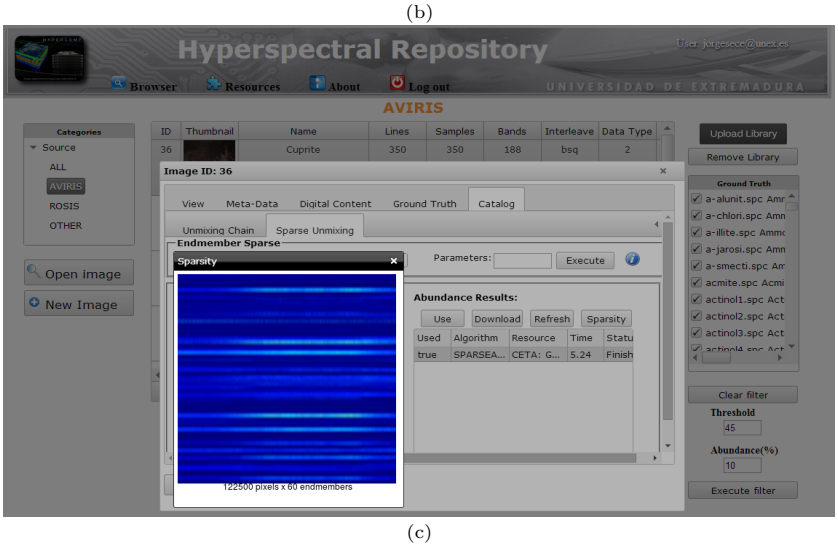
Scenes		**VD**			**OSP-GS**			**ISRA**			**Total**
		Initialization	**VD**	**Total**	**Initialization**	**OSP-GS**	**Total**	**Initialization**	**ISRA**	**Total**	
AVIRIS Cuprite	**CPU**	0.340	14,476	14.816	0.198	2.225	2.423	0.149	120.650	120.799	**138.0.38**
		±0.158	±0.921	±0.922	±0.015	±0.016	±0.022	±0.021	±0.335	±0.355	-
	GPU	0.185	0.324	0.509	0.138	0.045	0.183	0.173	2.343	2.516	**3.207**
		±0.089	±0.004	±0.099	± 0.092	±0.003	±0.092	±0.120	±0.005	±0.089	-
	Speedup	-	44.714	29.108	-	49.955	13.240	-	51.485	48.123	**43.042**
Synthetic SNR 10:1	**CPU**	0,012	1,170	1,182	0.080	0.050	0.130	0.081	24.653	24.734	**26.046**
		±0.001	±0.011	±0.011	± 0.001	±0.001	±0.001	±0.001	±0.021	±0.021	-
	GPU	0.065	0,087	0.152	0.070	0.004	0.074	0.077	0.267	0.344	**0.570**
		±0.004	±0.001	±0.00	± 0.022	±0.001	0.021	±0.006	±0.004	±0.006	-
	Speedup	-	13.431	7.788	-	12.390	1.752	-	92.381	71.984	**45.726**
Synthetic SNR 30:1	**CPU**	0,053	1,592	1,645	0.090	0.050	0.140	0.057	26.176	26.233	**28.019**
		±0.001	±0.012	±0.012	± 0.001	±0.001	±0.001	±0.001	±0.022	±0.022	-
	GPU	0,064	0,087	0,150	0.079	0.004	0.083	0.076	0.269	0.344	**0.578**
		±0.002	±0.001	±0.001	± 0.003	±0.001	0.002	±0.006	±0.004	0.005	-
	Speedup	-	18,407	10.940	-	11.745	1.692	-	97.485	76.209	**48.507**
Synthetic SNR 50:1	**CPU**	0,044	1,160	1,203	0.059	0.005	0.109	0.119	26.201	26.320	**27.632**
		±0.001	±0.019	±0.019	± 0.001	±0.001	±0.001	±0.001	±0.021	±0.022	-
	GPU	0,063	0,087	0,150	0.077	0.004	0.081	0.076	0.267	0.343	**0.573**
		±0.001	±0.001	±0.001	± 0.002	±0.001	±0.002	±0.005	±0.004	0.005	-
	Speedup	-	13,362	8.017	-	12.136	1.346	-	98.109	76.814	**48.189**
Synthetic SNR 70:1	**CPU**	0.060	1.163	1.223	0.080	0.050	0.130	0.102	16.195	26.297	**27.649**
		±0.001	±0.015	±0.015	± 0.001	±0.001	±0.001	±0.001	±0.020	±0.021	-
	GPU	0.063	0.086	0.149	0.078	0.004	0.082	0.076	0.269	0.345	**0.576**
		±0.002	±0.001	±0.001	± 0.003	±0.001	±0.003	±0.005	±0.022	±0.021	-
	Speedup	-	13,523	8.235	-	12.019	1.590	-	97.285	76.167	**48.041**
Synthetic SNR 110:1	**CPU**	0.044	1.158	1.202	0.061	0.050	0.111	0.079	26.190	26.269	**27.581**
		±0.001	±0.014	±0.014	± 0.001	±0.001	±0.001	±0.001	±0.001	-	-
	GPU	0.063	0.087	0.149	0.076	0.004	0.080	0.078	0.269	0.347	**0.654**
		±0.001	±0.001	±0.001	± 0.002	±0.001	±0.002	±0.007	±0.021	±0.021	-
	Speedup	-	13.309	8.033	-	12.677	1.380	-	97.520	75.777	**47.845**
Synthetic No noise	**CPU**	0.049	1.109	1.157	0.088	0.050	0.138	0.066	26.195	26.261	**27.656**
		±0.001	±0.017	±0.017	± 0.001	±0.001	±0.001	±0.001	±0.001	-	-
	GPU	0.063	0.090	0.153	0.077	0.004	0.081	0.079	0.268	0.347	**0.581**
		±0.004	±0.002	±0.002	±0.002	±0.001	±0.002	±0.010	±0.022	±0.022	-
	Speedup	-	12,279	7.550	-	12.157	1.709	-	97.634	75.658	**47.418**



(a)



(b)



(c)

Figure 4.3: Different operations with the sparse unmixing-based CBIR system. (a)Spectral library uploading. (b) Sparse catalog panel on the system. (c) Sparsity results.

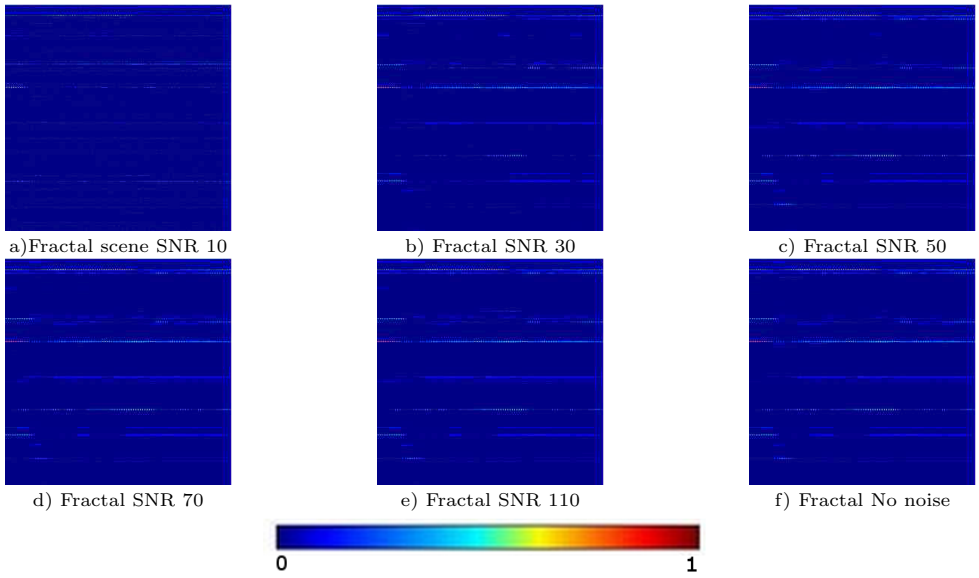


Figure 4.4: Abundance maps achieved over the synthetic fractal scenes using the CSUNSAL algorithm and the 481 mineral signatures from the USGS spectral library of minerals.

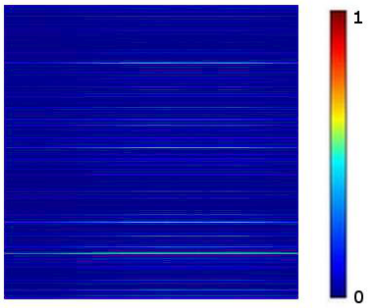


Figure 4.5: Abundance map achieved over the AVIRIS Cuprite scene using the CSUNSAL algorithm and the 481 mineral signatures from the USGS spectral library of minerals.

Table 4.5: Processing times (in seconds) and speedups achieved for the GPU implementation of CSUNSAL algorithm. The results obtained for the six synthetic and the real AVIRIS Cuprite scenes (on average) are reported. The table reports the mean values and the standard deviations measured across ten algorithm executions.

		Initialization	**CSUNSAL**	**Total**
AVIRIS Cuprite	**CPU**	0.460 ±0.008	77447.910 ±1642.342	**77448.370** ±1642.342
	GPU	0.517 ±0.009	10.382 ±0.059	**10.899** ±0.059
	Speedup	-	7459.826	**7106.007**
Synthetic SNR 10:1	**CPU**	0.021 ±0.003	1865.035 ±142.997	**1865.056** ±142.997
	GPU	0.105 ±0.005	9.924 ±0.024	**10.029** ±0.024
	Speedup	-	187.932	**185.966**
Synthetic SNR 30:1	**CPU**	0.017 ±0.006	1929.858 ±181.363	**1929.875** ±181.363
	GPU	0.103 ±0.012	6.132 ±0.035	**6.235** ±0.035
	Speedup	-	314.719	**309.523**
Synthetic SNR 50:1	**CPU**	0.015 ±0.005	1799.920 ±115.219	**1799.935** ±115.219
	GPU	0.103 ±0.007	5.434 ±0.034	**5.537** ±0.034
	Speedup	-	331.233	**325.074**
Synthetic SNR 70:1	**CPU**	0.016 ±0.007	1897.033 ±165.876	**1897.305** ±165.876
	GPU	0.107 ±0.006	5.452 ±0.036	**5.559** ±0.036
	Speedup	-	347.952	**341.257**
Synthetic SNR 110:1	**CPU**	0.017 ±0.005	1876.436 ±177.225	**1876.453** -
	GPU	0.103 ±0.006	5.352 ±0.031	**5.455** -
	Speedup	-	350.605	**343.988**
Synthetic No noise	**CPU**	0.012 ±0.004	1958.481 ±196.183	**1958.493** ±196.183
	GPU	0.106 ±0.008	5.456 ±0.015	**5.562** ±0.015
	Speedup	-	358.959	**352.120**

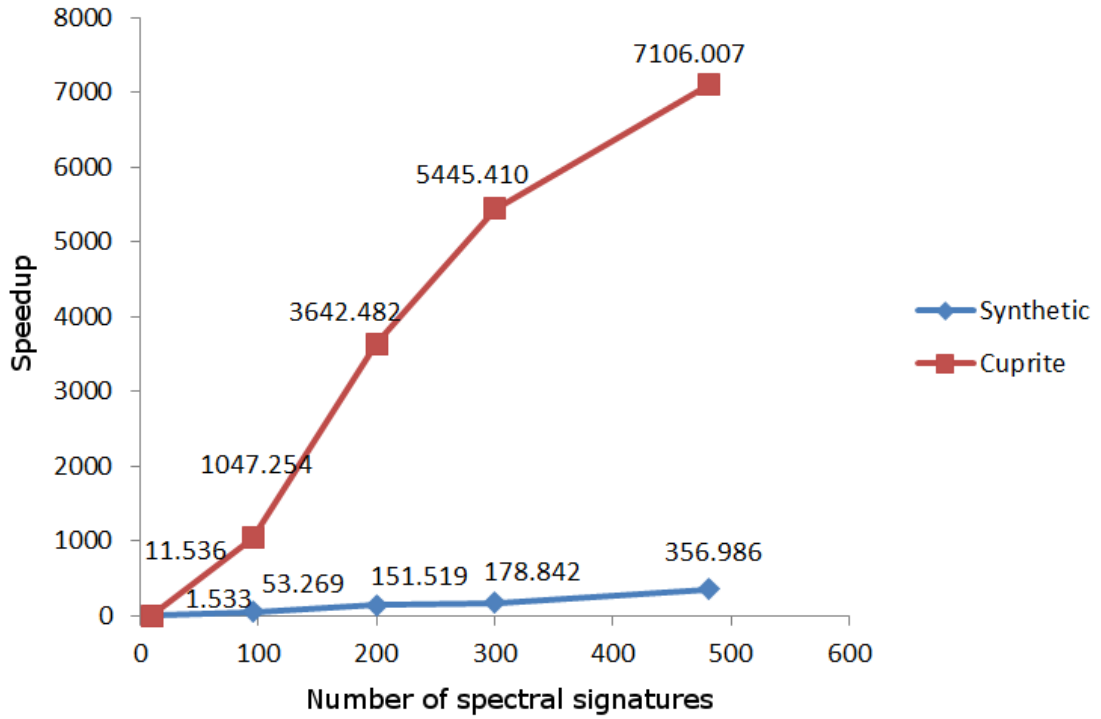


Figure 4.6: Speedup of parallel version as a function of the number of spectral signatures(n).

Table 4.6: A comparison of the processing times (in seconds) and speedups achieved for the GPU implementation of CSUNSAL algorithm using different number of spectral signatures. The results obtained for the no-noise synthetic and the real AVIRIS Cuprite scenes (on average) are reported. The table reports the mean values measured across ten algorithm executions.

Spectral Signatures		Synthetic scene			AVIRIS Cuprite scene		
		Initialization	**CSUNSAL**	**Total**	**Initialization**	**CSUNSAL**	**Total**
$n = 10$	**CPU**	0.020	0.840	**0.860**	0.271	27.163	**27.503**
	GPU	0.078	0.483	**0.561**	0.340	2.113	**2.384**
	Speedup	-	1.739	**1.533**	-	12.855	**11.536**
$n = 80$	**CPU**	0.023	52.127	**52.150**	0.302	2500.540	**2500.842**
	GPU	0.084	0.895	**0.979**	0.356	2.032	**1047.254**
	Speedup	-	58.242	**53.269**	-	1230.581	**1047.254**
$n = 200$	**CPU**	0.030	317.553	**312.583**	0.342	14871.910	**14872.252**
	GPU	0.092	2.004	**2.096**	0.396	3.687	**4.083**
	Speedup	-	317.553	**151.519**	-	4033.607	**3642.482**
$n = 300$	**CPU**	0.042	764.865	**764.907**	0.384	32024.070	**32024.454**
	GPU	0.101	4.176	**4.277**	0.432	5.449	**5.881**
	Speedup	-	183.157	**178.842**	-	5877.055	**5445.410**
$n = 481$	**CPU**	0.062	1985.492	**1985.554**	0.460	77447.910	**77448.370**
	GPU	0.106	5.456	**5.562**	0.517	10.582	**10.899**
	Speedup	-	358.959	**356.986**	-	7459.826	**7106.007**

Chapter 5

Multispectral product repository

A large number of remote sensing data sets have been collected in recent years by Earth observation instruments such as the MODIS sensor aboard the Terra/Aqua satellite, or the SEVIRI sensor aboard the geostationary platform MSG. The advanced remote sensing products resulting from the analysis of these data are useful in a wide variety of applications, but require significant resources in terms of storage, retrieval and analysis. Despite the wide availability of these MODIS/SEVIRI products, the data coming from these instruments are spread among different locations and retrieved from different sources, and there is no common data repository from which the data or the associated products can be retrieved. In this chapter, we take a first step towards the development of a geo-portal [104] for storing and efficiently retrieving MODIS/SEVIRI remote sensing products. The products are obtained using an automatic system that processes the data as soon as they are provided by the collecting antennas, and then the final products are uploaded one day delay in the geo-portal. Our focus in this work is on describing the design and efficient implementation of the geo-portal, which allows for a user-friendly and effective access to a full repository of MODIS/SEVIRI advanced products (comprising hundreds of terabytes of data) using geo-location retrieval capabilities over the Southwestern Europe area. The geo-portal, which is available online at http://ceosspain.lpi.uv.es, has been implemented as a web application composed of different layers. Its modular design provides quality of service and scalability (capacity for growth without any quality losing), allowing for the addition of components without the need to modify the entire system. On the client layer, an intuitive web browser interface provides users with remote access to the system. On the server layer, the system provides advanced data management and storage capabilities. On the storage layer, the system provides a secure massive storage service. An experimental evaluation of the geo-portal in terms of efficiency and product retrieval accuracy is also presented and discussed.

The remainder of this chapter is organized as follows. Section 5.1 introduces the main objectives of this chapter. Section 5.2 describes the processing chain used to generate the advanced MODIS/SEVIRI products that are distributed through the geo-portal. This chain is applied as soon as the data are collected, and the system automatically stores the resulting products in the geo-portal. Section 5.3 describes the implementation of the geo-portal, which is composed of three main layers: 1) client layer, which defines the interaction between the user and the system through a web interface; 2) server layer, which efficiently manages the requests coming from end-users; and 3) storage layer, in charge of safe storage and retrieval of remote sensing products. Section 5.4 presents an experimental validation of the system, with particular attention to its geolocation retrieval accuracy and to the performance of the system in responding to the queries carried out by end-users.

5.1 Main objectives

In the last years, a significant amount of data from satellite Earth Observation instruments has been made available to the public. Among these, the MODIS and the SEVIRI sensors improve on their predecessors. MODIS is a key instrument aboard the Terra/Aqua satellite. Terra's orbit around the Earth is timed so that it passes from North to South across the Equator around 10:30, while Aqua passes South to North over the Equator around 13.30. Global coverage of the Earth is provided by Terra/Aqua every one to two days. On the other hand, the SEVIRI instrument on board the geostationary platform MSG [54] provides (every 15 minutes) a low resolution scan of the Europe, Africa, the Middle East and the eastern tip of South America. The images have a sub-nadir resolution of 3 km that deteriorates towards the poles, reducing to an average of 5 by 3 km^2 over Europe.

In recent years, many research studies have been focused on the exploitation of data coming from the aforementioned instruments. For instance, in [58] an operational algorithm for retrieving the LST from SEVIRI data is provided. In [66], a technique for estimating the SST from SEVIRI data is given. A similar strategy is developed in [67, 59] for MODIS data, while [105] assessed the presence of trends in MODIS time series. In [106], a procedure for water vapor retrieval from SEVIRI observations is presented. Numerous classic studies, such as [56, 50], have also focused on the retrieval of advanced vegetation parameters from remote sensing data such as NDVI or FVC, with the ultimate goal of providing advanced products covering relatively large areas [107]. Several other recent applications have exploited MODIS data in a diversity of areas, such as estimation of vegetation phenology [108], mapping of snow cover [109], characterization of soil moisture [110], mapping of impervious cover [111], agriculture monitoring [112], analysis of LST and environmental factors [113], deriving water fraction and flood maps [114], or fire characterization [115], among many others [116, 117, 118]. SEVIRI data has also been recently exploited for phenology estimation [119], detection of tropical cyclones [120], monitoring aerosols [121], or estimating land surface radiation and energy budgets from ground measurements [122]. These studies are possible thanks to the availability of a wide range of advanced remote sensing products from the data collected by those instruments.

There are several institutions currently distributing multispectral sensors products. The most important institutions are NASA and USGS. On one hand, the NASA's EOSDIS system provides end-to-end capabilities for managing NASA's Earth science data from various sources (satellites, aircraft, field measurements and various other programs) in which the data is distributed through the *LAADS Web*[45] system. This system provides quick and easy access to MODIS Level 1, Atmosphere and Land data products and VIIRS sensor Level 1 and Land data products. On the other hand, LPDAAC from USGS distributes mainly land products through several web systems, from which two of them are focused on MODIS products. First, the *USGS EE* [49] tool offers data from the Landsat missions and a variety of other data providers, this tool now provides access to MODIS land data products (from the NASA Terra and Aqua missions) and to ASTER level-1B data products over the United States territories (from the NASA ASTER mission). The second web system of USGS is *MRTWeb 2.0*[44], which is a more complete data system that combines the interface of USGS Global Visualization Viewer [123] and the product database of MRT [124]. In our work, we propose a strong standardized repository by combining two very different resolution sensors: MODIS/SEVIRI, which provide images and products with different formats.

The Calibration of Earth Observation Satellites in Spain (CEOS-SPAIN) project has recently developed a full operational chain for advanced retrieval and processing of MODIS and SEVIRI data acquired in real-time at the UCG/IPL[71], University of Valencia (see Fig. 5.1). These final products are

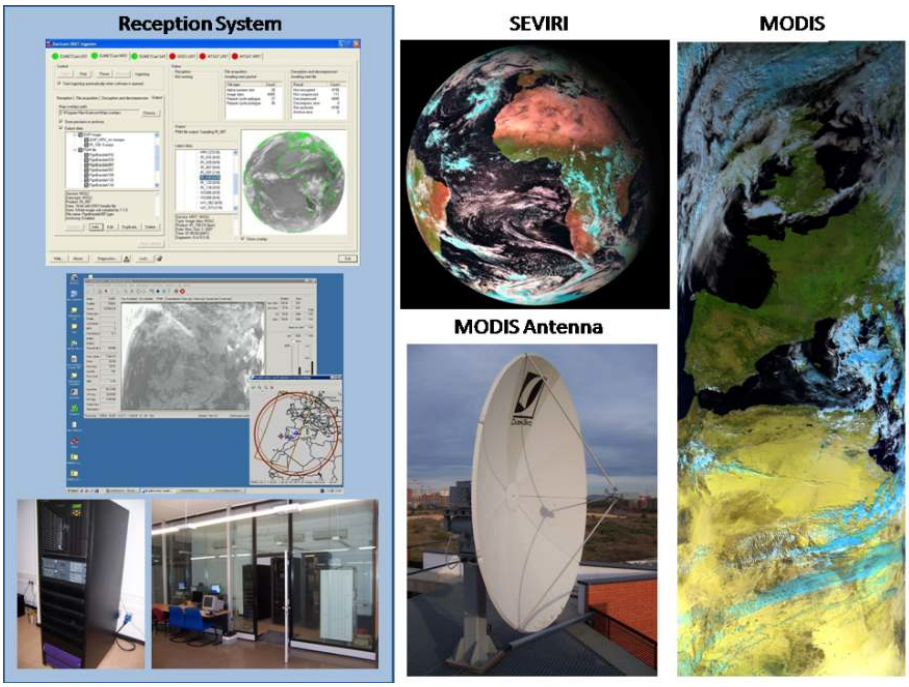


Figure 5.1: MODIS (Terra/Aqua) and SEVIRI (Meteosat Second Generation) real-time reception system at the Global Change Unit (UCG) within the Image Processing Laboratory (IPL) of the University of Valencia.

validated (quality control) using the on-ground data acquired in a variety of test sites. The UCG/IPL participated actively in the definition of products to be delivered to users from MODIS and SEVIRI data. The resulting remote sensing products are of great interest for the wide remote sensing community. These MODIS products cover Southwestern Europe and most of Northwestern Africa while SEVIRI products cover half the surface of the Earth. However, the wide availability of these products is difficult due to the massive volume of information that they comprise. In the case of MODIS, the amount of data acquired and processed by UCG/IPL from July 2010 to February 2014 exceeds eight thousand images and two hundred thousand resulting products (more than nine terabytes of data). In the case of SEVIRI, the amount of data acquired and processed by UCG/IPL from July 2007 to February 2014 comprises more than two hundred thousand images and four hundred thousand resulting products (more than seventeen terabytes of data). This massive amount of information can be extremely useful in climate change studies and in many other studies, specially if the products could be easily retrieved for a given area of interest. In fact, this capability has the potential to impact the activities of many remote sensing users such as farmers, engineers, water managers, as well as the scientific community.

In this chapter, we describe a geo-portal that has been specifically designed to provide efficient access to a large collection of products of MODIS/SEVIRI image acquisitions, which are generated every day at the facilities of UCG/IPL. This work is the first to be published on the reception system at IPL. The geo-portal has been designed in the framework of the CEOS-SPAIN project, which ultimately pursues the development of an integrated system for optical data validation and information extraction, expected to serve as a reference to the remote sensing community through the heterogeneous data distribution and dissemination mechanisms planned for the project. The geo-portal has been designed using an advanced computing infrastructure to speed-up the supported data distribution and processing

tasks. The geo-portal works as a digital repository which has been designed in order to store and efficiently retrieve advanced MODIS/SEVIRI products. Of particular importance are the geolocation-based retrieval functionalities included in the geo-portal, which facilitates searching for remote sensing scenes (in geolocated form) over a map using Google Maps[1] services. Although the geo-portal is focused on Southwestern Europe, it is designed to store images from any location. Another feature of our geo-portal is that it provides advanced products in near real-time, which may be useful to end-users without the possibility to perform advanced processing of the collected remote sensing data. In fact, the idea of using the final products instead of the raw data is quite appealing from the viewpoint of the end-user, who can simply retrieve the processed products after applying different types of indices and perform a more intuitive exploitation of the data.

The proposed geo-portal has been implemented as a web application (available online: http://ceosspain.lpi.uv.es) which is composed of different layers. Its modular design provides quality of service and scalability, and allows adding and/or modifying components without the need to modify the entire system. These properties result from the fact that the geo-portal has been developed as a digital repository, using advanced software and hardware design techniques with the most advanced tools for such purposes currently available. On the client layer, an intuitive web browser interface provides to users with remote access to the system. On the server layer, the system provides advanced data management capabilities. On the storage layer, the system provides a safe massive storage service handling several terabytes of data. Combined, these parts provide (for the first time in the literature) a completely open standardized repository of (easily searchable) MODIS/SEVIRI products that is expected to be useful in a variety of remote sensing data exploitation tasks, as originally intended by the CEOS-SPAIN initiative.

5.2 Processing chain

We have implemented a processing chain for MODIS (Terra/Aqua) and SEVIRI (MSG) data acquired in real-time with the antennas located at the UCG/IPL of the University of Valencia. SEVIRI data encompasses the hemisphere centered on the $(0, 0)$ latitude/longitude coordinates, while acquired MODIS data consist of Aqua and Terra overpasses within the reach of the antenna located at the IPL, which covers mostly Western Europe and Maghreb. These data allow for the generation of classic and advanced remote sensing products for inter-comparison with other sensor products. The proposed chain has been designed for processing image acquisitions every day. The chain is implemented in the interactive data language[2]. This section describes succinctly the process carried out over MODIS (subsection 5.2.2), and then SEVIRI (subsection 5.2.3) data. Both cases need a previous pre-processing step which is described in subsection 5.2.1.

5.2.1 Pre-processing

For both MODIS and SEVIRI sensors, images are acquired as digital counts, which are converted to radiances using the gain and offset values provided in the meta-data associated to the images. Then, the day-time visible and near infra-red channels of the data are atmospherically corrected using SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum [64]. The choice of standard input values for atmospheric correction with SMAC software results of a compromise between atmospheric correction accuracy and instantaneous processing of the received data.

[1] https://www.google.com/maps
[2] http://www.exelisvis.com/ProductsServices/IDL.aspx

By using standard values for atmosphere type (continental), atmospheric pressure (1013 hPa), aerosol optical thickness (0.05) and ozone concentration (0.33 atm.cm), the atmospherically corrected values may be slightly over or underestimated, although the resulting error is generally lower than the error resulting from the use of the top of atmosphere values. This method runs on a pixel by pixel basis, and needs additional information such as (standard values in parentheses): atmosphere type (continental), atmospheric pressure (1013 hPa), aerosol optical thickness (0.05) and ozone concentration (0.33 atm.cm). Additionally, a water vapor product (see below) is needed as an input of the SMAC software. Finally, for thermal infra-red channels, brightness temperatures are estimated by inverting Planck's law:

$$\mathbf{T}_i = \frac{\frac{\mathbf{c}_2}{\lambda_{eff}}}{ln\left(\frac{\mathbf{c}_1}{{\lambda_{eff}}^5 \cdot \mathbf{B}(\mathbf{T}_i)} + 1\right)}, \tag{5.1}$$

where $\mathbf{T}_i$ is the brightness temperature in each thermal band, $\mathbf{c}_1 = 1.1491047 \cdot 10^8 W/(m^2 sr \mu m^{-4})$, $\mathbf{c}_2 = 1.4387752 \cdot 10^4 K \cdot \mu m$, $\mathbf{B}(\mathbf{T}_i)$ is the at sensor radiance, expressed in $W/(m^2 sr \mu m)$, and λ_{eff} is the effective wavelength for each band i.

5.2.2 MODIS products

The complete set of products available in our system is described hereafter. Ancillary data include Latitude, Longitude, Elevation (Height), Sensor Azimuth Angle, Sensor Zenith Angle, Solar Azimuth Angle, Solar Zenith Angle. Several masks can also be downloaded: LSM, day/night mask (Day), cloud mask (Clouds), fire mask (Fire), and snow mask (Snow). These masks have been calculated with MOD35 and MOD14 software[3]. Advanced products, including VCI [74] and BRDF [48] corrected NDVI and reflectances for bands 1 to 7, will be included in a close future. Besides, other products are available in our system:

- NDVI [56] is traditionally calculated as $NDVI = (NIR - RED)/(NIR + RED)$, where NIR stands for reflectivity in Near Infra-Red, and RED for reflectivity in red channels, which for MODIS correspond, respectively, to MODIS bands 2 and 1:

  $$NDVI = \frac{\rho_2 - \rho_1}{\rho_2 + \rho_1}, \tag{5.2}$$

  where ρ_1 and ρ_2 are the at-surface reflectivities obtained from sensor bands located in RED and NIR spectral regions.

- FVC is estimated from the NDVI following [50] for day-time acquisitions, as a normalization of NDVI between standard bare soil and dense vegetation values. In the case of MODIS, these values are 0.15 and 0.90, respectively [107]:

  $$FVC = \frac{NDVI - 0.15}{0.90 - 0.15} \tag{5.3}$$

- Emissivities are estimated for day-time acquisitions from FVC and MODIS band 1 information, following the methodology presented in [59]. These emissivities correspond to MODIS thermal bands 31 and 32, and are estimated differently depending on the vegetation proportion within a given pixel. In the case of night-time acquisitions, such method cannot be implemented due

[3] http://eostation.scanex.ru/software.html

Table 5.1: MODIS emissivity values

	ε	$\mathbf{\Delta}\varepsilon$
vegetation: $NDVI > 0.5$	0.99	0
Mixed: $0.2 \leq NDVI \leq 0.5$	$0.971 + 0.018FVC$	$0.006(1 - FVC)$
Bare soil: $NDVI < 0.2$	$0.9832 - 0.058\rho_1$	$0.0018 - 0.060\rho_1$

to the lack of solar radiation; therefore the emissivity estimates during the previous day are re-projected to a latitude/longitude grid, averaged and re-projected back to the night-time acquisition configuration for further calculations. Due to computational constrains, the re-projection algorithm is limited to the Iberian Peninsula and North West Africa. Therefore, night products are also limited to this area. As Table 5.1 shows, the emissivities are expressed as average emissivities ε (for bands 31 and 32) and spectral difference of emissivities $\mathbf{\Delta}\varepsilon$.

- Water vapor is then estimated using the method developed in [59]. This method is based on the attenuation of surface reflected solar radiation and clouds in near infra-red due to water vapor. To this end, water vapor absorbing bands centred at 0.905, 0.936 and 0.94μm (bands 17, 18 and 19) are used in addition to a water vapor transparent band centred at 0.865μm (band 2) as follows:

$$\mathbf{W} = 0.192 \cdot \mathbf{W}_{17} + 0.453 \cdot \mathbf{W}_{18} + 0.355 \cdot \mathbf{W}_{19} \tag{5.4}$$

where:

$$\mathbf{W}_{17} = 28.449 \cdot {\mathbf{G}_{17}}^2 - 54.434 \cdot \mathbf{G}_{17} + 26.314,$$
$$\mathbf{W}_{18} = 27.884 \cdot {\mathbf{G}_{18}}^2 - 23.017 \cdot \mathbf{G}_{18} + 5.012,$$
$$\mathbf{W}_{19} = 19.914 \cdot {\mathbf{G}_{19}}^2 - 26.887 \cdot \mathbf{G}_{19} + 9.446,$$

with:

$$\mathbf{G}_{17} = \mathbf{L}_{17}/\mathbf{L}_2,$$
$$\mathbf{G}_{18} = \mathbf{L}_{18}/\mathbf{L}_2,$$
$$\mathbf{G}_{19} = \mathbf{L}_{19}/\mathbf{L}_2,$$

where $\mathbf{W}$ is the atmospheric water vapor ($g \cdot cm^{-1}$), and $\mathbf{L}_2$, $\mathbf{L}_{17}$, $\mathbf{L}_{18}$ and $\mathbf{L}_{19}$ are the radiance values (RAD-TOA) of MODIS in 2, 17, 18 and 19 bands respectively. In the case of night-time acquisitions, such method cannot be implemented due to the lack of solar radiation, therefore the water vapor estimates during the previous day are re-projected to a latitude/longitude grid, averaged and re-projected back to the night-time acquisition configuration for further calculations. Even though different water vapor values are estimated for any given day, only one water vapor value (daylight average of valid values) is considered per day. Assuming a constant water vapor during 24 hour may lead to over and under-estimations of this parameter, resulting in small differences in final LST. A sensibility analysis shows that an error of $1g \cdot cm^2$ in water vapor leads to a 0.23 K error in LST.

- Land surface temperature (LST) is estimated for day- and night-time acquisitions using the method developed in [59]:

$$LST = \mathbf{T}_{31} + \mathbf{a}_1 + \mathbf{a}_2(\mathbf{T}_{31} - \mathbf{T}_{32}) + \mathbf{a}_3(\mathbf{T}_{31} - \mathbf{T}_{32}) + (\mathbf{a}_4 + \mathbf{a}_5 \cdot \mathbf{W})(1 - \varepsilon) + (\mathbf{a}_6 + \mathbf{a}_7 \cdot \mathbf{W})\delta\varepsilon, \tag{5.5}$$

Table 5.2: SEVIRI emissivity values

	ε_{108}	ε_{120}
vegetation: $NDVI > 0.5$	0.99	0.99
Mixed: $0.2 \leq NDVI \leq 0.5$	$0.968 + 0.021 FVC_{seviri}$	$0.976 + 0.015 FVC_{seviri}$
Bare soil: $NDVI < 0.2$	$0.977 - 0.048 Vis06$	$0.981 - 0.026 Vis06$

where $\mathbf{T}_{31}$ and $\mathbf{T}_{32}$ are brightness temperature for MODIS bands 31 and 32 respectively, ε and $\delta\varepsilon$ are respectively the average emissivity and the spectral emissivity difference for these bands, $\mathbf{W}$ is the total amount of water vapor estimated above, and $\mathbf{a}_1 = 1.02, \mathbf{a}_2 = 1.79, \mathbf{a}_3 = 1.20, \mathbf{a}_4 = 34.83, \mathbf{a}_5 = -0.68, \mathbf{a}_6 = -73.27$ and $\mathbf{a}_7 = -5.19$.

- Sea surface temperature (SST) [125] is estimated for day- and night-time acquisitions using:

$$SST = \mathbf{T}_{31} + (\mathbf{a}_1 + \mathbf{a}_2 \cdot \mathbf{W})(\mathbf{T}_{31} - \mathbf{T}_{32}) + \mathbf{a}_3 \cdot \mathbf{W} + \mathbf{a}_4, \tag{5.6}$$

where $\mathbf{T}_{31}$ and $\mathbf{T}_{32}$ are brightness temperature for MODIS bands 31 and 32 respectively, $\mathbf{W}$ is the total amount of water vapor estimated above, and $\mathbf{a}_1 = 1.90, \mathbf{a}_2 = 0.44, \mathbf{a}_3 = 0.05$ and $\mathbf{a}_4 = 0.34$.

5.2.3 SEVIRI products

The SEVIRI products available in our processing chain can be summarized as follows:

- NDVI is retrieved as in the case of MODIS data, by using red (visible band at 0.6 microns) and near-infra-red (visible band at 0.8 microns) information.
- Emissivities are estimated for day-time acquisitions from NDVI and red information at 0.6 microns ($Vis06$), following the threshold method. These emissivities correspond to SEVIRI infrared $\mathbf{T}_{108}$ and $\mathbf{T}_{120}$ spectral bands centred at 10.8, and 12.0 μm, and are estimated differently depending on the vegetation proportion within a given pixel. These emissivities are expressed as ε_{108} and ε_{120}. Table 2 shows the coefficients for emissivity estimation for different NDVI ranges of value:
where $FVC_{seviri} = \dfrac{(NDVI - 0.2)^2}{0.009}$.
- Water vapor is then estimated using the method developed by [59]. This method is based on the difference in temperature between two different acquisitions ($\mathbf{T}^A$ and $\mathbf{T}^B$) for both infra-red channels ($\mathbf{T}_{108}$ and $\mathbf{T}_{120}$) centred at 10.8, and 12.0 μm, and depends of the observation zenith angle (θ). This temperature difference should be greater than 10 K for the method to be accurate:

$$\mathbf{W} = \mathbf{a} \cdot arg^2 + \mathbf{b} \cdot \beta + \mathbf{c}, \tag{5.7}$$

where $\mathbf{a} = -15.1 \cdot sec\theta + 5.1$, $\mathbf{b} = 16.4 \cdot sec\theta - 2.8$, $\mathbf{c} = 0.336 \cdot sec\theta - 0.117$, and $\beta = \dfrac{1}{sec\theta} ln\left(\dfrac{\mathbf{T}^A_{108} - \mathbf{T}^B_{108}}{\mathbf{T}^A_{120} - \mathbf{T}^B_{120}}\right)$. However, this method provides only a few values per day, which can lead to errors in the case of rapidly evolving weather conditions. To address this issue, methods for instantaneous estimation of water vapor are being investigated.

- LST is estimated for day- and night-time acquisitions using the method developed in [106] as follows:

$$
\begin{aligned}
LST = \; & \mathbf{T}_{108} \\
& + \left[1.34 - \tfrac{0.11}{cos^2\theta}\right] \cdot (\mathbf{T}_{108} - \mathbf{T}_{120}) \\
& + \left[0.29 + \tfrac{0.08}{cos^2\theta}\right] \cdot (\mathbf{T}_{108} - \mathbf{T}_{120})^2 \\
& + \left[60.67 - \tfrac{10.01}{cos^2\theta}\right] \cdot (1 - \varepsilon) \\
& + \left[-6.71 + \tfrac{2.47}{cos^2\theta}\right] \cdot \mathbf{W} \cdot (1 - \varepsilon) \\
& + \left[-125.91 + \tfrac{15.09}{cos^2\theta}\right] \cdot \mathbf{\Delta}\varepsilon \\
& + \left[19.44 - \tfrac{4.27}{cos^2\theta}\right] \cdot \mathbf{W} \cdot \mathbf{\Delta}\varepsilon \\
& + \left[-0.44 + \tfrac{0.57}{cos^2\theta}\right],
\end{aligned}
\tag{5.8}
$$

where $\mathbf{T}_{108}$ and $\mathbf{T}_{120}$ are brightness temperature for the infra-red bands centred at 10.8 and 12.0 μm (Ir108 and Ir120) respectively, ε and $\mathbf{\Delta}\varepsilon$ are respectively the average emissivity and the spectral emissivity difference for these bands, W is the total amount of water vapor estimated above, and θ is the observation zenith angle.

- SST is estimated for day- and night-time acquisitions using the method developed by [66]:

$$
SST = \mathbf{T}_{108} + (0.99 \cdot cos\theta + 0.21) \cdot (\mathbf{T}_{108} - \mathbf{T}_{120}) + \left(\frac{0.364}{cos\theta} + 0.15\right) \cdot (\mathbf{T}_{108} - \mathbf{T}_{120}) + \frac{0.327}{cos\theta} + 0.11, \tag{5.9}
$$

where $\mathbf{T}_{108}$ and $\mathbf{T}_{120}$ are the brightness temperatures for MSG bands Ir108 and Ir120, respectively.

5.3 Geo-portal

This section is devoted to the description of our proposed geo-portal system. In subsection 5.3.1 we describe the system architecture of the geo-portal and the different layers that compose it. In subsection 5.3.2, we describe the structure of the database that stores the associated meta-data of the MODIS/SEVIRI scenes and their products. Subsection 5.3.3 describes how the queries to the database are performed in the system. Finally, subsection 5.3.4 describes the procedure used by the geo-portal to retrieve and provide advanced remote sensing scenes and their products to end-users.

5.3.1 System architecture

As shown in Fig. 5.2, the architecture of the proposed system is composed by different layers, which can be defined by their roles. The system follows a modular design in which the communication between layers is performed using standard data exchange formats and transfer protocols, so that any layer can be easily modified and/or enhanced as long as its connectivity with the remaining layers of the system is maintained. More specifically, our design has been carried out using free software tools such as `Symfony2` (described in chapter 2), a full-stack web framework, while the adopted format for data exchange is `JSON`[4], an open standard format that uses human readable text to transmit data objects. In the following, we describe the different layers that compose the geo-portal.

[4]http://www.json.org

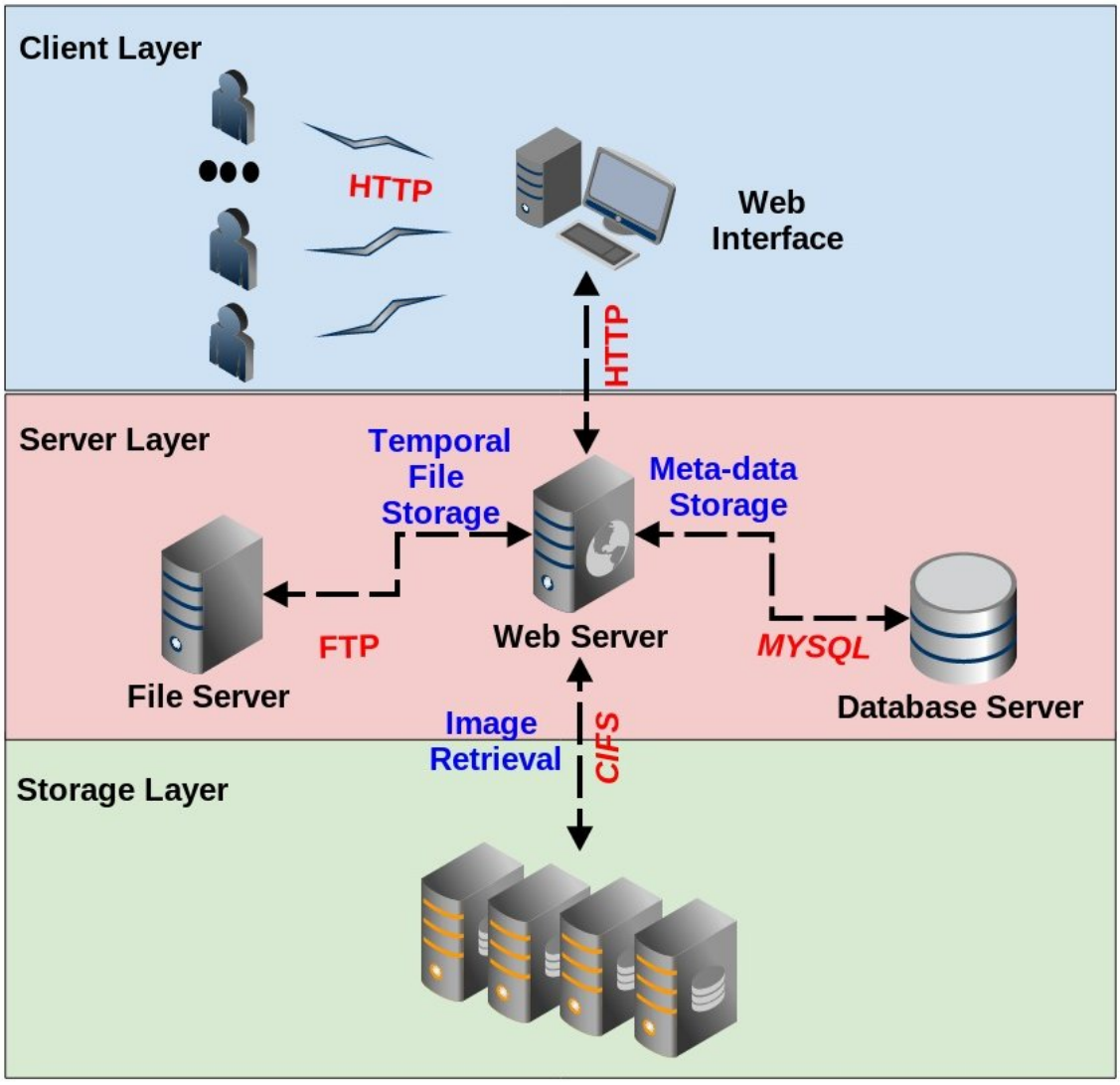


Figure 5.2: Architecture of the proposed geo-portal.

5.3.1.1 Client layer

This layer defines the interactions between the users (through an internet web browser) and our geo-portal, and is responsible for providing users with remote and interactive access to the system. The web interface has been designed using `HTML5`[5], which is the most widely used web programming language, and `CSS3`[6], which is a standard style language for improving the `HTML5` web appearance. In addition, we also used `jQuery`[7], which is a JavaScript library for controlling the web behaviour. The interaction between the user and the web interface is captured by the event handlers of the `jQuery` libraries. The web interface transmits the requests to the server layer via `HTTP`, an application protocol for distributed, collaborative, hypermedia information systems and the foundation of data communication for the World Wide Web. Most of the views are actually generated in the server, using `Symfony2`, a robust web

[5]http://www.w3.org/TR/html51
[6]http://www.w3.org/Style/CSS
[7]http://jquery.com

development framework.

5.3.1.2 Server layer

The system is implemented as a web application, in the sense that it provides remote access to a set of clients (i.e., the end-users of our system) through internet connexions, so that web pages are delivered to the clients while user requests are received and immediately processed. An advantage of this approach is that no additional software has to be installed on the client node, since only a web browser is required for interacting with the geo-portal. The services provided by the system are managed and executed on the server layer, which is composed of several elements with different roles. As Fig. 5.2 shows, the web server handles web interface requests (via HTTP) and manages the system resources, such as meta-data and file data storage. The meta-data are associated to the image scene and their products, and are used for efficient data retrieval as it will be explained in subsequent sections. The server layer can be considered as the main engine of the system since it is in charge of managing and connecting the different components that integrate it. The server layer is also in charge of storing image and product meta-data, following a database scheme that is described in subsection 5.3.2. For the management of the database we have selected MySQL[8], an open-source database manager which provides all the necessary requirements for our system, such as stored functions and procedures, fast queries, and low computational cost. This database manager is widely used for building high performance webs. The MySQL database manager allows for the efficient execution of stored procedures directly, which is needed in our context in order to deal with the high computational cost of executing a matching algorithm for data retrieval involving several terabytes of data. On the other hand, we also used a FTP[9] file storage server in this layer for providing remote user access to non-remotely accessible storage layer files. This module is ultimately in charge of providing a temporal file storage of the files related with the user download requests. The files are automatically deleted once a given limit time (parametrized value) is expired. This strategy allows for efficient downloads of the products retrieved by the geo-portal after the end-user queries are completed.

5.3.1.3 Storage layer

The proposed geo-portal has been specifically designed bearing in mind the need to manage and share a large amount of remote sensing data products obtained after applying the MODIS/SEVIRI processing chain described in section 5.2. The hardware resources utilized in the storage layer comprise a set of DELL NAS[10] processing nodes. First, a DELL PowerVault NX3200[11] system is dedicated to storing the MODIS related products. It consists of a chassis with up to 12 hard SAS[12] drives, and an Intel Xeon E5-2609[13] CPU with 32 GB of RAM memory. Security and high availability is provided by a PERC H710[14] integrated RAID controller which combines multiple hard disks into a logical unit for storage purposes. The overall storage capacity comprises 30 terabytes, using RAID 5 and a hot spare unit. On the other hand, a second DELL node is dedicated to storing SEVIRI products. It consists of another DELL PowerVault NX3200 connected to a DELL PowerVault MD1200[15] which provides

[8] http://www.mysql.com
[9] http://www.w3.org/Protocols/rfc959/
[10] http://www.dell.com/us/business/p/network-file-storage
[11] http://www.dellstorage.com/WorkArea/DownloadAsset.aspx?id=2966
[12] http://www.dell.com/learn/us/en/04/campaigns/dell-hard-drives
[13] http://ark.intel.com/products/64588/intel-xeon-processor-e5
[14] http://www.dell.com/downloads/global/products/pvaul/en/perc-technical-guidebook.pdf
[15] http://www.dell.com/us/business/p/powervault-md1200/pd

24 terabytes of additional storage capacity, for a total of 54 terabytes capacity for the SEVIRI products. Both the PowerVault MD1200 and the PowerVault NX3200 are connected through PERC H800[16] adapters. As mentioned before, the main purpose of this layer is to provide advanced storage capabilities, so all interactions are performed through other layers of the system that access the layer directly. The storage layer and the server layer are located inside a private network which is remotely non-accessible. Both layers are connected by using CIFS[17], a protocol that defines a standard for remote file access allowing up to millions of connections at the same time.

5.3.2 Database structure

Fig. 5.3 illustrates the database scheme used to store remote sensing products in the proposed geo-portal. The database has been designed in order to store information not only from the products, but also from the original remote sensing data (although in our current version only the products are available through the geo-portal). The information stored in the database for each image entry comprises the number of samples, lines, bands, data type, byte order, wavelength information, interleave, thumbnails, geolocation and sensor used (limited to MODIS/SEVIRI in the present version but fully extensible to other sensor data in future developments). Most importantly, our system also contains information about the presence of cloud areas (called *cloudinesss* hereinafter) in the scene. Clouds were identified through IMAPP [126] Level2 Cloud Mask (MOD35) software. In order to minimize the database size, the geolocation data is confined to the four extreme image coordinates of each data set. In practice, each image is decomposed into thirty parts (called quarters) and we store the geolocation and cloudiness information for each quarter. In addition, relevant information about the data products is also stored in the geo-portal, including type of product, resolution, sensor of acquisition, etc. All this information can be used for retrieval purposes using different queries. At this point, we reiterate that the database structure has been designed to store not only the final products but also the raw data. Although in the current version of the geo-portal we only make available the final products to external users, the system is prepared to provide also other different data levels (including the raw data). In the following we describe widely the database tables design.

The database scheme tables could be classified, according to the above statement, as follows, first the tables which contain features about the image, second tables related with product storage and last the authentication/authorization data tables.

5.3.2.1 Image features

This subsection describes the features of each of the tables which are related with the image features.

1. **Images**
   The images table is the main table of the database, since its attributes have been designed to describe the relevant information used for image analysis; furthermore it includes additional information through related tables such as image sensor, quick-look, geolocation coordinates or cloudiness. In the following, the table attributes are listed:

   - **Id**. The identifier of the images table.
   - **Name**. Name given by the image owner.

[16] http://www.dell.com/downloads/global/products/pvaul/en/perc-technical-guidebook.pdf
[17] http://www.samba.org/samba/docs/man/Samba4-HOWTO/protocol.html

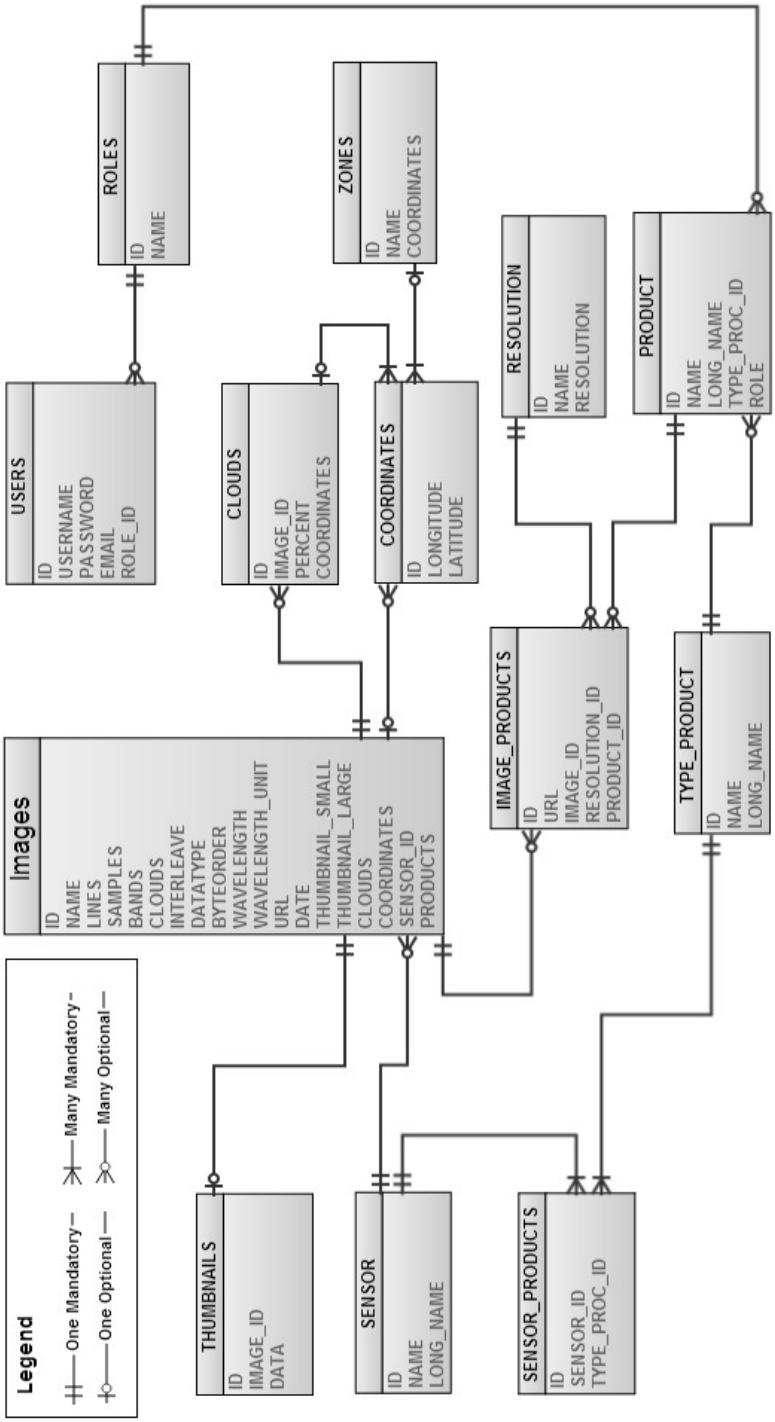


Figure 5.3: Structure of the database used to store data and products in the proposed geo-portal.

- **Interleave**. Interleave feature of the image, following the `ENVI` standard notation. This attribute refers to whether the data are band sequential (BSQ), band interleaved by pixel (BIP), or band interleaved by line (BIL).
- **Lines**. Lines feature of the image, following the `ENVI` standard notation. It refers to the number of lines per image for each band.
- **Samples**. Samples feature of the image, following the `ENVI` standard notation. It refers to the number of samples (pixels) per image line for each band.
- **Bands**. Bands feature of the image, following the `ENVI` standard notation. It refers to the number of bands per image file.
- **DataType**. Data type feature of the image, following the `ENVI` standard notation. It is a parameter identifying the type of data representation, where 1=8 bits (byte); 2 = 16 bits signed integer; 3 = 32 bits signed long integer; 4 = 32 bits floating point; 5 = 64 bits double precision floating point; 6 = 2∗32 bits complex, real-imaginary pair of double precision; 9 = 2∗64 bits double precision complex, real-imaginary pair of double precision; 12 = 16 bits unsigned integer; 13 = 32 bits unsigned long integer; 14 = 64 bits signed long integer; and 15 = 64 bits unsigned long integer.
- **ByteOrder**. Byte order feature of the image, following the `ENVI` standard notation. It describes the order of the bytes in integer, long integer, 64 bits integer, unsigned 64 bits integer, floating point, double precision, and complex data types. Byte order value 0 is Least Significant Byte First (LSF) data (DEC and MS-DOS systems) and byte order value 1 is Most Significant Byte First (MSF) data (all others - SUN, SGI, IBM, HP, DG).
- **Wavelenght**. Lists the center wavelength values of each band in an image.
- **WavelenghtUnit**. Wavelength values metric.
- **ThumbnailSmall**. Special attribute which contains a small image quick-look encoded as text base64 encoded .
- **ThumbnailLarge**. Special attribute which contains a medium resolution quick-look of the image. Foreign key to `Thumbnail` table.
- **URL**. Special attribute which contains the image link to the location in the file storage server, it is assigned automatically by the server in the upload image process.
- **Date**. Image acquisition date.
- **Coordinates**. Coordinates that specify the image location. Foreign key to `Coordinates` table.
- **SensorID**. Sensor that acquired the image. Foreign key to `Sensor` table.
- **Products**. Products performed over the image in processing chain. Foreign key to `Image Products` table.

2. **Thumbnails**
   Medium resolution quick-looks of the image are stored in this table. In the following, the table attributes are listed:

   - **Id**. The identifier of the thumbnails table.
   - **ImageID**. Foreign key to `Image` table.

- **Data**. Quick-look binary data encoded as base64 text.

3. **Sensor type**
   This table stores the information about the sensors which have acquired the images. In the following, the table attributes are listed:

   - **Id**. The identifier of the sensor type table.
   - **Name**. Short name of the sensor.
   - **LongName**. Descriptive name of the sensor.

4. **Coordinates**
   This table stores the coordinate points of the images. In the following the table attributes are listed:

   - **Id**. The coordinates table identifier.
   - **Longitude**. Longitude value.
   - **Latitude**. Latitude value.

5. **Zones**
   Predefined map zones are defined in the system. In the following the attributes of the involved table are listed:

   - **Id**. The zones table identifier.
   - **Name**. Short name of the zone.
   - **Coordinates**. Set of coordinates that geo-locate the zone.

6. **Clouds**
   This table stores the information of each part of the image (30 parts) and the cloudiness rate in them. In the following, the table attributes are listed:

   - **Id**. The clouds table identifier.
   - **Coordinates**. Set of coordinates which defines a quarter of the image.
   - **Clouds**. Cloudiness degree.

5.3.2.2 Product description

This subsection describes the database tables related with the advanced products achieved over the images in our system.

1. **Image products**
   This table stores products which are performed with different algorithms over images from a variety of heterogeneous sensors and resolutions. In the following, the table attributes are listed:

   - **Id**. The image products table identifier.
   - **Url**. Link to the product location on the storage layer.
   - **ResolutionID**. Product spatial resolution. Foreign key to `Resolution` table.
   - **ImageID**. Foreign key to `Image` table.

- **ProductID**. Foreign key to `Products` table.

2. **Resolution**
   The resolution table is designed to store the product spatial resolution, since products can be obtained from images with different spatial resolutions. In the following, the table attributes are listed:

   - **Id**. The resolution table identifier.
   - **Name**. Descriptive name of the resolution.
   - **Resolution**. Resolution value.

3. **Products**
   This table defines available products in the system and the their access constraints. In the following, the table attributes are listed:

   - **Id**. The identifier of the products table.
   - **Name**. Short name of the product.
   - **LongName**. Descriptive name of the product.
   - **Role**. Minimum user role required to access to the product. The system follows a hierarchical role constraints.

4. **Product types**
   Products are grouped according to their characteristics. In the following, the attributes of the involved table are listed:

   - **Id**. The identifier of the product types table.
   - **Name**. Descriptive name of the product type.
   - **LongName**. Descriptive name of the product type.

5. **Sensor products**
   This table specifies what products are available for each sensor. In the following, the table attributes are listed:

   - **Id**. The identifier of the sensor products table.
   - **SensorId**. Foreign key to `Sensor` table.
   - **ProductId**. Foreign key to `Products` table.

5.3.2.3 Access features

This subsection describes tables related with the system authentication/authorization data in our system.

1. **Users**
   User authentication/authorization data is stored in this table. In the following, the table attributes are listed:

   - **Id**. The identifier of the users table.
   - **Name**. User name.

- **Password**. User authentication password .
- **Email**. Email user address.
- **Role**. Foreign key to `Role` table.

2. **Roles** This table describes the roles which are used to restrict user access to the system. In the following, the table attributes are listed:
   - **Id**. The identifier of the roles table.
   - **Name**. Descriptive name of the role.

5.3.3 Queries

Although the aim of our system is to offer remote sensing products, it has been designed to retrieve remote sensing scenes which have associated products. Thus, the queries are performed using the image associated meta-data, which is the fastest procedure since all the products of an image have the same query parameters.

In this subsection, we outline the queries that can be performed to retrieve remote sensing products in the proposed geo-portal system. The geo-portal includes an advanced retrieval system which provides high query performance. The query filters include acquisition date, used sensor, geolocation or cloudiness. It should be noted that the last two filters are only available for MODIS images, due to the geostationary character of the MSG platform, geolocation information for SEVIRI data needs not being included within each image. As for cloud information for SEVIRI data, it is still not available, although we plan to include this information in the geo-portal in near future. In the following, we first describe the searching methodology, which relies on a newly developed geolocation matching algorithm. Then, we describe the procedure used to perform the search from an end-user's point of view (with particular emphasis on the developed web browser interface and the searching options available in the system). Finally, we describe the procedure used by the system to allow for remote downloads of data from end-users.

5.3.3.1 Searching methodology

In order to deal with the time needed to perform image retrieval on a large database such as the one available in our geo-portal, the searching functionality has been designed in a way that the queries are completely executed on the database manager engine. This is a very important aspect, since the possibility to perform the queries directly in `MySQL` allows for much increased performance as the queries are executed as simple `SQL` calls embedded in the database manager. In previous versions of the geo-portal we intended to use an external programming language such as C++ in order to perform the queries, which resulted in a much slower response time. As a result, the embedding of the queries into `MySQL` is considered to be one of the most attractive features of the proposed geo-portal system as this design choice guarantees a high quality of service. The interest region-based searches are implemented by means of a newly developed geolocation matching algorithm, which works as a stored procedure in the database to resolve geolocation-based queries. This algorithm is particularly useful for the retrievals based on the MODIS sensor, since this instrument provides images with irregular geolocation that makes it difficult to link the data with a set of given geo-coordinates. This introduced significant difficulties since our idea was to visualize and retrieve the obtained products in interactive fashion, based on available map services such as Google Maps. In order to address this issue, we have developed a fast matching algorithm which compares two geographic regions using their four extreme coordinates. This methodology is summarized in Algorithm 4.

Algorithm 4 Geolocation matching algorithm

```
1: procedure GelocationMatching(IMAGES,selectedPoints,maxCloudiness)
2:     vectorResults= new Vector
3:     for all IMAGES do
4:         imagePoints = image.Points
5:         resultIn = false
6:         if (No selectedPoints) OR PlaneIntersection(imagePoints, selectedPoints) OR AnyPointInPolygon(imagePoints,
   selectedPoints) then
7:             resultIn=true
8:             if (maxCloudiness) then
9:                 quarters=image.quarters
10:                for all quarters do
11:                    quarterPoints=quarter.Points
12:                    if (PlaneIntersection(quarterPoints, selectedPoints) OR AnyPointInPolygon(quarterPoints, selected-
   Points)) then
13:                        if quarter.cloudPercent > maxCloudiness then
14:                            resultIn=false
15:                            Break For loop
16:                        end if
17:                    end if
18:                end for
19:            end if
20:        end if
21:        if resultIn then
22:            vectorResults.push(image)
23:        end if
24:    end for
25:    return vectorResults;
26: end procedure
```

As shown by Algorithm 4, the main idea of the proposed geolocation procedure is to establish the overlapping between a user-defined region and the corresponding image region. In other words, the user interactively defines an area of interest over a map service (in our case, by drawing a polygon in Google Maps) and the system should retrieve all the stored products of the images that have some area overlapping with the region defined by the user in interactive fashion. The matching procedure is described in detail in Algorithm 5. As shown by this algorithm, all comparisons are based on the four extreme coordinates for the user-defined region and for the associated quarters. In order to manage the computational cost of the geolocation procedure, this technique has been designed in the form of an efficient stored procedure implementation, which exploits the high performance of `MySQL` stored procedure engine and provides low latency in the associated complex queries. At this point, it is important to reiterate that the geographic matching of the regions/quarters is performed using geometric concepts which are based on interpreting the coordinates as points and the regions as planes formed by those points, as described in detail in Algorithm 5. In order to cover all the matching possibilities, we first evaluate if there is any intersection of lines from both planes and, if this is not the case, we evaluate if any point from either plane is inside the other plane by using a point-in-polygon approach developed in [127]. As described in Algorithms 4 and 5, the geolocation matching technique executes different queries based on different filter inputs, which can be simply summarized as follows:

- *No filters:* all the scenes from the same sensor are retrieved.
- *Interest region filter:* only those scenes in which its geographic region matches with the interest region are retrieved.
- *Cloudiness threshold filter:* only those scenes in which all quarters contain cloudiness below the threshold are retrieved.
- *Cloudiness threshold and interest region filters:* only those scenes in which all quarters within the region of interest contain cloudiness below the threshold are retrieved.

Algorithm 5 Matching procedure used by the geolocation matching algorithm

```
1: procedure PointInside(polygonPoints,testx,testy)
2:     NUMPOINTS=4
3:     out=0
4:     if testx < MIN(polygonP.x) OR testx > MAX(polygonP.x) OR testy < MIN(polygonP.y) OR testy > MAX(polygon.y)
   then
5:         j=NUMPOINTS-1
6:         for i = 0 to NUMPOINTS do
7:             z = polygonP[j].x - polygonP[i].x)* (testy - polygonP[i].y) / (polygonP[j].y - polygonP[i].y) + polygonP[i].x
8:             if (polygonP[i].y > testy =! polygonP.y[j] > testy) AND (testx < z) then
9:                 c =! c
10:            end if
11:            j=i
12:        end for
13:    end if
14:    return out
15: end procedure
16: procedure AnyPointInPolygon(imagePoints,selectedPoints)
17:    NUMPOINTS = 4
18:    for i = 0 to NUMPOINTS-1 do
19:        point = imagePoints[i]
20:        if PointInside(selectedPoints,point.x,point.y) then
21:            return true
22:        end if
23:    end for
24:    for i = 0 to NUMPOINTS-1 do point = selectPoints[i]
25:        if PointInside(imagePoints,point.x,point.y) then
26:            return true
27:        end if
28:    end for
29:    return false
30: end procedure
31: procedure PlaneIntersection(imagePoints,selectedPoints)
32:    NUMPOINTS = 4
33:    j = NUMPOINTS − 1
34:    for i = 0 to NUMPOINTS-1 do
35:        if (Line(imagePoints[j], imagePoints[i]) ∩ Line(selectedPoints[0], selectedPoints[1])) OR
36: (Line(imagePoints[j], imagePoints[i]) ∩ Line(selectedPoints[1], selectedPoints[2])) OR
37: (Line(imagePoints[j], imagePoints[i]) ∩ Line(selectedPoints[2], selectedPoints[3])) OR
38: (Line(imagePoints[j], imagePoints[i]) ∩ Line(selectedPoints[3], selectedPoints[0])) then
39:            return true
40:        else
41:            j=i
42:        end if
43:    end for
44:    return false
45: end procedure
```

5.3.3.2 Searching Procedure from End-User Point of View

In this subsection, we outline how the searching procedure is conducted from an end-user's point of view. Fig. 5.4 shows the browser interface of the geo-portal, which allows the user to select several filters using a friendly interface. As shown by Fig. 5.4, the leftmost panel of the browser allows specifying the type of sensor for the retrieval of associated image products, as well as the maximum percentage of clouds allowed in the data to be retrieved. The leftmost panel also allows for the retrieval of image products by date (or range of dates) and by geographic coordinates. On the other hand, as Fig. 5.4 shows the main panel of the browser includes a map obtained from Google Maps service. The map panel is fully interactive, and allows the user to draw a polygon directly on the map so as to perform the retrieval only in the geographical area delimited by the polygon. In order to achieve this functionality, the geolocation matching technique described in Algorithm 4 and its associated matching procedures, described in Algorithm 5, are used. In the following, we summarize the different steps that a simple query involves:

- First, the user can define simple criteria for the query, such as sensor or dates of acquisition, in the leftmost panel of the browser as indicated in Fig. 5.4.

- In a second step, the user can refine the query by drawing a polygon in the map panel. The geolocation matching technique will be used to retrieve only the images located in a geographic area that has some overlapping with the polygon. The system provides other additional options for selecting the region of interest coordinates, such as simple input form or a set of pre-established zones, as indicated in the leftmost panel (see Fig. 5.4).

- As an optional step, the system allows defining a maximum cloudiness filter, which is used to restrict images with low visibility due to cloud contamination. After the user clicks the search button, all the images matching the aforementioned searching criteria are retrieved and the user can now download the desired image products. As Fig. 5.5 shows, the system retrieves, in a table, a list of the image acquisitions together with their associated products, this table is fully interactive and allows to browse among the image meta-data and products. In the following subsection, we explain how the downloads (which can involve multiple users accessing concurrently the database) have been implemented in our geo-portal.

5.3.4 Downloading

The developed geo-portal has been designed to facilitate the access to the products which are required by users. Once the scenes of interest have been retrieved using the searching procedure described in the previous subsection, the user can select and download the products associated to the image matching the search criteria. In this regard, the geo-portal offers the capability to browse among the products of the

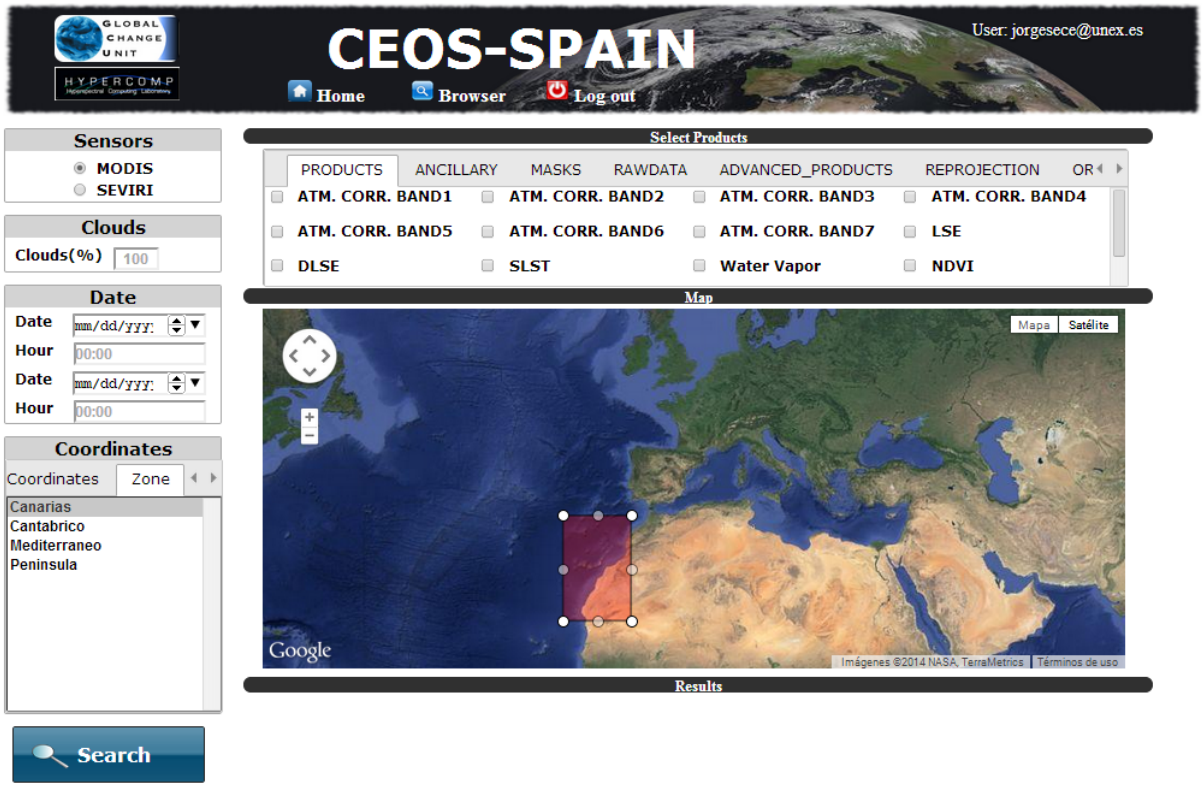


Figure 5.4: Browser interface of the developed geo-portal.

images resulting from the filtering procedure and to select them individually for download. In addition, as Fig. 5.5 shows, the main panel of the browser includes a list of the available products which offers the capability to download the selected products of all the images selected in the table. A file storage server (based on the FTP protocol) is used for downloading purposes, so that the server creates a folder with all the requested products that are available for the user to download. The selected products will be deleted from the server after a number of hours (variable value). In this way, once the user executes the download option by clicking the corresponding button in the browser (see Fig. 5.5), the system creates the temporal folder, copies the selected products to the folder and shows a pop-up message to the user together with the FTP download link. As mentioned before, the system has been implemented so as to support a very large number of simultaneous connections and concurrent data downloads, being able to execute the queries with high performance and with significant quality of service. In the following section we provide an experimental evaluation of the system in terms of geolocation accuracy and computational efficiency in resolving the incoming queries.

5.4 Experimental results

In this section, we evaluate the proposed geo-portal from two different perspectives: the efficiency in the retrieval of data products and the accuracy of the proposed geolocation matching algorithm. The section is organized as follows. In subsection 5.4.1, we describe the MODIS/SEVIRI data sets and products used for our experimental validation. In subsection 5.4.2, we illustrate the accuracy of the proposed

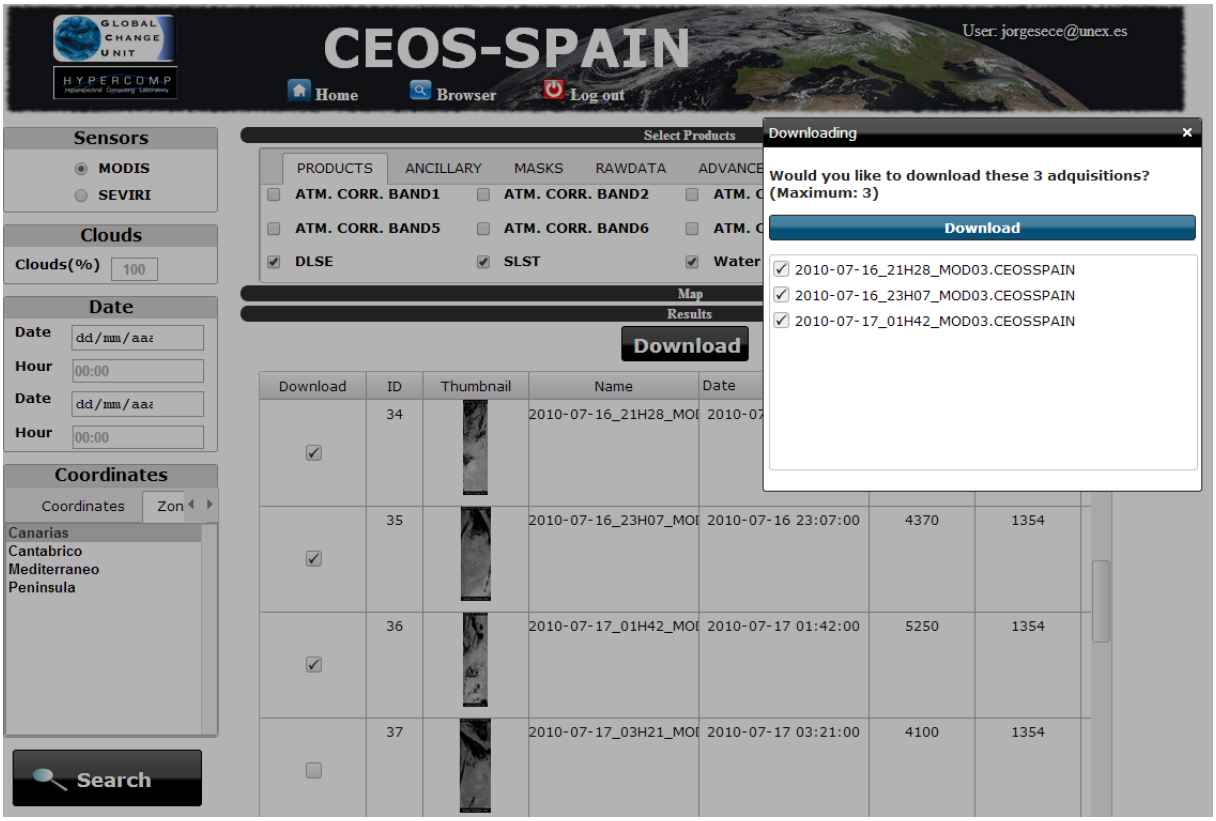


Figure 5.5: Search results retrieved by the browser interface.

geolocation matching algorithm used to retrieve products from our geo-portal. Finally, subsection 5.4.3 performs an evaluation of the computational performance and response times measured in the retrieval process.

5.4.1 Data and products

The experiments have been carried out using the full database of MODIS/SEVIRI images which have been processed (using the methodology described in section 5.2) at UCG/IPL from July 2010 to January 2014 (in the case of MODIS) and from July 2007 to January 2012 (in the case of SEVIRI). This comprises a total of 7626 scenes and 165406 products of MODIS (occupying a total space of more than five terabytes of products) and a total of 163813 scenes and 313530 products of SEVIRI (occupying a total space of more than twelve terabytes of products). For the MODIS data, it should be noted that the MODIS instrument provides images from (plus/minus) 55-degree scanning pattern at the EOS orbit of 705 km, thus achieving a 2330-km swath. Thus, the features of these images are different in every acquisition. This includes width, height or geolocation. In this way, two images from the same area can have different dimensions (width of 1354 plus/minus 10 pixels and height of 5000 plus/minus 200 pixels). This aspect complicated the geolocation process for these images, as indicated before. In order to illustrate this issue further, the following subsection provides an evaluation of the geolocation retrieval accuracy in the case of MODIS data.

5.4.2 Geolocation retrieval accuracy

The accuracy of the proposed geolocation-based retrieval system has been evaluated using a case study based on a query centred in the Canary Islands (35 degrees north latitude, minus 9 degrees south latitude, 20 degrees east longitude and minus 20 degrees west longitude), as indicated in the user-drawn polygon depicted in Fig. 5.4. For illustrative purposes, the query was performed for a specific day (July 12, 2010), in which five different MODIS image were returned. For simplicity, we did not apply the cloudiness threshold in this experiment to remain focused on the evaluation of the geolocation results. For illustrative purposes, Fig. 5.6 graphically illustrates the geolocation accuracy achieved by our system in this particular experiment. More specifically, in Fig. 5.6 the red polygons represent a region of interest defined by the user (see Fig. 5.4), and the blue polygons represent the available MODIS images collected at different times on July 12, 2010. The intersection areas are displayed in cyan color. As shown by Fig. 5.6, three different images exhibited some degree of overlapping with the user-defined area. Note that the images acquired by MODIS at 01h24m and 12h29m did not overlap at all with the user-defined area, hence these images were not retrieved by the geo-portal. In turn, the image acquired by MODIS at 03h02m, 10h48m and 21h52m exhibited some (partial) overlapping with the user-defined search area, thus being retrieved. We conducted several additional experiments using a larger number of involved images and user-defined searching areas with the same accuracy results, which confirmed the high precision of the geolocation matching algorithms developed for the geo-portal.

5.4.3 Efficiency in retrieval

In this subsection we evaluate the retrieval performance of the proposed geo-portal. For this purpose, we considered three different types of queries and measured the time in seconds from the query execution to the query completion, i.e. the effective time that the geo-portal took in order to provide the data solicited by the user for download purposes. Specifically, the types of queries considered in our tests were the following ones:

- In the first type of query, we used a *no filter* scenario so that all the images from the same sensor could be retrieved. We performed this query for all the MODIS data and also for all the SEVIRI data, so that we can evaluate the performance of the system in retrieving all the data stored in the system by sensor type.

- In the second type of query, we analyzed the performance of the *interest region filter*. In this case, we used the same case study illustrated in the previous subsection, in which a query centered in the Canary Islands (35 degrees north latitude, minus 9 degrees south latitude, 20 degrees east longitude and minus 20 degrees west longitude) was used after a user-defined region was established in the map panel of the geo-portal browser as depicted in Fig. 5.4.

- The third type of query analyzed the *cloudiness threshold filter* in combination with the *interest region filter*. This means that, for the considered user-defined region of interest centered in the Canary Islands and depicted in Fig. 5.4, we retained only those images that contain less than 80% cloudiness and discarded the rest. In all cases, we avoided establishing a filter for the retrieval of images based on the date, in order to have a complete exploration of the whole database in multi-temporal fashion.

As shown by Table 5.3, the retrieval times were quite fast for the three types of considered queries. The fastest cases correspond to the retrieval of all MODIS images and all SEVIRI images, which took less than one second to be completed. This is because the only filter applied in the query was the one

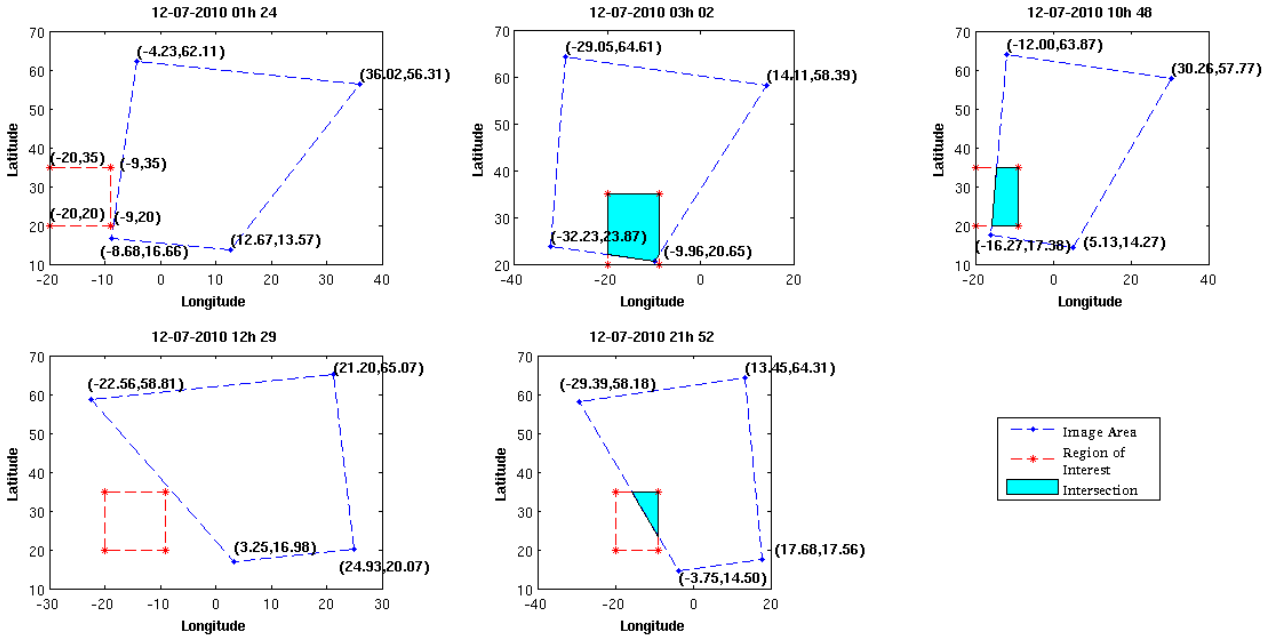


Figure 5.6: A case study illustrating the geolocation matching accuracy of the developed geo-portal. The red polygons represent a region of interest defined by the user (centered over the Canary Islands, as depicted in Fig. 5.4) and the and blue polygons represent the available MODIS images at different hours on July 12, 2010. The intersection areas are displayed in cyan color. In our experiment, only the cases in which there was overlapping between the user-defined area and the available images resulted in data retrievals, which confirmed the good performance of the geolocation matching algorithm.

Table 5.3: Time (in seconds) that the geo-portal took to complete several types of queries applying different filters when exploring the whole database of MODIS/SEVIRI images. The images are filtered by date from July 2010 to January 2014 (in the case of MODIS) and from July 2007 to January 2012 (in the case of SEVIRI). In the queries based on a region of interest, we used an area centred over the Canary Islands depicted in Fig. 5.4. The cloudiness threshold considered in experiments was 80%, so that we filtered out the images containing more clouds than established in this threshold.

	Only Date filter (SEVIRI images)	Only Date filter (MODIS images)	Region of interest (MODIS images)	Cloudiness and region of interest (MODIS images)
Images retrieved	163813	7626	5153	2592
Time (seconds)	0.008	0.008	9.101	37.078

given by the sensor type, which allowed for a very fast retrieval of the images corresponding to each type of sensor. In any event, given the very high number of images retrieved (7626 for MODIS and 163813 for SEVIRI), we consider that the retrieval times were extremely fast. This indicates that the architecture of the proposed system can handle large volumes of data while providing very fast response times for the queries, implemented directly in the `MySQL` database manager. When the region of interest was defined, the query based on the considered region centered around the Canary Islands returned 5153 MODIS scenes, which is also a considerable amount of information. In this case, the geo-portal could provide the result of the query in approximately 9 seconds. Finally, the most complex query executed was completed in approximately 37 seconds. Such response time is due to the fact that the cloudiness threshold was applied in combined fashion with the region of interest filter, which means 30 combined comparisons for each image, since the cloudiness into the image is split into 30 quarters. Thus, this query is more complex in terms of execution than the previously considered ones. Still, the retrieval time in this case is considered to be quite appropriate bearing in mind that the query took as input the whole MODIS image database which comprises approximately nine terabytes of data. At this point we reiterate that the queries retrieve the images and also their products, no product-related filters are available in the searching process, since all the products of an image have the same query parameters.

Chapter 6

Conclusions and future work

In this thesis we have presented several new techniques and methodologies for storing and managing remotely sensed images and their products, with the ultimate goal to develop and make available online new digital repositories for sharing a big amount of remotely sensed multispectral and hyperspectral data. A particularly innovative feature of our newly developed repositories is that they have the capacity to exploit algorithms implemented on GPUs for efficient remote sensing data retrieval. Here, we implemented several algorithms for spectral unmixing purposes in order to generate suitable meta-data for retrieval purposes, based on relatively complex queries including both endmember and abundance information (in the case of hyperspectral image repositories) and advanced products (in the case of multispectral image repositories). With regards to the latter, we have developed a standardized repository for storing a wide amount of remote sensing image products, which is also available on-line providing geo-location retrieval capabilities. The work developed in this thesis is expected to increase the value of the data acquired by available and new airborne/satellite hyperspectral/multispectral imaging instruments, and to improve the availability of these data and their associated information and meta-data. More precisely, the main achievements of this thesis can be summarized as follows:

- First and foremost, we have presented a new digital repository for hyperspectral image data (available online from http://hypercomp.es/repository) that allows uploading and retrieving high-dimensional hyperspectral images. The current version is implemented as a web service, which allows remote user access through a web interface while a server is in charge of managing the repository database and performing algorithm executions. The system is fully available for public use and represents a first step towards a standardized hyperspectral data repository data intended to distribute and share hyperspectral data sets in the community.

- Second, we have included CBIR capabilities based on spectral unmixing concepts in the aforementioned system. State of art of unmixing algorithms such as geometric techniques are used to generate effective meta-data for image retrieval purposes. In order to deal with the high computational cost of extracting spectral information from hyperspectral images, these algorithms have been implemented in parallel, so that, the system executes the algorithms in remote distributed computing resources (always using GPU implementations). The performance of the retrieval system was evaluated using synthetic data sets which allowed an evaluation of the system in a fully controlled scenario, while real image data sets provided a practical illustration of the system using well-known hyperspectral data sets.

- Third, we have extended the aforementioned system by the inclusion of sparse unmixing techniques

for the process of cataloguing and retrieving hyperspectral scenes. The use of sparse unmixing offers an important advantage: the generation of meta-data and the CBIR process can be guided by a spectral library of laboratory measured endmembers instead of endmembers extracted from the hyperspectral image. Since the endmembers in the library are acquired in ideal conditions, this allows us to have a more reliable process for the generation of meta-data and the retrieval itself, while circumventing problems related to the potential impact of the mixture problem, which may prevent the existence of completely pure spectral signatures in the hyperspectral images. The sparse unmixing-based CBIR approach also offers the possibility of a more robust catalog since the meta-data is not generated from the image but extracted from a previously available spectral library with high-quality signatures measured in ideal conditions.

- Last but not least, an on-line geo-portal has been developed (available online from http://ceosspain.lpi.uv.es) to manage a standardized data repository for multispectral images and their advanced products, which includes the possibility of performing geolocation-based data retrieval by defining regions of interest (and applying several types of filters) in a user-friendly browser that allows for a complete interaction with a map server such as Google Maps. The geo-portal has been implemented using advanced software and hardware infrastructures, allowing for the fast execution of queries embedded into our database engine as stored procedures. In the newly developed system the data processing is accomplished in one day, i.e. the multispectral data (e.g., data collected from MODIS/SEVIRI) collected by the receiving antenna is processed by our system once a day. Then the system obtains the resulting products and stores them in our database, which also handles the raw data. This system currently provides impressive features: more than eight thousand MODIS acquisitions (with products that occupy a total space of more than nine terabytes of data) and more than two hundred thousand SEVIRI acquisitions (with products that occupy a total space of more than seventeen terabytes of data). This results in a massive amount of information that is fully available to the remote sensing community through the developed geo-portal.

As future extension of the developed systems, we will implement multicore versions of the spectral unmixing algorithms used to implement the system and explore in more details its potential multi-GPU functionality. Specifically, a multicore implementation would perhaps exhibit less restrictions in terms of communication overhead than a GPU cluster, due to the availability of shared memories. In both cases (multi-GPU and multicore) our CBIR system is expected to adapt quite well to these architectures since the searching part can be efficiently performed in parallel, and the cataloguing part (in spite of the fact that it would depend on the considered endmember extraction implementation in multiple processors) would only need to be used once for each scene. Regarding the multispectral product repository, our future work will be mainly focused on including additional and products collected from other sensors. However, another possible development is to integrate both systems (CBIR and geo-portal). Another important future research line is a full standardized repository for multispectral/hyperspectral data with a retrieval retrieval system based on image content and geolocation criteria. This may bring some challenges, as the characteristics of multispectral images and hyperspectral images are different, and particularly the way to exploit them and extract information may be quite different depending on the considered scenario. However, the idea to combine all developments into a unified repository is appealing and we are currently exploring ways to achieve such desired integration. In terms of data storing, we plan to migrate the proposed systems to big data services in the cloud, in order to overcome current storage limits and foster the adoption of our proposed framework as a standard for data sharing in the

remote sensing community.

Apendix A

Publications

The results of this thesis work have been published in several international journal papers and peer-reviewed international conference papers. Specifically, the candidate candidate is the first author of 4 journal citation reports (JCR) papers (2 submitted and the rest already published), and 8 peer-reviewed international conference papers directly related to this thesis work. This thesis work has been partially supported by the following programs, fund and resources:

1. CEOS-SPAIN project funded by the Spanish Ministry of Science and Innovation, reference AYA2011-29334-C02-02.

2. High Performance Computing Service Centre project of West University of Timisoara (Romania), reference FP7-REGPOT-CT-2011-284595-HOST.

3. Basic support to research groups by the "Gobierno de Extremadura" (Spain), reference GR10035.

4. CAPAP-H4 network, funded by the Spanish Ministry of Science and Innovation, reference TIN2011-15734-E.

The computing resources used in this work have been provided by two institutions. First, Extremadura Research for Advanced Technologies center (CETA-CIEMAT) which is funded by the European Regional Development Fund (ERDF). This center belongs to the Spanish Ministry of Science and Innovation. And last but not least, the High Performance Computing center at West University of Timisoara, Romania.

The candidate has been a pre-doctoral researcher in the Hyperspectral Computing Laboratory (HyperComp), Department of Technology of Computers and Communications, University of Extremadura, Spain, funded by the project AYA2011-29334-C02-02. Below, we provide a description of the publications achieved by the candidate providing also a short description of the journal or workshop where they were presented.

A.1 International journal papers

1. J. Sevilla, A. Bernabe and A. Plaza. Unmixing-based content retrieval system for remotely sensed hyperspectral imagery on GPUs. *Springer Journal of Supercomputing*, accepted for publication, in press, 2014 [JCR(2013)=0.841]. This paper was published in the *Journal of Supercomputing*, which is a very important journal in the second quartile of the Computer science, hardware and

architecture category of JCR. This paper, which provided the core of chapter 2 of this thesis, shows a new unmixing-based retrieval system for remotely sensed hyperspectral imagery using spectral information. It particularly focuses on the system design and its efficient implementation on GPUs. The spectral information relevant to a query is extracted through a partial unmixing chain which includes algorithms such as VD and HySime for estimation of the number of endmembers of a given scene and N-FINDR and OSP-GS for extraction of these endmembers.

2. J. Sevilla and A. Plaza. A new digital repository for hyperspectral imagery with unmixing-based retrieval functionality implemented on GPUs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2267-2280, June 2014 [JCR(2013)=2.827]. This paper was published in the journal *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, which is a very important journal in the first quartile of the remote sensing and electrical and electronic engineering areas of JCR. This paper presents a new digital repository for hyperspectral image data that allows to upload and retrieve images through a CBIR (content-based image retrieval) functionality based on spectral unmixing concepts. The paper, which provided the core of chapter 3 of this thesis, is focused on the efficient implementation of the system as web service, particularly, it describes widely the digital repository design and CBIR implementation on GPUs, furthermore, it presents a new methodology for image retrieval. Such retrieval is based on spectral/spatial information which is extracted using several full spectral unmixing chains which, on the one hand, extract the spectral information through algorithms such as VD and HySime for estimation of the number of endmembers of a given scene and N-FINDR and OSP-GS for extraction of these endmembers and, on the other hand, incorporate the spatial information in the search using algorithms to estimate the endmember abundance, such as UCLS or ISRA.

A.2 International journal papers (submitted)

1. J. Sevilla, L.I. Jiménez and A. Plaza. Sparse unmixing-based hyperspectral imagery retrieval system implemented on GPUs. Submitted to *IEEE Geoscience and Remote Sensing Letters* [JCR(2013)=1.809]. This paper has been submitted for publication to the journal *IEEE Geoscience and Remote Sensing Letters* which is a journal in the second quartile of the *Remote Sensing* category of JCR. This paper, which provided the core of chapter 4 of this thesis, is focused on developing a content-based image retrieval system which uses sparse unmixing techniques to generate the relevant information to a query. The retrieval is based on spectral/spatial information which is efficiently extracted using the implementation of CSUNSAL algorithm on GPUs. The paper shows a comparison, in terms of efficiency and accuracy, between the proposed system and a previous CBIR implemented using a full spectral unmixing chain which derives the endmembers from the images to be processed.

2. J. Sevilla, Y. Julien, G. Sória, J.A. Sobrino and A. Plaza. A new geo-portal for MODIS/SEVIRI image products with geolocation-based retrieval functionality. Submitted to the *Journal of Applied Remote Sensing* [JCR(2013)=0.892]. This paper, which provided the core of chapter 5 of this thesis, has been submitted for publication to the *Journal of Applied Remote Sensing*, which is a journal in the second quartile of the *Remote Sensing* category of JCR. The paper describes a new data repository for multispectral imagery products. Particularly, it describes a standardized repository for storing a wide amount of remote sensing image formats and products. The paper is focussed

on the efficient implementation of the system as web service, particularly, it describes the digital repository design and the efficient implementation of the geo-portal, which allows for a user-friendly and effective access to a full repository of MODIS/SEVIRI advanced products using geolocation retrieval capabilities.

A.3 Peer-reviewed international conference papers

1. J. Sevilla, S. Bernabe, P. Garcia and A. Plaza. A new digital repository for remotely sensed hyperspectral imagery with unmixing-based retrieval functionality, *SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference, San Diego, USA*. This work was presented as an oral contribution in the *SPIE Optics and Photonics* workshop, which is a very important event held yearly and which gathers more than 2000 international researchers in the area of remote sensing in the city of San Diego, California. In this paper we presented the first step towards the development of a digital repository for remotely sensed hyperspectral data. Particularly, it describes the design of the repository and the integration of the the different unmixing algorithms. The proposed system in this work allows searching for images stored in the database using spectral unmixing concepts.

2. J. Sevilla, S. Bernabe and A. Plaza. Unmixing-based retrieval system for remotely sensed hyperspectral imagery on GPUs, *13th the International Conference on Computational and Mathematical Methods in Science and Engineering, Almería, Spain.* This work was presented as an oral contribution in the workshop *CMMSE 2013 - Minisymposium: HPC.* This symposium is often attended by hundreds of worldwide researches with a fair knowledge/interest in high performance computing applied to complex large-scale computational problems. In this work, we introduced a new unmixing-based image retrieval system for remote sensed hyperspsectral imagery which deals with the computational cost of executing spectral unmixing algorithms using GPUs.

3. J.M. Franco, J. Sevilla and A. Plaza. Parallel implementation of pixel purity index for a GPU cluster, *13th the International Conference on Computational and Mathematical Methods in Science and Engineering, Almería, Spain.* This work was presented as an oral contribution in the workshop *CMMSE 2013 - Minisymposium: HPC.* This symposium is often attended by hundreds of worldwide researches with a fair knowledge/interest of high performance computing applied to complex large-scale computational problems. In this work, we presented a parallel implementation of the Pixel Purity Index (PPI) algorithm, a well-known algorithm for the analysis of remotely sensed hyperspectral data sets, targeted to a GPU cluster, and showed a comparison of execution times and speedups among different parallel implementations of the algorithm.

4. J. Sevilla and A. Plaza. A new digital repository for remotely sensed hyperspectral imagery on GPUs, *15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania*, published in *IEEE* as proceeding. This work was presented as an oral contribution in the workshop *SYNASC 2013 -HPCReS.* This workshop is focused on applications needing high-performance computing, physical modeling of large-scale problems, and the development of scalable algorithms for solving large scale problems on modern parallel and distributed high-performance computing platforms. This paper introduces an on-line spectral unmixing-based content based image retrieval (CBIR) system which allows searching for images

from the database using spectral/spatial information of the scene. The relevant information to a query is performed using efficient implementations of spectral unmixing algorithms on GPUs.

5. R. Ramos-Pollan, J.M. Franco, J. Sevilla, M.A. Guevara-Lopez, N. Gonzalez de Posada, J. Loureiro and I. Ramos. Grid infrastructures for developing mammography CAD systems, *Proceedings of the IEEE Conference of the Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference*. The conference program consisted of plenary lectures, symposia, workshops and invited sessions of the latest significant findings and developments in all the major fields of biomedical engineering. This work was presented as an oral contribution in the *EMBC 2010* conference. This paper presents a set of technologies developed to exploit Grid infrastructures for breast cancer CAD, that include (1) federated repositories of mammography images and clinical data over Grid storage, (2) a workstation for mammography image analysis and diagnosis and (3) a framework for data analysis and training machine learning classifiers over Grid computing power specially tuned for medical image based data.

6. R. Ramos-Pollan, J.M. Franco, J. Sevilla, N.G. Posada, N.P. Perez, M.A.P. Vaz, J. Loureiro, I. Ramos and M.A. Guevara-Lopez. Grid computing for breast cancer CAD. A pilot experience in a medical environment, *Ibergrid: 4th Iberian Grid Infrastructure Conference*. This work was presented as an oral contribution in the *Ibergrid 2010* conference. The conference presents joint research and development agreements between Portugal and Spain in the field of advanced computing infrastructures for e-Science. This paper presents a novel Grid based software platform to store and manage large mammography digital image repositories (MDIR) including associated patient information (clinical history, biopsies, etc.), a mammography image workstation for analysis and diagnosis (MIWAD) and a data training and analysis framework (DTAF). MDIR simplifies and reduces the cost of hosting digitalized content and meta-data stored on Grid infrastructures, exploiting its features such as strong security contexts, data federation, and large storage and computing capacities.

7. A. Calanducci, R. Barbera, J. Sevilla, A. De Filippo, M. Saso, S. Iannizzotto, F. De Mattia and D. Vicinanza. Data grids for conservation of cultural inheritance, *Proceedings of the 1st ACM workshop on Data grids for eScience, Ischia, Italy*, published in *ACM* as proceeding. This work was presented as an oral contribution in the *DataGreS 2009* workshop. The main purpose of this workshop was to bring researchers and practitioners to identify and explore open issues as well as discuss and propose novel data grid oriented solutions for e-Science. This paper presents the online gLibrary platform, which is a grid-based system to host and manage digital libraries. In order to demonstrate how digital libraries on the grid can guarantee enduring preservation of literary heritage, a working prototype of the digital repository for manuscripts of an italian writer called Federico de Roberto was implemented on gLibrary.

8. A. Calanducci, J. Sevilla, R. Barbera, G. Andronico, M. Saso, A. De Filippo, S. Iannizzotto, D. Vicinanza and F. De Mattia. Cultural heritage digital libraries on data grids, *13th European Conference on Digital Libraries (ECDL), Corfu, Grecia*. This work was presented as an oral contribution in the *ECDL 2009* conference. ECDL conference is the major European forum focusing on digital libraries and associated technical, practical, and social issues, meeting the needs of a large and diverse constituency, which includes practitioners, researchers, educators, policy makers and users. This work shows how we deployed some digital repositories of ancient manuscripts making use of gLibrary, a grid-based system, to host and manage digital libraries.

Bibliography

[1] A.W.M Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Fast dimensionality reduction and simple PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000. [Cited in pag. 1]

[2] A. Plaza and C.-I Chang. *High performance computing in remote sensing*. Taylor & Francis: Boca Raton, FL, 2007. [Cited in pag. 1]

[3] A.F.H. Goetz, G. Vane, J E. Solomon, and B.N. Rock. Imaging spectrometry for Earth remote sensing. *Science*, 228:1147–1153, 1985. [Cited in pag. 1]

[4] A. Plaza, J.A. Benediktsson, J.W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113:S110–S122, 2009. [Cited in pags. 1 and 4]

[5] J.D.M. McKeown, S.D. Cochran, S.J. Ford, J.C. McGlone, J.A. Shufelt, and D.A. Yocum. Fusion of HYDICE hyperspectral data with panchromatic imagery for cartographic feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 37(3):1261–1277, 1999. [Cited in pag. 1]

[6] R.O. Green, M.L. Eastwood, C.M. Sarture, T.G. Chrien, M. Aronsson, B.J. Chippendale, J.A. Faust, B.E. Pavri, C. J. Chovit, M. Solis, et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sensing of Environment*, 65(3):227–248, 1998. [Cited in pags. 1 and 11]

[7] P. Gamba, F. Dell'Acqua, A. Ferrari, J.A. Palmason, and J.A. Benediktsson. Exploiting spectral and spatial information in hyperspectral urban data with high resolution. *IEEE Geoscience and Remote Sensing Letters*, 1:322–326, 2004. [Cited in pags. 1 and 11]

[8] E.M. Middleton, S.G. Ungar, D.J. Mandl, L. Ong, S.W. Frye, P.E. Campbell, D.R. Landis, J.P. Young, and N.H. Pollack. The Earth observing one (EO-1) satellite mission: over a decade in space. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):243–256, 2013. [Cited in pag. 1]

[9] K. Segl, L. Guanter, C. Rogass, T. Kuester, S. Roessner, H. Kaufmann, B. Sang, V. Mogulsky, and S. Hofer. EeteS: The EnMAP end-to-end simulation tool. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):522–530, 2012. [Cited in pag. 1]

[10] C. Galeazzi, A. Sacchetti, A. Cisbani, and G. Babini. The PRISMA program. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 4, pages 105–108, 2008. [Cited in pag. 1]

[11] M.J. Barnsley, J.J. Settle, M.A. Cutter, D.R. Lobb, and F. Teston. The PROBA/CHRIS mission: a low-cost smallsat for hyperspectral multiangle observations of the Earth surface and atmosphere. *IEEE Transactions on Geoscience and Remote Sensing*, 42(7):1512–1520, 2004. [Cited in pag. 1]

[12] M. Abrams, D. Pieri, V. Realmuto, and R. Wright. Using EO-1 Hyperion data as HyspIRI preparatory data sets for volcanology applied to Mt Etna, Italy. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):375–385, 2013. [Cited in pag. 1]

[13] H. Wu, L. Ni, N. Wang, Y. Qian, B.-H. Tang, and Z.L Li. Estimation of atmospheric profiles from hyperspectral infrared IASI sensor. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1485–1494, 2013. [Cited in pag. 1]

[14] J. 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. [Cited in pags. 2, 4 and 74]

[15] A. Plaza, J. Plaza, A. Paz, and S. Sánchez. Parallel hyperspectral image and signal processing. *IEEE Signal Processing Magazine*, 28(3):119–126, 2011. [Cited in pags. 2 and 35]

[16] A. Plaza, Q. Du, Y.-L-Chang, and R. King. High performance computing for hyperspectral remote sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):528–544, 2011. [Cited in pag. 2]

[17] C. Lee, S. Gasster, A. Plaza, C.-I. Chang, and B. Huang. Recent developments in high performance computing for remote sensing: a review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):508–527, 2011. [Cited in pag. 2]

[18] N. Chen, Z. Chen, L. Di, and J. Gong. An efficient method for near-real-time on-demand retrieval of remote sensing observations. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):615–625, 2011. [Cited in pag. 2]

[19] D. Brunner, G. Lemoine, F.-X. Thoorens, and L. Bruzzone. Distributed geospatial data processing functionality to support collaborative and rapid emergency response. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2(1):33–46, 2009. [Cited in pag. 2]

[20] M. Stasolla and P. Gamba. Spatial indexes for the extraction of formal and informal human settlements from high-resolution SAR images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 1(2):98–106, 2008. [Cited in pag. 2]

[21] S.S. Durbha, R.L. King, S.K. Amanchi, S. Bheemireddy, and N.H. Younan. Standards-based middleware and tools for coastal sensor web applications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 3(4):451–466, 2010. [Cited in pag. 2]

[22] A.F.H. Goetx and B. Kindel. Comparison of unmixing result derive from aviris, high and low resolution, and hydice images at cuprite. In *Proceedings IX NASA/JPL Airbone Earth Science Workshop, Pasadena, California*, 1999. [Cited in pag. 3]

[23] D. Landgrebe. Multispectral data analysis, a signal theory perspective. Technical report, Purdue University, West Lafayette, 1998. [Cited in pag. 3]

[24] R.A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing, 2nd edition.* Academic Press: New York, 1997. [Cited in pag. 3]

[25] J. Chanussot, M. M. Crawford, and B.-C. Kuo. Foreword to the special issue on hyperspectral image and signal processing. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):3871–3876, 2010. [Cited in pags. 3 and 57]

[26] A. Plaza, Q. Du, J. Bioucas-Dias, X. Jia, and F. Kruse. Foreword to the special issue on spectral unmixing of remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4103–4110, 2011. [Cited in pags. 3 and 4]

[27] P.B. Garcia-Allende, O.M. Conde, J. Mirapeix, A. Cobo, and J.M. Lopez-Higuera. Quality control of industrial processes by combining a hyperspectral sensor and fisher's linear discriminant analysis. *Sensors and Actuators B: Chemical*, 129(2):977–984, 2008. [Cited in pag. 4]

[28] A.A. Gowen, C.P. O'Donnell, P.J. Cullen, G. Downey, and J.M. Frias. Hyperspectral imaging–an emerging process analytical tool for food quality and safety control. *Trends in Food Science & Technology*, 18(12):590–598, 2007. [Cited in pag. 4]

[29] K. Flynn, R. O Leary, C. Lennard, C. Roux, and B.J. Reedy. Forensic applications of infrared chemical imaging: multi-layered paint chips. *Journal of forensic sciences*, 50(4):832, 2005. [Cited in pag. 4]

[30] V.-D. Tuan. A hyperspectral imaging system for in vivo optical diagnostics. *Engineering in Medicine and Biology Magazine, IEEE*, 23(5):40–49, Sept 2004. [Cited in pag. 4]

[31] C. Grabinski, J. Schlager, and S. Hussain. Hyperspectral microscopy for characterization of gold nanoparticles in biological media and cells for toxicity assessment. In *Nanomaterial Interfaces in Biology*, pages 167–178. Springer, 2013. [Cited in pag. 4]

[32] K.J. Zuzak, T.J. Perumanoor, S.C. Naik, M. Mandhale, and E.H. Livingston. A multimodal reflectance hyperspectral imaging system for monitoring wound healing in below knee amputations. In *Engineering in Medicine and Biology Workshop, 2007 IEEE Dallas*, pages 16–18, Nov 2007. [Cited in pag. 4]

[33] R.C. Lyon, D.S. Lester, E.N. Lewis, E. Lee, X .Y. Lawrence, E.H. Jefferson, and A.S. Hussain. Near-infrared spectral imaging for quality assurance of pharmaceutical products: analysis of tablets to assess powder blend homogeneity. *AAPS PharmSciTech*, 3(3):1–15, 2002. [Cited in pag. 4]

[34] G. Camps-Valls and L. Bruzzone. Kernel-based methods for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43:1351–1362, 2005. [Cited in pag. 4]

[35] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, 2002. [Cited in pags. 4, 11, 36, 37, 53 and 77]

[36] 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. [Cited in pags. 4, 11, 35 and 74]

[37] A. Plaza, P. Martinez, R. Perez, and J. Plaza. Spatial/spectral endmember extraction by multidimensional morphological operations. *IEEE Transactions on Geoscience and Remote Sensing*, 40(9):2025–2041, 2002. [Cited in pag. 5]

[38] D.M. Rogge, B. Rivard, J. Zhang, A. Sánchez, J. Harris, and J. Feng. Integration of spatial–spectral information for the improved extraction of endmembers. *Remote Sensing of Environment*, 110(3):287–303, 2007. [Cited in pag. 5]

[39] G. Martin and A. Plaza. Region-based spatial preprocessing for endmember extraction and spectral unmixing. *IEEE Geoscience and Remote Sensing Letters*, 8:745–749, 2011. [Cited in pags. 5, 60 and 78]

[40] M.-D. Iordache, J. Bioucas-Dias, and A. Plaza. Sparse unmixing of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 49(6):2014–2039, 2011. [Cited in pags. 5, 42, 44, 73, 74 and 78]

[41] A. Plaza, Q. Du, Y.-L. Chang, and R.L. King. Foreword to the special issue on high performance computing in earth observation and remote sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):503–507, 2011. [Cited in pags. 5 and 45]

[42] M.A. Veganzones and M. Graña. A spectral/spatial cbir system for hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):488–500, 2012. [Cited in pags. 8 and 35]

[43] A. Plaza, J. Plaza, and A. Paz. Parallel heterogeneous CBIR system for efficient hyperspectral image retrieval using spectral mixture analysis. *Concurrency and Computation: Practice and Experience*, 22(9):1138–1159, 2010. [Cited in pags. 8 and 36]

[44] MRTWeb 2.0. USGS MRT Web 2.0. *Available online: https://mrtweb.cr.usgs.gov/ (accessed on 07th June 2014)*, 2013. [Cited in pags. 9, 11 and 88]

[45] LAADS. Level 1 and atmosphere archive and distribution system. *Available online: http://ladsweb.nascom.nasa.gov/ (accessed on 07th June 2014)*. [Cited in pags. 9, 11 and 88]

[46] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011. [Cited in pags. 11 and 74]

[47] J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado. Parallel morphological endmember extraction using commodity graphics hardware. *IEEE Geoscience and Remote Sensing Letters*, 43(3):441–445, 2007. [Cited in pags. 11 and 45]

[48] W. Lucht, A.H. Hyman, A.H. Strahler, M.J. Barnsley, P. Hobson, and J.-P. Muller. A comparison of satellite-derived spectral albedos to ground-based broadband albedo measurements modeled to satellite spatial scale for a semidesert landscape. *Remote Sensing of Environment*, 74:85–98, 2000. [Cited in pags. 11 and 91]

[49] EarthExplorer. USGS EarthExplorer. *Available online: http://earthexplorer.usgs.gov (accessed on 07th June 2014)*. [Cited in pags. 11 and 88]

[50] G. Gutman and A. Ignatov. The derivation of the green vegetation fraction from NOAA/AVHRR data for use in numerical weather prediction models. *International Journal of Remote Sensing*, 19(8):1533–1543, 1998. [Cited in pags. 11, 88 and 91]

[51] J. M. Bioucas-Dias and J. M. P. Nascimento. Hyperspectral subspace identification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2435–2445, 2008. [Cited in pags. 11, 30, 40, 41 and 45]

[52] M.E. Daube-Witherspoon and G. Muehllehner. An iterative image space reconstruction algoritm suitable for volume ETC. *IEEE Transcaction on Medical Imaging*, 5:61–66, 1985. [Cited in pags. 11, 30, 44 and 45]

[53] A. Green, M. Berman, P. Switzer, and M. Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on Geoscience and Remote Sensing*, 26:65–74, 1988. [Cited in pags. 11 and 41]

[54] F. Pasternak, P. Hollier, and J. Jouan. SEVIRI, the new imager for meteosat second generation. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, volume 3, pages 1094–1099, Aug 1993. [Cited in pags. 11 and 88]

[55] J.B. Lee, S. Woodyatt, and M. Berman. Enhancement of high spectral resolution remote-sensing data by noise-adjusted principal components transform. *IEEE Transactions on Geoscience and Remote Sensing*, 28(3):295–304, 1990. [Cited in pags. 11 and 41]

[56] C.J. Tucker. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, 8(2):127 – 150, 1979. [Cited in pags. 11, 88 and 91]

[57] 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. [Cited in pags. 11, 30, 35, 42 and 45]

[58] J.C. Jiménez Muñoz, J.A. Sobrino, C. Mattar, G. Hulley, and F.-M. Gottsche. Temperature and emissivity separation from MSG/SEVIRI data. *IEEE Transactions on Geoscience and Remote Sensing*, accepted for publication, in press, 2014. [Cited in pags. 11 and 88]

[59] J.A. Sobrino, J. El Kharraz, and Z.-L. Li. Surface temperature and water vapour retrieval from MODIS data. *International Journal of Remote Sensing*, 24(24):5161–5182, 2003. [Cited in pags. 11, 88, 91, 92 and 93]

[60] J. Harsanyi and C.-I. Chang. Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):779–785, Jul 1994. [Cited in pags. 11, 30, 43, 45, 56 and 76]

[61] A. Plaza S. Bernabe, S. Lopez and R. Sarmiento. GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 10:221–225, 2013. [Cited in pags. 11 and 45]

[62] J.A. Richards and X. Jia. *Remote sensing digital image analysis: an introduction.* Springer, 2006. [Cited in pags. 11 and 41]

[63] J.W. Boardman, F.A. Kruse, and R.O. Green. Mapping target signatures via partial unmixing of AVIRIS data. *Proceedings of the JPL Airborne Earth Science Workshop*, pages 23–26, 1995. [Cited in pags. 11, 35 and 45]

[64] H. Rahman and G. Dedieu. SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum. *International Journal of Remote Sensing*, 15(1):123–143, 1994. [Cited in pags. 11 and 90]

[65] C.-I Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Kluwer Academic/Plenum Publishers: New York, 2003. [Cited in pags. 11, 35 and 45]

[66] M. Romaguera, J A. Sobrino, and F.-S. Olesen. Estimation of sea surface temperature from SEVIRI data: algorithm testing and comparison with AVHRR products. *International Journal of Remote Sensing*, 27(22):5081–5086, 2006. [Cited in pags. 11, 88 and 94]

[67] M. Atitar and J.A. Sobrino. A split-window algorithm for estimating LST from Meteosat 9 data: Test and comparison with in situ data and MODIS LSTs. *IEEE Geoscience and Remote Sensing Letters*, 6(1):122–126, Jan 2009. [Cited in pags. 11 and 88]

[68] J. Bioucas-Dias and M. Figueiredo. Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing. *Proceedings of the 2nd WHISPER*, 2010. [Cited in pags. 11, 73 and 74]

[69] L. Scharf. *Statistical Signal Processing, Detection Estimation and Time Series Analysis*. Addison-Wesley, 1991. [Cited in pags. 11 and 41]

[70] S. Bernabe, S. Sánchez, A. Plaza, S. Lopez, J.A. Benediktsson, and R. Sarmiento. Hyperspectral unmixing on GPUs and multi-core processors: A comparison. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1386–1398, 2013. [Cited in pags. 11, 30, 44 and 45]

[71] GCU/IPL. Global Change Unit, Image Processing Laboratory at University of Valencia. *http://www.uv.es/ucg*. [Cited in pags. 11 and 88]

[72] USGS. United States Geological Survey. *Available online: http://www.usgs.gov/ (accessed on 07th June 2014)*. [Cited in pag. 11]

[73] J. Nascimento and J. Bioucas-Dias. Vertex Component Analysis: A Fast Algorithm to Unmix Hyperspectral Data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(4):898–910, 2005. [Cited in pags. 11 and 45]

[74] Kogan F. N. Noaa plays leadership role in developing satellite technology for drought watch. *Earth Observation Magazine*, pages 18–21, September 1994. [Cited in pags. 11 and 91]

[75] C.-I Chang and Q. Du. Estimation of number of spectrally distinct signal sources in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 42(3):608–619, 2004. [Cited in pags. 11, 30, 35, 40, 45 and 76]

[76] J. Sevilla, S. Bernabe, and A. Plaza. Unmixing-based content retrieval system for remotely sensed hyperspectral imagery on GPUs. *The Journal of Supercomputing*, accepted for publication, in press, 2014. [Cited in pags. 13 and 35]

[77] J. Sevilla and A. Plaza. A new digital repository for hyperspectral imagery with unmixing-based retrieval functionality implemented on GPUs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2267–2280, June 2014. [Cited in pags. 13 and 35]

[78] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services*. Springer, 2004. [Cited in pag. 13]

[79] R. Ramakrishnan and J. Gehrke. Database management systems. *Osborne McGraw-Hill*, 2000. [Cited in pag. 16]

[80] T.M.H. Reenskaug. The original mvc reports. Technical report, XEROX PARC, 1978-1979. [Cited in pag. 16]

[81] M. Graña and M.A. Veganzones. An endmember-based distance for content based hyperspectral image retrieval. *Pattern Recognition*, 45(9):3472 – 3489, 2012. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2011). [Cited in pag. 35]

[82] M.A. Veganzones, J.O. Maldonado, and M. Graña. On content-based image retrieval systems for hyperspectral remote sensing images. In Manuel Graña and Richard J. Duro, editors, *Computational Intelligence for Remote Sensing*, volume 133 of *Studies in Computational Intelligence*, pages 125–144. Springer Berlin Heidelberg, 2008. [Cited in pag. 35]

[83] C.-I Chang and A. Plaza. A fast iterative algorithm for implementation of pixel purity index. *IEEE Geoscience and Remote Sensing Letters*, 3(1):63–67, 2006. [Cited in pag. 35]

[84] M.A. Veganzones. Contributions to hyperspectral image processing from lattice computing and computational intelligence. *PhD Dissertation, University of the Basque Country, Spain*, pages 1–202, 2012. [Cited in pag. 35]

[85] R.N. Clark, G.A. Swayze, A.J. Gallagher, T.V.V. King, and W.M. Calvin. The U.S. Geological Survey, digital spectral library: Version 1: 0.2 to 3.0 microns. *U.S. Geological Survey Open File Report*, pages 93–592, 1993. [Cited in pag. 38]

[86] M. Craig. Minimum-volume transforms for remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing*, 32:542–552, 1994. [Cited in pag. 42]

[87] L. Miao and H. Qi. Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 45(3):765–777, 2007. [Cited in pag. 42]

[88] J. Li and J. Bioucas-Dias. Minimum volume simplex analysis: a fast algorithm to unmix hyperspectral data. In *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, volume 3, pages III–250. IEEE, 2008. [Cited in pag. 42]

[89] E.M.T. Hendrix, I. García, J. Plaza, and A. Plaza. Minimum volume simplicial enclosure for spectral unmixing. Technical Report MWP-50, Mansholt Graduate School, Wageningen University, The Netherlands, 2010. [Cited in pag. 42]

[90] 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. [Cited in pag. 42]

[91] J. Nickolls and W. J. Dally. The GPU computing era. *IEEE Micro*, 30:56–69, 2010. [Cited in pag. 45]

[92] A. Plaza, J. Plaza, and H. Vegas. Improving the performance of hyperspectral image and signal processing algorithms using parallel, distributed and specialized hardware-based systems. *Journal of Signal Processing Systems*, 61:293–315, 2010. [Cited in pag. 45]

[93] X. Wu, B. Huang, A. Plaza, Y. Li, and C. Wu. Real-time implementation of the pixel purity index algorithm for endmember identification on GPUs. *IEEE Geoscience and Remote Sensing Letters*, in pres, 2013. [Cited in pag. 45]

[94] A. Barberis, G. Danese, F. Leporati, A. Plaza, and E. Torti. Real-time implementation of the Vertex Component Analysis algorithm on GPUs. *IEEE Transactions on Geoscience and Remote Sensing*, 10(2):251–255, 2013. [Cited in pag. 45]

[95] S. Sánchez and A. Plaza. Fast determination of the number of endmembers for real-time hyperspectral unmixing on GPUs. *Journal of Real-Time Image Processing*, accepted for publication, in press, 2014. [Cited in pag. 45]

[96] S. Sánchez, R. Ramalho, L. Sousa, and A. Plaza. Real-time implementation of remotely sensed hyperspectral image unmixing on GPUs. *Journal of Real-Time Image Processing*, accepted for publication, in press, 2014. [Cited in pag. 45]

[97] S. Sánchez, A. Paz, G. Martin, and A. Plaza. Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units. *Concurrency and Computation: Practice and Experience*, 23(13):1538–1557, 2011. [Cited in pag. 45]

[98] C. Gonzalez, J. Resano, A. Plaza, and D. Mozos. FPGA implementation of abundance estimation for spectral unmixing of hyperspectral data using the image space reconstruction algorithm. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(1):248–261, 2012. [Cited in pag. 45]

[99] A. Remon, S. Sánchez, S. Bernabe, E.S. Quintana-Ortí, and A. Plaza. Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms. *EURASIP Journal on Advances in Signal Processing*, accepted for publication, in press, 2014. [Cited in pag. 45]

[100] S. Sánchez and A. Plaza. GPU implementation of the image space reconstruction algorithm for remotely sensed hyperspectral unmixing. In *SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference, San Diego, CA*, 2012. [Cited in pags. 45 and 76]

[101] J. Sevilla, L.I. Jiménez, and A. Plaza. Sparse unmixing-based hyperspectral imagery retrieval system implemented on GPUs. *IEEE Geoscience and Remote Sensing Letters*, Submited. [Cited in pag. 73]

[102] M.-D. Iordache, A. Plaza, and J. Bioucas-Dias. On the use of spectral libraries to perform sparse unmixing of hyperspectral data. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2010 2nd Workshop on*, pages 1–4, June 2010. [Cited in pag. 74]

[103] J.M.P. Nascimento, J.M. Bioucas-Dias, J.M. Rodriguez Alves, V. Silva, and A. Plaza. Parallel hyperspectral unmixing on GPUs. *IEEE Geoscience and Remote Sensing Letters*, 11(3):666–670, March 2014. [Cited in pags. 74 and 80]

[104] J. Sevilla, Y. Julien, G. Sória, J.A. Sobrino, and A. Plaza. A new geo-portal for MODIS/SEVIRI image products with geolocation-based retrieval functionality. *Journal of Applied Remote Sensing*, Submited. [Cited in pag. 87]

[105] J.A. Sobrino and Y. Julien. Trend analysis of global MODIS-Terra vegetation indices and land surface temperature between 2000 and 2011. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(5):2139–2145, Oct 2013. [Cited in pag. 88]

[106] J.A. Sobrino and M. Romaguera. Water vapour retrieval from Meteosat 8/SEVIRI observations. *International Journal of Remote Sensing*, 29(3):741–754, 2008. [Cited in pags. 88 and 93]

[107] F. Camacho de Coca, J.C. Jimenez Munoz, B. Martinez, P. Bicheron, R. Lacaze, and M. Leroy. Prototyping of cover product over Africa based on existing CYCLOPES and JRC products for VGT4Africa. *Proceedings of the 2nd Workshop on Recent Advances in Quantitative Remote Sensing (RAQRS)*, 1:722–727, 2006. [Cited in pags. 88 and 91]

[108] B. Tan, J.T. Morisette, R.E. Wolfe, F. Gao, G.A. Ederer, J. Nightingale, and J.A. Pedelty. An enhanced TIMESAT algorithm for estimating vegetation phenology metrics from MODIS data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(2):361–371, June 2011. [Cited in pag. 88]

[109] L. Lei, Z. Zeng, and B. Zhang. Method for detecting snow lines from MODIS data and assessment of changes in the Nianqingtanglha mountains of the Tibet plateau. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(3):769–776, June 2012. [Cited in pag. 88]

[110] T.-Y. Chang, Y.-C. Wang, C.-C. Feng, A.D. Ziegler, T.W. Giambelluca, and Y.-A. Liou. Estimation of root zone soil moisture using apparent thermal inertia with MODIS imagery over a tropical catchment in northern Thailand. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(3):752–761, June 2012. [Cited in pag. 88]

[111] J. Knight and M. Voth. Mapping impervious cover using multi-temporal MODIS NDVI data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(2):303–309, June 2011. [Cited in pag. 88]

[112] N. Torbick, W.A. Salas, S. Hagen, and X. Xiao. Monitoring rice agriculture in the Sacramento Valley, USA with multitemporal PALSAR and MODIS imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(2):451–457, June 2011. [Cited in pag. 88]

[113] F. Tian, G.Y. Qiu, Y.H. Yang, Y.J. Xiong, and P. Wang. Studies on the relationships between land surface temperature and environmental factors in an inland river catchment based on geographically weighted regression and MODIS data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(3):687–698, June 2012. [Cited in pag. 88]

[114] D. Sun, Y. Yu, and M.D. Goldberg. Deriving water fraction and flood maps from MODIS images using a decision tree approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(4):814–825, Dec 2011. [Cited in pag. 88]

[115] A.S. Ardakani, M.J.V. Zoej, A. Mohammadzadeh, and A. Mansourian. Spatial and temporal analysis of fires detected by MODIS data in northern Iran from 2001 to 2008. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(1):216–225, March 2011. [Cited in pag. 88]

[116] B. Geiger, D. Carrer, L. Franchisteguy, J.-L Roujean, and C. Meurey. Land surface albedo derived on a daily basis from Meteosat Second Generation observations. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3841–3856, Nov 2008. [Cited in pag. 88]

[117] M.O. Rasmussen, F.-M. Gottsche, F.-S. Olesen, and I. Sandholt. Directional effects on land surface temperature estimation from Meteosat Second Generation for Savanna landscapes. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4458–4468, Nov 2011. [Cited in pag. 88]

[118] B. Hirn, C. Di Bartola, and F. Ferrucci. Combined use of SEVIRI and MODIS for detecting, measuring, and monitoring active lava flows at erupting volcanoes. *IEEE Transactions on Geoscience and Remote Sensing*, 47(8):2923–2930, Aug 2009. [Cited in pag. 88]

[119] J.A. Sobrino, Y. Julien, and G. Soria. Phenology estimation from Meteosat Second Generation data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1653–1659, June 2013. [Cited in pag. 88]

[120] N. Jaiswal and C.M. Kishtawal. Objective detection of center of tropical cyclone in remotely sensed infrared images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):1031–1035, April 2013. [Cited in pag. 88]

[121] R.M.A. Timmermans, A.J. Segers, P.J.H. Builtjes, R. Vautard, R. Siddans, H. Elbern, S.A.T. Tjemkes, and M. Schaap. The added value of a proposed satellite imager for ground level particulate matter analyses and forecasts. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2(4):271–283, Dec 2009. [Cited in pag. 88]

[122] S. Liang, K. Wang, X. Zhang, and M. Wild. Review on estimation of land surface radiation and energy budgets from ground measurement, remote sensing and model simulations. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 3(3):225–240, Sept 2010. [Cited in pag. 88]

[123] Glovis. USGS global visualization viewer. *Available online: http://glovis.usgs.gov/ (accessed on 07th June 2014)*, 2013. [Cited in pag. 88]

[124] MRT. MODIS reprojection tool. *Available online: https://lpdaac.usgs.gov/tools/modis_reprojection_tool (accessed on 07th June 2014)*, 2013. [Cited in pag. 88]

[125] J. El Kharraz. Determinación de la temperatura de la superficie terrestre a partir de datos MODIS. *Ph.D Thesis, University of Valencia, Spain*, 2004. [Cited in pag. 93]

[126] IMAPP. International MODIS/AIRS processing package. *http://eostation.scanex.ru/software.html (accessed on 27th May 2014).* [Cited in pag. 97]

[127] M. Shimrat. Algorithm 112: Position of point relative to polygon. *Communications of the ACM*, 5(8):434–, 1962. [Cited in pag. 103]