DEPARTAMENTO DE TECNOLOGÍA DE LOS COMPUTADORES Y DE LAS COMUNICACIONES

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS INFORMÁTICOS Y TELEMÁTICOS

UNIVERSIDAD DE EXTREMADURA

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF ICELAND

Técnicas eficientes para extracción de información en imágenes obtenidas de forma remota

Efficient algorithms for information retrieval from remote sensing images

Tesis Doctoral

Sergio Bernabé García

2013

DEPARTAMENTO DE TECNOLOGÍA DE LOS COMPUTADORES Y DE LAS COMUNICACIONES

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS INFORMÁTICOS Y TELEMÁTICOS

UNIVERSIDAD DE EXTREMADURA

FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

UNIVERSITY OF ICELAND

Tesis Doctoral

Author: Sergio Bernabé García

Advisors: Antonio Plaza Miguel (Univ. of Extremadura)
Pablo García Rodríguez (Univ. of Extremadura)
Jón Atli Benediktsson (Univ. of Iceland)

Conformity of the advisors:

Fdo: Dr. Antonio Plaza Miguel	Fdo: Dr. Pablo García Rodríguez	Fdo: Dr. Jón Atli Benediktsson

Resumen

Las principales contribuciones del presente trabajo de tesis doctoral vienen dadas en primer lugar por la propuesta de nuevos algoritmos paralelos para la recuperación de información en imágenes obtenidas mediante sensores de observación remota de la Tierra. Dichos algoritmos se fundamentan en el problema de la mezcla (problema característico en imágenes de gran dimensionalidad espectral, denominadas hiperespectrales), que permite expresar los píxeles de una imagen como una combinación lineal o no lineal de elementos espectralmente puros, ponderados por sus correspondientes fracciones de abundancia. Una vez descrita la base teórica del estudio, la tesis doctoral presenta una comparativa del uso de diferentes arquitecturas paralelas para abordar el problema de desmezclado espectral o unmixing con las siguientes etapas: 1) estimación automática del número de endmembers en una imagen, 2) identificación automática de dichos endmembers, y 3) estimación de la abundancia de cada endmember en cada píxel de la imagen.

Con vistas a resolver el problema del desmezclado espectral en imágenes de satélite con baja resolución espectral (generalmente denominadas multiespectrales), se ha propuesto un esquema para expandir la dimensionalidad de dichas imágenes mediante la inclusión de información espectral y espacial (utilizando técnicas de morfología matemática). Posteriormente, el resultado es usado como entrada para mejorar la interpretación de la imagen con respecto al caso en que solamente se utiliza la información espectral original. Finalmente, el presente trabajo de tesis doctoral integra las técnicas anteriormente desarrolladas en un sistema de información para realizar clasificación de imágenes obtenidas mediante sensores aerotransportados o satélites de observación remota de la Tierra usando una cadena de procesamiento basada en 3 etapas: 1) pre-procesamiento, 2) análisis e interpretación de datos, y 3) post-procesamiento. El diseño inicial en forma de aplicación de escritorio ha sido mejorado tras la utilización de un servidor remoto y el uso de tecnologías web.

Con vistas a validar todas las contribuciones algorítmicas anteriormente mencionadas, se ha realizado un detallado estudio cuantitativo y comparativo en cuanto a precisión y rendimiento computacional de cada una de las técnicas descritas, haciendo uso de imágenes hiperespectrales obtenidas por los sensores Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), Reflective Optics Spectrographic Imaging System (ROSIS) e Hyperion de NASA en el contexto de varias aplicaciones reales, utilizando también imágenes de gran resolución espacial pero con limitada resolución espectral para validar las técnicas de análisis de imágenes multiespectrales. Por otra parte, la validación del sistema de información se ha realizado utilizando el repositorio de imágenes de Google Maps.

Palabras clave: análisis hiperespectral y multiespectral, teledetección, procesamiento de imágenes, desmezclado espectral, matemática morfológica, sistemas de información y recuperación de imágenes basada en cotenido (CBIR).

Abstract

The main contributions of the present thesis work are given first, by the proposal of new parallel algorithms for information retrieval from remotely sensed images of the surface of the Earth. These algorithms are based on the unmixing problem (characteristic of remotely sensed images with high spectral resolution), which allows to express the pixels of an image as a linear or nonlinear combination of spectrally pure elements, weighted by their corresponding abundance fractions. Once the theoretical foundations of the proposed study are described, the thesis work presents a comparison of different parallel architectures to address the spectral unmixing or unmixing problem with the following steps: 1) automatic identification of the number of endmembers in the image; 2) automatic extraction of endmember signatures; and 3) estimation of the fractional abundance of endmembers on a sub-pixel basis.

In order to solve the spectral unmixing problem in satellite images with low spectral resolution (usually called multispectral), a methodology to expand the dimensionality of such images has been developed by including spectral and spatial information (using mathematical morphology concepts). Subsequently, the result is used as input to improve the interpretation of the image with respect to the case where only the original spectral information is used. Finally, the present thesis work integrates the techniques previously developed on an information system for classification of images obtained by satellite or airborne sensors for remote sensing of the Earth using a processing chain based on three steps: 1) pre-processing; 2) data interpretation and analysis; and 3) post-processing. The initial design as a desktop application has been improved through the use of a remote server using web technologies.

In order to validate all the aforementioned techniques, a detailed quantitative and comparative study in terms of accuracy and computational performance has been conducted for each of the techniques described, using hyperspectral imaging obtained by the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), Reflective Optics Spectrographic Imaging System (ROSIS) and Hyperion of NASA sensors in the context of several real applications, also using images with high spatial resolution and limited spectral resolution to validate techniques for analysis multispectral images. Moreover, the validation of the information system has been performed using the Google Maps repository.

Keywords: hyperspectral and multispectral imaging, remote sensing, image processing, spectral unmixing, mathematical morphology, information systems, content-based image retrieval (CBIR).

Nuestra recompensa se encuentra
en el esfuerzo y no en el resultado.
Un esfuerzo total es una victoria completa.

Mahatma Gandhi

Pueden parecer pobres nuestras
reflexiones ante los demás, aún
sin serlo, pero tal juicio no alivia
la carga del esfuerzo que cuesta alcanzarlas.

José Vasconcelos

Agradecimientos

Es difícil de entender la importancia de los agradecimientos de una tesis doctoral hasta que no se ha terminado. En ese momento te das cuenta de cuánto tienes que agradecer a tanta gente. He de sintetizar en unas breves líneas mi sentida y sincera gratitud hacia las personas que me han ayudado durante esta etapa, haciendo posible que hoy deje de ser un sueño para pasar a ser una realidad.

Al **Dr. Antonio Plaza**, director de esta tesis, por toda su ayuda y sabios consejos. Le agradezco que me haya abierto las puertas de su gran grupo de investigación, dándome la oportunidad de tener una visión amplia de este mundo y descubrir cuánto me motiva. Por ser el principal responsable de que este trabajo llegara a buen puerto, estando incondicionalmente siempre a mi lado, en los buenos y malos momentos, animándome siempre a continuar. Le doy las gracias por todos los esfuerzos que ha hecho, por haberme hecho creer cada día que podía hacerlo. Agradezco también al **Dr. Pablo García-Rodríguez**, persona importante en este trabajo, siendo co-director de esta tesis, por su disponibilidad y colaboración en todo momento, teniendo siempre la puerta abierta para resolver dudas y solucionar todo tipo de problemas que han surgido, demostrando en todo momento que es un amigo al que siempre podré recurrir. Por todo esto y mucho más este trabajo también les pertenece.

Mi gratitud, para el **Dr. Jón Atli Benediktsson**, co-director de esta tesis, por haberme abierto las puertas de la Universidad de Islandia, además de haberme dado la oportunidad de conocerle como persona y conocer su país, Islandia, del que me traje muy buenos recuerdos. Nunca podré corresponder como merecería tan buen trato, tantos conocimientos y sabiduría empleados en mi formación. Gracias, por ser un verdadero maestro.

Deseo también expresar mi agradecimiento a mis compañeros, con los que he compartido mi espacio de trabajo, **Grupo Hypercomp**, por su inestimable ayuda incondicional, para la elaboración de este trabajo, tanto a los que ya estaban cuando llegué (Javi, Abel, Sergio, Gabri, Inma, Daniel, J. Li, ...) y también aquellos que llegaron dando un aire fresco al grupo (Jorge, Nacho, José Manuel, Benquin, Mahdi, ...).

No me puedo olvidar tampoco de las personas con las que he ido trabajando y compartiendo grandes momentos en mis distintas estancias de investigación (**Canarias 2011**, **Islandia 2012** y **Brasil 2013**). De todas ellas, me llevo grandes recuerdos y bonitas experiencias.

A mi **familia**, a mis padres y hermanos por haberme enseñado que la vida es para los valientes, apoyándome en todas las decisiones que he tomado a lo largo de la vida, hayan sido buenas o malas, y especialmente por enseñarme a luchar por lo que quiero y a terminar lo que he empezado. Sin ellos nunca habría terminado esta Tesis Doctoral.

Por último, quiero expresar mi más especial agradecimiento a mi novia **Sheila**, no solo por su amor, sino también por estar conmigo en cada instante apoyando y empujándome en los momentos difíciles y por hacer realidad este sueño que empezó hace unos cuantos años y que también te pertenece. Sin ti no lo hubiera cumplido, te quiero.

Y como cierre de estos agradecimientos, a todas las personas que, aunque no aparecen aquí con nombres y apellidos, han estado presentes de alguna forma durante el desarrollo de este trabajo y han hecho posible que hoy vea la luz, expresarles mi eterno agradecimiento.

To Sheila

Contents

List of Figures

Table Index

Chapter 1

Introduction

1.1 Context and motivations

The work developed in this thesis is part of the actual research lines of the Hyperspectral Computing Laboratory (HyperComp) group coordinated by Prof. Antonio Plaza Miguel at the Department of Technology of Computers and Communications, University of Extremadura, who has served as advisor of this work together with Prof. Pablo García Rodríguez, with the Group of Media Engineering (GIM) at the Department of Systems Engineering and Telematics, University of Extremadura, and Prof. Jón Atli Benediktsson, with the Faculty of Electrical and Computer Engineering, at the University of Iceland. This work focuses on the development of new methodologies and parallel techniques for efficient information extraction and classification of large repositories of images obtained by remote sensing instruments for Earth observation, installed on satellite or airborne platforms.

The availability of advanced techniques for the analysis and interpretation of remotely sensed images, as well as the availability of advanced computing techniques and platforms, has resulted in an exponential growth of the community of remote sensing users and practitioners [2, 3]. In particular, the analysis and interpretation of these images can now effectively exploit both the spectral resolution (which is related to the possibility of an instrument to collect narrow spectral channels), and the spatial resolution (which is related to the pixel size that the imaging instrument is able to collect, based on factors such as altitude, angle of view, scanning speed and optical features). For these reasons, and because of the particularities of the plethora of sensors currently available for Earth observation, different types of airborne or satellite images with different spectral and spatial resolutions are now widely available (see Fig. 1.1). In this thesis, we will particularly focus on two main types of images: *multispectral* (generally with high spatial and low spectral resolution) and *hyperspectral* (generally with high spectral and lower spatial resolution).

One the main problems which arise when extracting useful information from remotely sensed images is the varying spectral and spatial resolutions of the scenes provided by different instruments. In addition, there is a lack of automatic or semiautomatic techniques for efficient feature extraction and subsequent classification of images available in large repositories. In particular, the availability of large remote sensing digital repositories such as Google Maps, which now provides very high resolution images with almost complete coverage of our planet, has opened the attractive perspective of performing advanced classification, interpretation and content-based retrieval tasks using large-scale data repositories. In fact, the combination of digital repositories such as Google Maps and computationally efficient techniques for advanced image analysis and interpretation [4] has attracted a significant interest in the development of sophisticated data processing techniques in the context of many different applications.

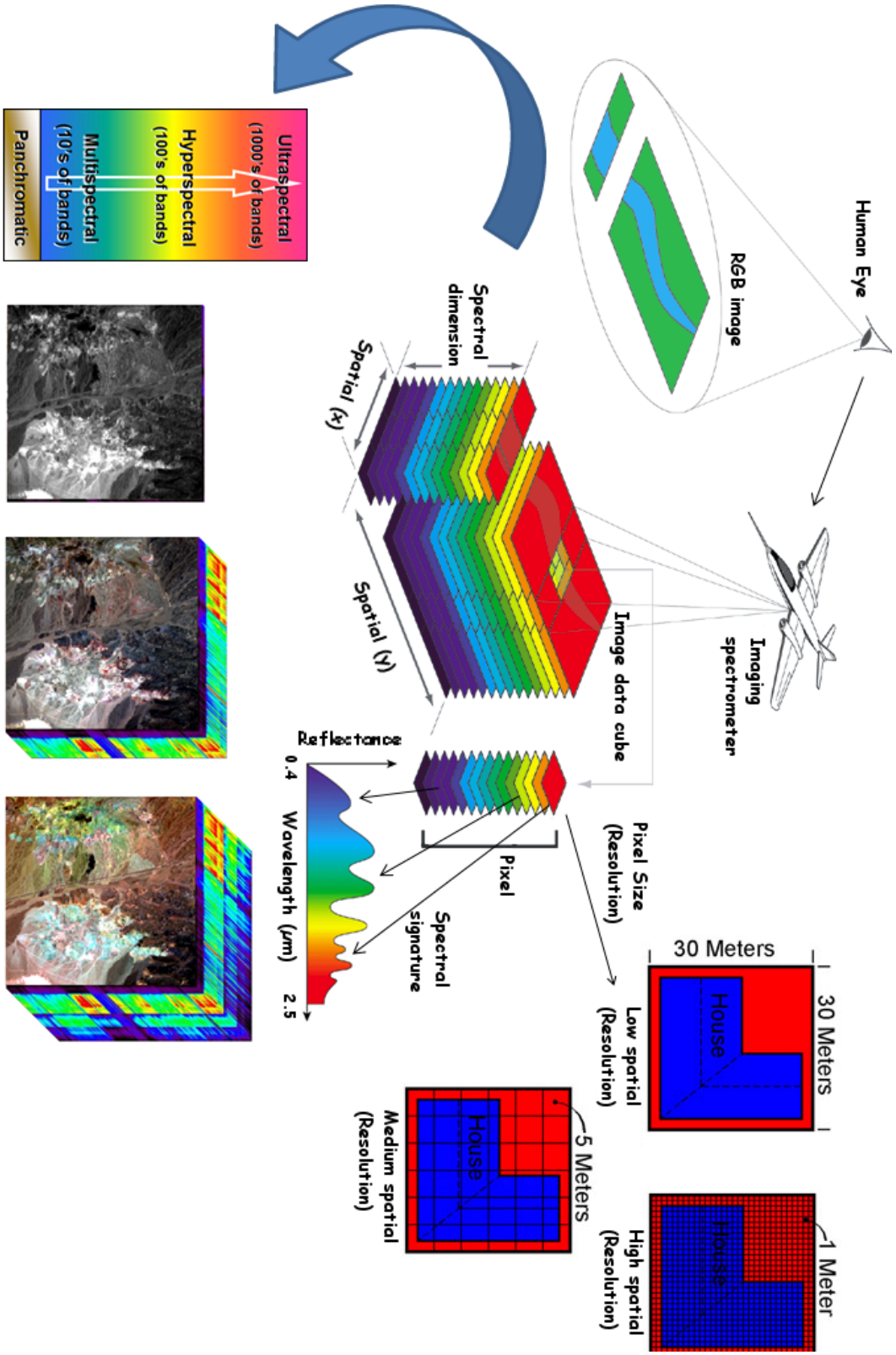


Figure 1.1: The importance of spectral and spatial resolution in remote sensing images.

On the other hand, the analysis of remotely sensed scenes with high spatial resolution but more limited spectral resolution (often called multispectral) is also subject to potential improvements. In this case, the limited spectral information has traditionally hindered the exploitation of techniques which have been quite successful in the analysis of hyperspectral images, e.g., *classification* or *spectral unmixing*. In this regard, the possibility of applying techniques that successfully integrate and exploit both spectral and spatial information is very interesting in the context of multispectral imaging. To address this problem, in this thesis new techniques able to expand the dimensionality of multispectral images have been developed using spatial information (through mathematical morphology concepts [5, 6]), so as to significantly improve the interpretation of such scenes, bringing them to the domain of hyperspectral image analysis in which many successful analysis techniques have been developed in recent years.

It should be noted that currently there are many satellite image repositories. However, Google Maps is the only one that provides high spatial resolution, advanced navigation mapping, adaptivity to different views and layers for desktop and web applications, and most importantly a wide availability worldwide. Despite these advanced aspects, there are other features that are not yet implemented in systems such as Google Maps, e.g., advanced techniques for the analysis and interpretation of the data at different zoom levels, with the incorporation of techniques for advanced classification (both supervised and unsupervised), spectral unmixing, etc. [7–12]. The incorporation of such features in Google Maps could allow information retrieval from large databases of satellite images and the ability to perform content-based image retrieval [13–15], tasks which are of great interest for the advanced exploitation of remotely sensed imagery.

An important aspect in the aforementioned goal is the fact that capturing and processing a large amount of high-dimensional images available from state-of-the-art Earth observation platforms involves the collection and interpretation of significant amounts of high dimensional data that must be stored and processed efficiently [16]. However, in certain applications it is desirable to perform the analysis of such data quickly enough for practical use [17–21]. From a computational standpoint, many image analysis algorithms for remote sensing data exploitation present parallelism at multiple levels: pixel-based (coarse grain parallelism pixel level), through spectral information (fine grain parallelism spectral level), and even through different tasks that can be completed in parallel. Considering that data accesses in these algorithms are often regular and predictable, it is feasible to assume that these algorithms can be efficiently implemented [22] and parallelized on high performance computing systems [23].

In recent years, there have been numerous studies demonstrating that the processing of high-dimensional remote sensing scenes (such as hyperspectral data, with high spectral resolution) can be efficiently performed using clusters of computers and heterogeneous networks of workstations [24, 25]. These systems are generally expensive and difficult to adapt to onboard data processing scenarios needed in real-time applications such as fire monitoring, detecting biological threats, detection of oil spills and other types of chemical contamination. In order to investigate the efficient implementation of remote sensing data processing algorithms, such as spectral unmixing [17, 26], other platforms with low power consumption have been investigated, most notably FPGAs (field programmable gate arrays) [27–29] are very important to reduce the weight of the sensor, but generally require a significant effort from the viewpoint of design and programming. Thus, more realistic and domestic alternatives are multi-core processors [30–32] and graphics processing units (GPUs) [33, 34], which both offer significant computing power at low cost, providing the opportunity to bridge the gap towards (near) real-time exploitation of high-dimensional remote sensing scenes.

As mentioned above, in this thesis our efforts have been primarily focused on the development of new techniques for remote sensing data exploitation, with particular attention to the efficient implementation

of such techniques on a variety of parallel computing architectures, and considering different types of remotely sensed scenes with high spectral resolution and more limited spatial resolution (hyperspectral), and also with high spatial resolution and more limited spectral resolution (multispectral). Taking advantage of the newly developed techniques, we have also given a first step towards the incorporation of these techniques into advanced information systems with large remote sensing data repositories such as Google Maps, which offers an unprecedented opportunity to exploit the newly developed approaches in a wider context that is highly accessible worldwide. It is important to emphasize that the analysis techniques developed in the framework of this thesis can be extrapolated to other application areas, such as medical applications. In this regard, another interesting result of this present thesis lies in the possibility of adapting the techniques developed to different applications in addition to remote sensing, which has been used in the present work as a practical case for illustrating the different techniques proposed.

1.2 Objectives and novel contributions of the thesis

The main objective of the thesis work is to develop new (computationally efficient) algorithms and techniques for efficient information extraction and classification of large remote sensing data repositories, and the integration of these techniques on a system for efficient data analysis and retrieval. In order to achieve this general objective, we will address a number of specific objectives which are listed below:

1. To develop new techniques for the advanced analysis of remote sensing images with high spatial resolution but moderate spectral resolution (called multispectral), evaluating their advantages, disadvantages and computational requirements in order to achieve relevant analysis results. Particular attention will be given to the development of mechanisms able to expand the dimensionality of these scenes by taking advantage of their fine spatial resolution in order to complement their more limited spectral resolution (through the use of mathematical morphology-based approaches). Attention will be also given to the efficient implementation of these techniques on parallel computing architectures.

2. To develop new algorithms for the advanced analysis of remote sensing images with high spectral resolution (called hyperspectral), using techniques based on spectral unmixing concepts and with particular attention to their efficient implementation on parallel systems such as multi-core processors and graphics processing units (GPUs), obtaining new unmixing-based processing chains for remote sensing data exploitation which could be used in real missions for remote Earth observation.

3. To design and implement an advanced information extraction system for remotely sensed images obtained through remote sensing Earth observation platforms, taking advantage of the computationally efficient algorithms described above and using both desktop and web-based design strategies in order to fully exploit the capacities of the newly design system and make it widely available to the scientific community. When designing this system, particular attention will be given to the integration of the aforementioned techniques in the considered platform.

4. To perform a detailed quantitative and comparative study (in terms of analysis accuracy and computational performance) of the newly techniques developed, using hyperspectral images (e.g., data sets collected by NASA's AVIRIS sensor or the EO-1 Hyperion instrument) and also

multispectral images (e.g., satellite images from the Google Maps repository and scenes collected by instruments with high spatial resolution and moderate spectral resolution such as IKONOS).

1.3 Thesis organization

Fig. 1.2 graphically represents the main contributions in this work and their relationship. As shown by this figure, we particularly focus on the problem of spectral unmixing of hyperspectral images and the classification of multispectral images. As mentioned before, another important goal of this thesis work is to include all the developed techniques into an advanced automatic system for information extraction and classification of large remote sensing data repositories. In the following, the main contributions of the different chapters that compose this thesis is summarized. In fact, the remainder of the thesis can be viewed as a combination of three different types of contributions: *algorithms* (Chapter 2), *efficient implementations* (Chapters 3 and 4) and *systems* (Chapter 5). The thesis concludes with some remarks in Chapter 6.

Chapter 2: Advanced classification of remote sensing images with limited spectral resolution

In this chapter, a novel methodology to incorporate spectral and spatial information into the classification of multispectral images is proposed. Due to the low spectral resolution of these scenes, the classification results obtained may not be comparable with the classification results that can be obtained after processing other images with much better spectral resolutions. In order to address this relevant issue, a new framework is presented in which, first, we propose to use kernel feature extraction as a mechanism to expanded the dimensionality of multispectral data and then extract the spatial information based on the extracted spectral features. The next step is to include spatial information by means of morphological characterization of the expanded set of features. The proposed methodology is tested with different multispectral data sets obtaining state-of-the-art classification results. This chapter is focused on addressing the problems involved with multispectral images with limited spectral resolution.

Chapter 3: Design and implementation of efficient algorithms for analyzing remote sensing images with high spectral resolution

This chapter is dedicated to the design and implementation of efficient algorithms for processing remote sensing images with high spectral resolution, with particular attention to the use of specialized systems for onboard processing such as FPGAs and GPUs. These systems are now used in real missions for remote Earth observation. These algorithms focus on different parts of the spectral unmixing chain. More specifically, alternatives to clusters of computers are proposed, oriented to professional environments such as a cloud system which is part of the NASA SensorWeb suite of web services, as well as FPGA devices using tools such as Mentor Graphics' Catapult to facilitate the work of developers in companies to take advantage of their C/C++ or Matlab codes. On the other hand, more domestic alternatives such as commodity GPUs (also seen in this chapter) or multi-core systems will be described in the next chapter.

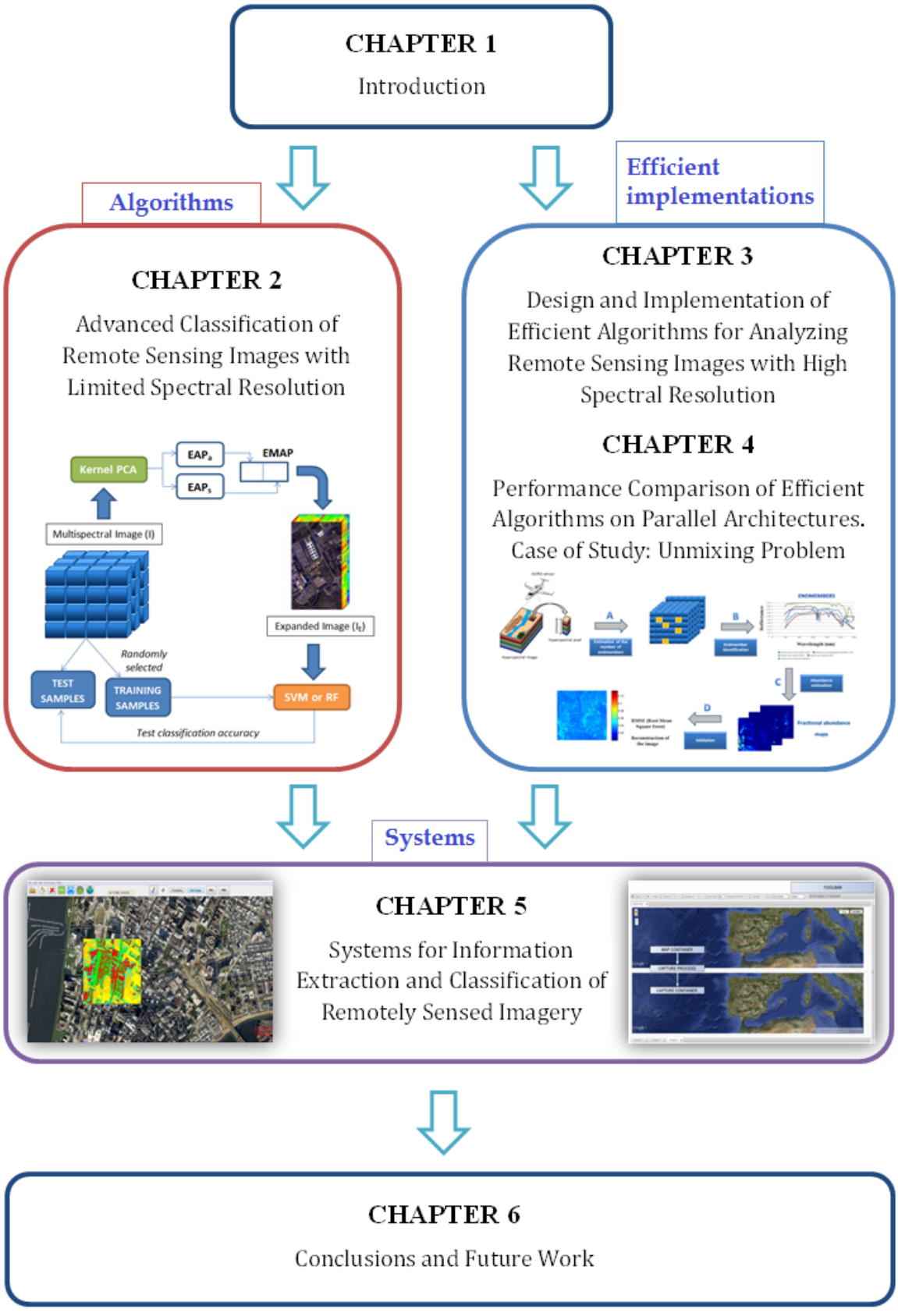


Figure 1.2: Thesis structure.

Chapter 4: Performance comparison of efficient algorithms on parallel architectures. Case of study: unmixing problem

In this chapter, a comparative study between two kinds of architectures which are widely used in domestic environments and accessible to most users in the field of remote sensing is developed: GPUs and multi-core systems. The comparison is focused on one of the most pressing problems with remotely images with high spectral resolution (called hyperspectral). This is the so-called spectral unmixing problem, which we address in this chapter by applying an unmixing chain made up of three steps: 1) automatic identification of the number of endmembers in the image; 2) automatic extraction of the signatures of these endmembers; and 3) estimation of the fractional abundance of endmembers on a sub-pixel basis. This comparative study is required to adapt the best architecture into a system for information extraction and classification of remotely sensed imagery (described in the next chapter), which offers highly relevant computational power at low cost and the opportunity to bridge the gap towards real-time analysis of remotely sensed hyperspectral data.

Chapter 5: Systems for information extraction and classification of remotely sensed imagery

In this chapter, we take a first step towards the development of advanced information extraction and classification systems for efficient remote sensing data retrieval and interpretation. The developed systems can be used with large data repositories such as Google Maps or even applied to collections of images with high spectral resolution and low spatial resolution (hyperspectral images) or to images with high spatial resolution and lower spectral resolution (multispectral images). The systems described in this chapter integrate techniques previously developed in previous chapters using a processing chain based on three steps: 1) pre-processing; 2) data analysis and interpretation; and 3) post-processing. Besides, the possibility to integrate high performance computing systems at low cost, such as GPU-based architectures to perform computationally efficient information extraction from scenes in large data repositories such as Google Maps is expected to bring the exploitation and use of these data to a new level. This is enforced by the incorporation of graphics coprocessors (desktop system) or a remote server (web-based system) for efficient data processing.

Chapter 6: Conclusions and future work

This chapter summarizes the advantages, disadvantages and main contributions introduced by the methodologies, efficient implementations and systems developed in this thesis, as well as with a presentation of the most plausible future research lines that should be explored after this research.

Finally, the thesis concludes with the list of references used during the elaboration of this document. In an appendix the publications resulting from the present thesis are listed, together with a statement on the main contributions and relevant aspects which are highlighted for each individual contribution. Specifically, the results of this thesis work have been published in 8 journal citation reports (JCR) papers and 10 peer-reviewed conference papers, for a total of 18 publications with international scope.

Chapter 2

Advanced Classification of Remote Sensing Images with Limited Spectral Resolution

Outline - Over the last few years, several new strategies have been proposed for spectral-spatial classification of remotely sensed image data, for cases when high spatial and spectral resolutions are available. In this chapter, we focus on the possibility to perform advanced spectral-spatial classification of remote sensing images with limited spectral resolution (often called multispectral). A new strategy is proposed, where the spectral dimensionality of the multispectral data is first expanded by using non-linear feature extraction with kernel methods such as kernel principal component analysis (KPCA). Then, extended multi-attribute profiles (EMAPs), built on the expanded set of spectral features, are used to include spatial information. This strategy allows us to first decompose different spectral clusters into different spectral features and further improve the spatial discrimination. The resulting EMAPs are used for classification using advanced classifiers such as support vector machines (SVMs) and random forests (RFs). The proposed methodology is tested with different multispectral data sets, obtaining state-of-the-art classification results.

2.1 Overview

Over the last few years, many efforts have been directed towards the use of spatial information to refine spectral-based classifiers, assuming high spectral resolution in the remotely sensed data to be processed. For instance, morphological profiles [35] have been widely used for spatial characterization of hyperspectral imagery [8, 36]. Markov random fields (MRFs) have also been employed for spatial characterization in hyperspectral classification, as described in [37, 38], or as a post-processing step as in [39]. Despite of the success of these approaches, fewer efforts have been directed towards exploiting spatial and spectral information in remotely sensed data with limited spectral resolution [40]. In this

Part of this chapter will be published in:

S. Bernabé, P. R. Marpu, A. Plaza, M. Dalla Mura and J. A. Benediktsson, "Spectral-Spatial Classification of Multispectral Images Using Kernel Feature Space Representation", *IEEE Geoscience and Remote Sensing Letters*, in press, 2013 [JCR(2012)=1.823].

context, attribute profiles (APs) [41], and their extension to multi-attribute profiles (EMAPs) [42], have been successfully used for classification of remotely sensed data with more limited spectral resolution [43].

In this chapter, a new strategy to perform spectral-spatial classification of images with low spectral resolution (often called multispectral images) is developed. Multispectral images constitute a very important source of remotely sensed data. The proposed strategy first expands the spectral dimensionality of the data by using non-linear feature extraction with kernel methods, such as kernel principal component analysis (KPCA) [44]. While feature extraction is often performed with hyperspectral data to reduce dimensionality and select relevant features for classification, it is seldom used with lower spectral resolution data, where the multispectral data channels (along with the panchromatic spectral channel) are generally used to extract spatial features. Here, we propose to use kernel feature extraction as a mechanism to expand spectral dimensionality and then extract the spatial information based on the extracted spectral features. The motivation behind this strategy is to first decompose the multispectral data into features representing the inherent spectral clusters, which would increase the contrast between different classes after the kernel transformation. This is due to the fact that this transformation brings the data to a higher dimensional space, in which the data may be easier to separate. Then, EMAPs built on the expanded set of spectral features are used to extract spatial information more optimally prior to classification using advanced techniques such as support vector machines (SVMs) [45] or random forests (RFs) [46]. The proposed strategy is a simple but effective extension of the regular processing chain with EMAPs presented in [42].

The remainder of the chapter is organized as follows. Section 2.2 describes the proposed methodology, which is based on two main ingredients: KPCA and EMAPs. Section 2.3 describes the experimental results. Concluding remarks are given in section 2.4.

2.2 Proposed methodology

2.2.1 Kernel-based feature extraction using KPCA

PCA has been widely used for feature extraction in remote sensing image analysis [47]. The principal components (PCs) of a stochastic multivariate data are calculated based on a linear transformation, which produces uncorrelated data channels of decreasing variance using the covariance matrix as a dispersion matrix. The PCs are calculated so as to maximize the variance in every component. PC components are obtained as the linear combination of the original data channels with maximum variance subject to the constraint that it is uncorrelated with all the other components. For an image of n spectral channels represented as a random vector $\mathbf{G}$ with zero mean, the covariance matrix can be simply calculated as:

$$\Sigma = \left\langle \mathbf{G}\mathbf{G}^T \right\rangle . \tag{2.1}$$

Normally, only the first few principal components account for most of the variance in the data. This fact is used to reduce the dimensionality of the data by considering only the first few principal components, which account for most of the variance.

In turn, the kernel formulation of PCA [44] is obtained by replacing the inner products in the Gram matrix $\left\langle \mathbf{G}\mathbf{G}^T \right\rangle$ with the kernel functions $K(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent data observations. The kernel functions represent the inner products of vectors $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$, which are the non-linear mappings of $\mathbf{x}_i$ and $\mathbf{x}_j$ into a higher dimensional feature space. When there is a large number of observations, as is the case for remote sensing images, these observations are generally subsampled to

perform KPCA. Even that will lead to a high number of features, so only the top few features which account for most of the variance are considered, as mentioned above. Since the data are projected onto a higher dimensional feature space, the clusters may now be easier to discriminate in the kernel feature space [43]. That is the reason why KPCA is chosen as a pre-processing step to decompose the spectral information in the kernel feature space, so that the spectral clusters will have less variance in the obtained features, and may be easier to separate in the kernel feature space.

2.2.2 Spatial characterization using EMAPs

EMAPs [42] are an extension of APs [41] obtained using different types of attributes and stacked together. The filtering operation implemented in APs is based on the evaluation of how a generic attribute $\mathcal{A}$, computed for all the connected components of a scalar image, compares to a given reference value λ in a binary predicate $\mathcal{P}$ (e.g., $\mathcal{P} := \mathcal{A}(C_i) > \lambda$, with C_i being the $i-$th connected component of the image). If $\mathcal{P}$ holds true then the region is kept unaltered, otherwise it is set to the grayscale value of the adjacent region with closer value, thereby merging the connected components. When the region is merged to the adjacent region of a lower (or greater) gray level, the operation performed is a thinning (or thickening). Given a sequence of ordered threshold values $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, an AP is obtained by applying a sequence of attribute thinning and attribute thickening operations as indicated in (2.2), where ϕ_i and γ_i respectively denote the thickening and thinning transformations, and $f_j(\mathbf{x}_i)$ denotes a feature extracted from the original observation $\mathbf{x}_i$:

$$\mathrm{AP}(f_j(\mathbf{x}_i)) := \{\phi_n(f_j(\mathbf{x}_i)), \ldots, \phi_1(f_j(\mathbf{x}_i)), f_j(\mathbf{x}_i), \gamma_1(f_j(\mathbf{x}_i)), \ldots, \gamma_n(f_j(\mathbf{x}_i))\} \tag{2.2}$$

In [36], it was suggested to use several PCs of original data to address this issue. In this way, APs are built on each of the first q PCs. This leads to the following definition of the extended attribute profile (EAP) for the observation $\mathbf{x}_i$:

$$\mathrm{EAP}(\mathbf{x}_i) := \{\mathrm{AP}(f_1(\mathbf{x}_i)), \mathrm{AP}(f_2(\mathbf{x}_i)), \cdots, \mathrm{AP}(f_q(\mathbf{x}_i))\}, \tag{2.3}$$

where q is the number of retained features. From the EAP definition in (2.3), the consideration of multiple attributes leads to the concept of EMAP, which improves the capability to extract the spatial characteristics of the structures in the scene. An increase in the dimensionality of the data is also obtained. In this chapter, only two attributes are used: area and standard deviation of the pixel values. This specific choice was supported by previous works (e.g., [43]) in which these attributes were proved to effectively extract the spatial characteristics of the regions. However, it is important to note that any measure computed on image regions may be considered as an attribute, and can be used depending on the application. We refer to [41, 42] (and references herein) for additional details on APs and EMAPs, their implementation details and their computational complexity.

2.2.3 Multispectral image classification framework

The proposed framework for multispectral image classification is summarized by the flowchart in Fig. 2.1. First, KPCA is used to extract the features and to expand the dimensionality of the original multispectral image with n data channels. In this way, the contrast between different spectral clusters increases, but the variance within the cluster decreases in the corresponding KPCA component representative of the

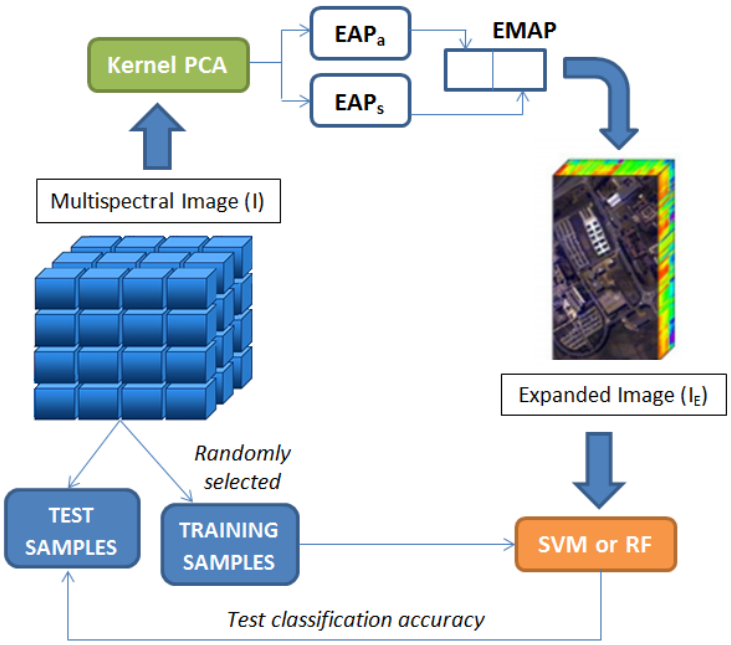


Figure 2.1: Proposed framework for multispectral image classification.

cluster. The second step is to build EMAPs based on the features derived using KPCA, exploiting the spatial information. This strategy will increase the dimensionality of the data. However, it embeds an intelligent combination of spatial and spectral information. Classification is finally performed on the obtained EMAPs using a non-linear classifier such as the SVM or RF. It has been observed in [43,48] that SVMs are more sensitive than RFs to the Hughes phenomenon [47] when dealing with high dimensional data such as EMAPs, whereas RFs provide consistent and better performance. Both SVMs and RFs are used in this chapter for comparative purposes.

2.3 Experimental results

In this section, a quantitative and comparative assessment of the proposed methodology is presented using multispectral data. The main goal is to incorporate spectral and spatial information in an effective way to improve the multispectral image classification results. The combination of feature extraction using KPCA and spatial characterization using EMAPs provides a processing approach that has not been explored in previous contributions focused on multispectral data. Before describing the results obtained in the experiments, we first describe the data sets and experimental setup considered in the experiments. Then, we continue with a discussion on the classification results obtained for the different approaches compared in this chapter.

2.3.1 Data set description

The experimental analysis was carried using three multispectral images, two of them are obtained by extracting the red, blue and green (RGB) data channels from two well-known hyperspectral images. The reasons for choosing the RGB images from hyperspectral images are twofold. First, this allows us to use the highly reliable reference data available for these scenes. Second, this also allows us to compare the

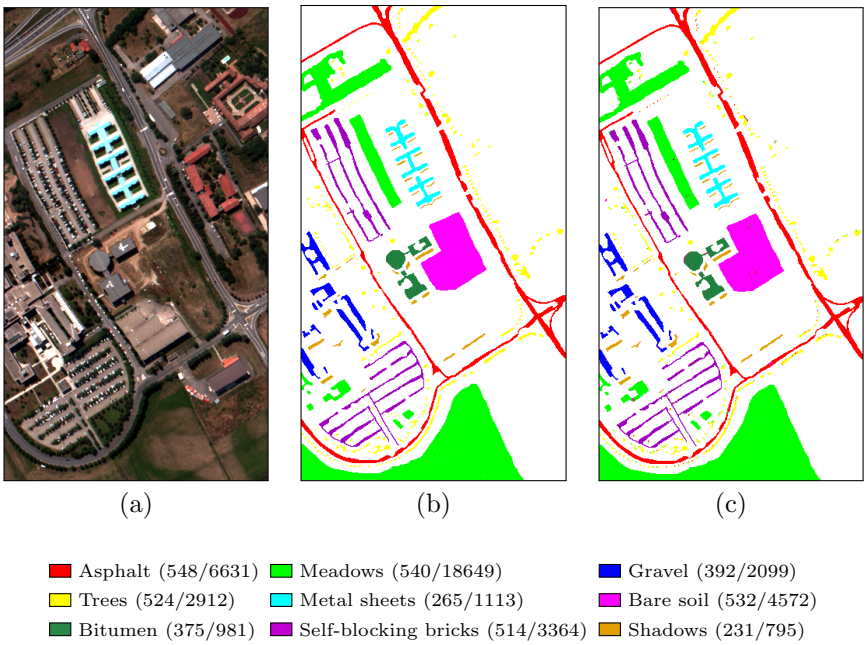


Figure 2.2: (a) RGB composite of the hyperspectral ROSIS Pavia scene. (b) Reference map for the ROSIS Pavia University data with nine reference classes. (c) Classification result with EMAP(KPCA) using 5% training and the RF classifier: 98.81% accuracy. The numbers in the parentheses represent the number of (training/test) pixels available for each class.

results obtained with the original hyperspectral data with much higher spectral resolution. The third image considered in experiments is a multispectral image acquired using the IKONOS instrument. This scene is used to validate the presented methodology with real multispectral data.

- The first image used in experiments was derived from a hyperspectral image acquired using the Reflective Optics System Imaging Spectrometer (ROSIS) sensor over the University of Pavia, Italy. For this scene, with 610×340 pixels and spatial resolution of 1.3 m/pixel, three spectral channels (12, 26 and 51) corresponding to the RGB channels have been selected, out of the 103 spectral channels (covering the wavelength range from 0.4 to 0.9 μm available in the original image. Fig. 2.2(b) shows the reference map available for the scene, which comprises urban features as well as soil and vegetation features.

- The second image used in experiments was derived from a hyperspectral image acquired using the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) sensor over the Indian Pines region in Northwestern Indiana in 1992. This scene, with a size of 145×145 pixels and 202 spectral channels in the range from 0.4 to 2.5 μm, was acquired over a mixed agricultural/forest area, early in the growing season. In this scene, three spectral channels (5, 12 and 24) corresponding to the RGB data channels have been selected, to form a multispectral (color) image. The spatial resolution of the scene is 20 m/pixel. Fig. 2.3(b) shows the reference map available for this scene[1].

- The third image used in experiments was derived from a multispectral image acquired by the IKONOS satellite on August 9, 2001, over a urban area of Reykjavik, Iceland. This scene is

[1] http://dynamo.ecn.purdue.edu/biehl/MultiSpec

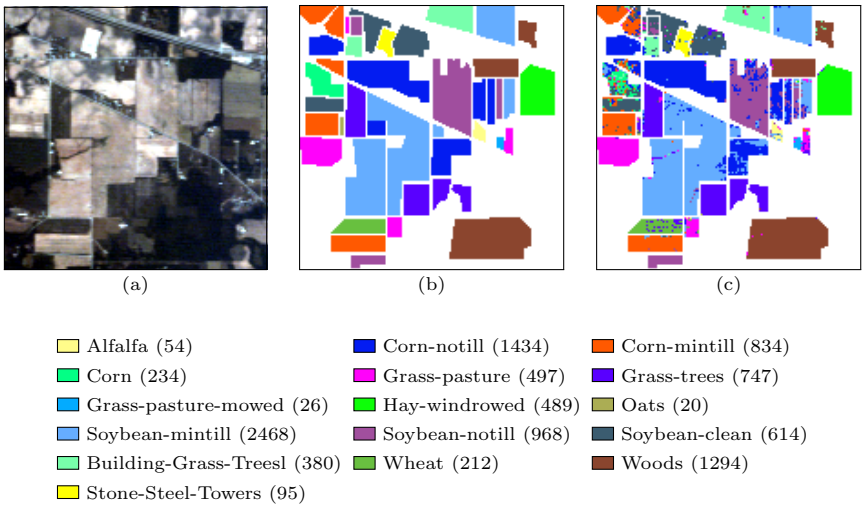


Figure 2.3: (a) RGB composite of the hyperspectral AVIRIS Indian Pines scene. (b) Reference map for the AVIRIS Indian Pines data with sixteen reference classes. (c) Classification result with EMAP(KPCA) using 5% training and the RF classifier: 88.74% accuracy.

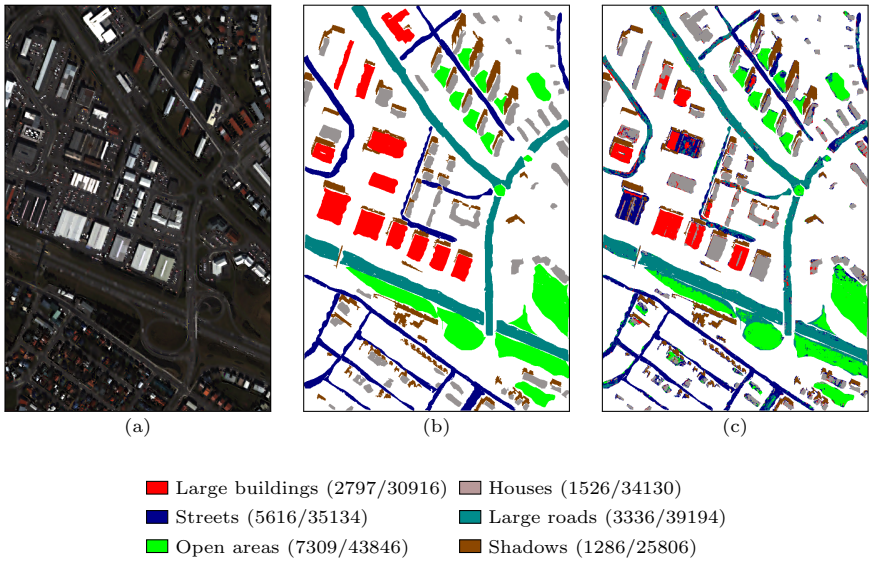


Figure 2.4: (a) RGB composite of the Multispectral IKONOS Reykjavik scene. (b) Reference map for the IKONOS Reykjavik data with six reference classes. (c) Classification result with EMAP(KPCA) using the fixed training and the SVM classifier: 72.02% accuracy.

composed of a size of 975×637 pixels and 4 spectral channels and the spatial resolution is 1 m/pixel. Fig. 2.4(b) shows the reference map available for the scene. This data set is used to evaluate the proposed methodology using a real multispectral data set with standard broad data channels (as opposed to the other two data sets, which are obtained from hyperspectral scenes with narrow spectral channels).

2.3.2 Experimental setup

For the KPCA stage, the Gaussian Radial Basis Function (RBF) kernel was used: $K(\mathbf{x}_i, \mathbf{x}_j) := \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$ which is widely used in hyperspectral image classification. Three different values were chosen for parameter σ in order to study the impact of this parameter on the results. The values chosen were $\sigma = \{1.0, 1.5, 2.0\}$ times the mean value of the mutual distances between 2000 randomly chosen pixels. The corresponding KPCA features are referred to as $\text{KPCA}_{\sigma=1.0}$, $\text{KPCA}_{\sigma=1.5}$, and $\text{KPCA}_{\sigma=2.0}$, respectively.

The EMAPs are built using the area (related to the size of the regions) and standard deviation (which measures the homogeneity of the pixels enclosed by the regions) attributes. The threshold values λ were chosen in the range $\{50, 500\}$ with a stepwise increment of 50 for the area attribute. For the standard deviation, attribute values ranging from 2.5% to 20% of the mean of the feature were chosen with a stepwise increment of 2.5% [48]. These values are selected to accommodate the possible characterization of connected components of the classes of interest.

Then, a supervised classification process was performed using both the SVM classifier (with the RBF kernel) and the RF classifier. The parameters of the SVM were tuned by using 5-fold cross validation, while RF does not require any parameter tuning. This is because every individual decision tree is built to over-fit, which means that no parameters are required to facilitate the pruning of the trees. Classification was performed using three different configurations of the training process in each dataset:

- For the ROSIS Pavia University data set, we considered: 1) a standard training set widely used in the state-of-the-art (referred to hereinafter as *standard training set*); 2) a subset of the entire reference set in Fig. 2.2(b), where 50 pixels are randomly sampled for each reference class; and 3) a subset of the entire reference set, where 5% of the pixels in each class are randomly sampled.
- For the AVIRIS Indian Pines data, we randomly sampled 5%, 10% and 15% of the reference data in Fig. 2.3(b).
- For the IKONOS Reykjavik data, a subset of the entire reference set shown in Fig. 2.4(b) was considered.

KPCA was used to expand the dimensionality of the multispectral data prior to the construction of the EMAPs. Then, only the top 20 features resulting from KPCA (which in all cases comprise more than 99% of the data variance) were used, and the final obtained EMAPs consisted of 740 features (10 levels of filtering using area attribute resulting in 20 features and 8 levels of filtering for standard deviation attribute resulting in 16 features for each KPCA component, so 20 KPCA components $\times$ [20 area attribute features + 16 standard deviation attribute features + 1 KPCA component] = 740 total features).

2.3.3 Analysis and Discussion of Results

Table 2.1 shows the overall accuracy (in percentage) and the standard deviations (each reported value of accuracy is an average of ten Monte Carlo runs) obtained by the SVM classifier applied to different feature extraction methods for the two considered scenes. Here, the SVM classification results are compared using the multispectral (or RGB) scenes, the features resulting from dimensionality expansion of the multispectral/RGB scenes using KPCA (with different values for parameter σ in the RBF kernel), and the features resulting from applying EMAPs to the first 20 features resulting from KPCA dimensionality expansion of the multispectral data. Similarly, Table 2.2 shows the classification results obtained by the

Table 2.1: Overall accuracy [%] and standard deviation (after 10 Monte Carlo runs) obtained after aplying the SVM classifier to the ROSIS Pavia University, the AVIRIS Indian Pines and the IKONOS Reykjavik data sets using different types of features.

Overall Accuracy	**ROSIS Pavia University**			**AVIRIS Indian Pines**			**IKONOS Reykjavik**
Features	**Standard training set**	**Subset of 50 pixels**	**5% Training**	**5% Training**	**10% Training**	**15% Training**	**Standard training set**
Hyperspectral	80.64% ± 0.00	84.09% ± 2.06	93.38% ± 0.21	75.60% ± 1.15	81.95% ± 0.54	84.46% ± 0.45	–
Multispectral	65.81% ± 0.00	62.67% ± 5.62	78.87% ± 0.40	48.33% ± 0.71	49.63% ± 0.29	49.86% ± 0.38	58.73% ± 0.00
EMAP	76.85% ± 0.00	91.87% ± 0.63	96.89% ± 0.23	**79.63% ± 0.73**	**83.71% ± 0.47**	**85.89% ± 0.44**	65.40% ± 0.00
$KPCA_{\sigma=1.0}$	66.13% ± 0.00	65.60% ± 4.26	78.92% ± 0.26	47.97% ± 1.20	49.50% ± 0.46	49.89% ± 0.17	59.31% ± 0.00
$KPCA_{\sigma=1.5}$	66.01% ± 0.00	63.93% ± 4.61	78.62% ± 0.38	48.83% ± 0.38	49.59% ± 0.38	49.95% ± 0.38	61.33% ± 0.00
$KPCA_{\sigma=2.0}$	67.94% ± 0.00	62.86% ± 3.71	78.61% ± 0.32	48.25% ± 0.62	49.25% ± 0.46	49.70% ± 0.30	61.01% ± 0.00
$EMAP(KPCA_{\sigma=1.0})$	93.41% ± 0.00	**93.22% ± 0.71**	97.18% ± 0.64	62.75% ± 3.72	74.45% ± 0.63	77.32% ± 2.90	70.55% ± 0.00
$EMAP(KPCA_{\sigma=1.5})$	**94.09% ± 0.00**	90.61% ± 2.38	97.18% ± 0.62	63.86% ± 2.74	73.91% ± 2.16	77.34% ± 3.14	71.88% ± 0.00
$EMAP(KPCA_{\sigma=2.0})$	91.87% ± 0.00	91.61% ± 1.36	**97.25% ± 0.63**	61.40% ± 3.14	71.62% ± 3.68	77.86% ± 2.30	**72.02% ± 0.00**

Table 2.2: Overall accuracy [%] and standard deviation (after 10 Monte Carlo runs) obtained after aplying the RF classifier to the ROSIS Pavia University, the AVIRIS Indian Pines and the IKONOS Reykjavik data sets using different types of features.

Overall Accuracy	**ROSIS Pavia University**			**AVIRIS Indian Pines**			**IKONOS Reykjavik**
Features	**Standard training set**	**Subset of 50 pixels**	**5% Training**	**5% Training**	**10% Training**	**15% Training**	**Standard training set**
Hyperspectral	71.42% ± 0.13	72.63% ± 2.66	87.66% ± 0.28	69.84% ± 1.30	75.38% ± 0.56	77.70% ± 0.51	–
Multispectral	64.87% ± 0.11	61.68% ± 2.62	77.08% ± 0.27	44.27% ± 0.54	46.17% ± 0.43	47.05% ± 0.28	61.38% ± 0.09
EMAP	80.81% ± 0.30	88.61% ± 1.38	96.76% ± 0.23	80.93% ± 1.07	84.78% ± 0.49	86.46% ± 0.37	62.07% ± 0.42
$KPCA_{\sigma=1.0}$	65.26% ± 0.11	65.32% ± 2.83	78.06% ± 0.22	44.62% ± 0.62	45.95% ± 0.52	46.30% ± 0.37	60.91% ± 0.07
$KPCA_{\sigma=1.5}$	65.44% ± 0.18	65.64% ± 2.39	78.48% ± 0.26	44.36% ± 0.62	45.83% ± 0.47	46.24% ± 0.53	61.67% ± 0.09
$KPCA_{\sigma=2.0}$	65.97% ± 0.14	65.50% ± 2.59	78.46% ± 0.24	44.95% ± 0.73	46.45% ± 0.47	46.81% ± 0.36	61.69% ± 0.08
$EMAP(KPCA_{\sigma=1.0})$	92.45% ± 0.15	94.72% ± 0.88	98.61% ± 0.13	88.65% ± 0.63	**92.67% ± 0.59**	**94.25% ± 0.42**	**67.83% ± 0.33**
$EMAP(KPCA_{\sigma=1.5})$	**94.23% ± 0.35**	94.92% ± 1.15	**98.81% ± 0.14**	**88.74% ± 0.64**	92.65% ± 0.40	93.96% ± 0.28	67.29% ± 0.62
$EMAP(KPCA_{\sigma=2.0})$	93.96% ± 0.57	**95.05% ± 1.09**	98.67% ± 0.14	88.25% ± 0.53	92.26% ± 0.61	93.90% ± 0.33	67.76% ± 0.73

RF classifier. These results can be compared to those obtained after applying EMAPs to the original hyperspectral data in [43], or after applying a composite kernel-based approach in [49].

An interesting observation from Tables 2.1 and 2.2 is that the EMAP built on the multispectral/RGB data generally provides good results in terms of classification accuracy (even higher than with the original hyperspectral data in the ROSIS Pavia and AVIRIS Indian Pines scenes). This indicates the importance of including spatial information in the analysis. In turn, as expected the classification accuracies obtained using the multispectral/RGB data alone are always sensibly lower. On the other hand, from Tables 2.1 and 2.2 it can also be seen that, when the KPCA is applied to the multispectral/RGB data, the obtained classification accuracies are not significantly improved. However, the combination of EMAPs and KPCA produces results which are generally better than those obtained by the EMAPs alone (with the exception of the SVM classifier in the AVIRIS Indian Pines data). This indicates that the extraction of spatial information using EMAPs is better suited when the contrast between spectral clusters is high. The combination of EMAPs and KPCA produced better results (particularly for the RF classifier).

The experiments generally confirm the observations in [43] that SVMs may be more sensitive to the Hughes phenomenon than RFs. While both the SVM and RF classifiers provide similar results in terms of accuracies for the ROSIS Pavia University data and for the IKONOS Reykjavik data, Table 2.1 shows that the SVM provides inferior results in the case of AVIRIS Indian Pines with limited training samples (particularly when the –high dimensional– combination of EMAPs built on KPCA, resulting in 740 features, is used). In this case, the performance gradually increases as the size of the training set increases from 5% to 15%. However, this effect was not observed in the results of the RF classifier for the AVIRIS Indian Pines dataset, as shown by Table 2.2.

2.4 Summary and future research lines

In this chapter, a new methodology for spectral-spatial classification of remotely sensed multispectral images with limited spectral resolution has been developed. The proposed method first expands the dimensionality of multispectral data using kernel feature extraction, and then includes spatial information by means of morphological characterization of the expanded set of features. The experimental results indicate that the proposed approach is attractive for advanced classification of data sets with limited spectral resolution. Here, three-channel RGB images have been derived from the available hyperspectral data sets to create multispectral data sets. The classification accuracies obtained on such data sets were superior to those provided by EMAPs (built on the multispectral data), and even to those obtained using the full hyperspectral information with hundreds of spectral channels. This reveals the importance of spatial information and indicates that the combination of KPCA and EMAPs provides a simple yet powerful strategy to perform spectral-spatial classification of data sets with limited spectral resolution (in this chapter, we have considered both multispectral scenes and also RGB scenes derived from real hyperspectral data sets). In future work, we plan to use higher-level strategies to derive spatial features such as object-based image analysis and knowledge-based methods.

Chapter 3

Design and Implementation of Efficient Algorithms for Analyzing Remote Sensing Images with High Spectral Resolution

3.1 GPU Implementation of an automatic target detection and classification algorithm for hyperspectral image analysis

Outline - The detection of (moving or static) targets in remotely sensed hyperspectral images often require real-time responses for swift decisions that depend upon high computing performance of algorithm analysis. The automatic target detection and classification algorithm (ATDCA) has been widely used for this purpose. In this subchapter, several optimizations are developed for accelerating the computational performance of ATDCA. The first one focuses on the use of the Gram-Schmidt orthogonalization method instead of the orthogonal projection process adopted by the classic algorithm. The second one is focused on the development of a new implementation of the algorithm on commodity graphics processing units (GPUs). The proposed GPU implementation properly exploits the GPU architecture at low-level, including shared memory, and provides coalesced accesses to memory that lead to very significant speedup factors, thus taking full advantage of the computational power of GPUs. The GPU implementation is specifically tailored to hyperspectral imagery and the special characteristics of this kind of data, achieving real-time performance of ATDCA for the first time in the literature. The proposed optimizations are evaluated not only in terms of target detection accuracy, but also in terms of computational performance using two different GPU architectures by NVidia^{TM}: Tesla C1060 and GeForce GTX 580, taking advantage of the performance of operations in single precision floating point. Experiments are conducted using hyperspectral data sets collected by three different hyperspectral imaging instruments. These results reveal considerable acceleration factors while retaining the same target detection accuracy for the algorithm.

Part of this subchapter has been published in:

S. Bernabé, S. López, A. Plaza and R. Sarmiento, "GPU Implementation of an Automatic Target Detection and Classification Algorithm for Hyperspectral Image Analysis", *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 2, pp. 221–225, March 2013 [JCR(2012)=1.823].

3.1.1 Overview

Target detection and identification are very important tasks in remotely sensed hyperspectral data exploitation [50]. Over the last few years, several algorithms have been developed for the purpose of target identification, including the automatic target detection and classification (ATDCA) algorithm [51], an unsupervised fully-constrained least squares (UFCLS) algorithm [52], an iterative error analysis (IEA) algorithm [53], or the well-known RX algorithm developed by Reed and Xiaoli for anomaly detection purposes [54]. The ATDCA finds a set of spectrally distinct target pixels vectors using the concept of orthogonal subspace projection (OSP) [55] in the spectral domain. On the other hand, the UFCLS algorithm generates a set of distinct targets using the concept of least square-based error minimization. The IEA uses a similar approach, but with a different initialization condition. The RX algorithm is based on the well-known Mahalanobis distance. Many other target/anomaly detection algorithms use different concepts, such as background modeling and characterization [56]. Quantitative and comparative assessments of target detection algorithms reveal good performance of ATDCA with regard to other approaches in terms of detection accuracy and computational performance [50]. Depending on the complexity and dimensionality of the hyperspectral data the aforementioned algorithms may be computationally very expensive, a fact that limits the possibility of utilizing those algorithms in time-critical applications [57]. Despite the growing interest in parallel hyperspectral imaging research, only a few parallel implementations of automatic target detection algorithms for hyperspectral data have been reported in the open literature [58]. With the recent explosion in the amount and dimensionality of hyperspectral imagery, parallel processing is a requirement in many remote sensing missions.

In this subchapter, the first real-time implementation of the ATDCA algorithm is developed for remotely sensed hyperspectral data exploitation. It is based on two optimizations: i) use of the Gram-Schmidt orthogonalization method instead of the OSP process adopted by the classic algorithm, which has already demonstrated its effectiveness when applied to the vertex component analysis (VCA) endmember extraction algorithm [59], and ii) the development of an efficient implementation of the algorithm on commodity graphics processing units (GPUs), a low-weight hardware platform that offers a tremendous potential to bridge the gap towards real-time analysis of remotely sensed hyperspectral data [21, 60, 61]. The proposed implementation exploits the GPU architecture at low-level, including shared memory, and provides coalesced accesses to memory that lead to very significant speedup factors, thus taking full advantage of the computational power of GPUs. The remainder of this subchapter is organized as follows. Section 3.1.2 describes the original ATDCA and the adopted Gram-Schmidt optimization. Section 3.1.3 describes a new and fully optimized GPU implementation of this algorithm. Section 3.1.4 evaluates the proposed GPU implementation in terms of target detection accuracy and computational performance. Section 3.1.5 concludes the subchapter with some remarks and future research lines.

3.1.2 ATDCA and its Gram-Schmidt optimization

The original ATDCA is based on OSP concepts, and will be referred to hereinafter as ATDCA-OSP. This method is summarized in Algorithm 1, where $\mathbf{U}$ is a matrix of spectral signatures, $\mathbf{U}^T$ is the transpose of this matrix, and $\mathbf{I}$ is the identity matrix.

An optimization of the ATDCA-OSP algorithm consists of using the Gram-Schmidt method (instead of the OSP) for orthogonalization purposes. This version, called ATDCA-GS hereinafter, selects a finite set of linearly independent vectors $\mathbf{A} = \{\mathbf{a}_1, \cdots, \mathbf{a}_k\}$ 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 $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_k\}$ which

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

$$\begin{array}{ll}
\mathbf{b}_1 = \mathbf{a}_1, & \mathbf{e}_1 = \frac{\mathbf{b}_1}{\|\mathbf{b}_1\|} \\
\mathbf{b}_2 = \mathbf{a}_2 - proj_{\,\mathbf{b}_1}(\mathbf{a}_2), & \mathbf{e}_2 = \frac{\mathbf{b}_2}{\|\mathbf{b}_2\|} \\
\mathbf{b}_3 = \mathbf{a}_3 - proj_{\,\mathbf{b}_1}(\mathbf{a}_3) - proj_{\,\mathbf{b}_2}(\mathbf{a}_3), & \mathbf{e}_3 = \frac{\mathbf{b}_3}{\|\mathbf{b}_3\|} \\
\mathbf{b}_4 = \mathbf{a}_4 - proj_{\,\mathbf{b}_1}(\mathbf{a}_4) - proj_{\,\mathbf{b}_2}(\mathbf{a}_4) - proj_{\,\mathbf{b}_3}(\mathbf{a}_4), & \mathbf{e}_4 = \frac{\mathbf{b}_4}{\|\mathbf{b}_4\|} \\
\vdots & \vdots \\
\mathbf{b}_k = \mathbf{a}_k - \sum_{j=1}^{k-1} proj_{\mathbf{b}_j}(\mathbf{a}_k), & \mathbf{e}_k = \frac{\mathbf{b}_k}{\|\mathbf{b}_k\|},
\end{array} \tag{3.1}$$

where the projection operator is defined in Eq. (3.2), in which $< \mathbf{a}, \mathbf{b} >$ denotes the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$.

$$proj_{\,\mathbf{b}}(\mathbf{a}) = \frac{< \mathbf{a}, \mathbf{b} >}{< \mathbf{b}, \mathbf{b} >} \mathbf{b}. \tag{3.2}$$

The sequence $\mathbf{b}_1, \cdots, \mathbf{b}_k$ in Eq. (3.1) represents the set of orthogonal vectors generated by the Gram-Schmidt method, and thus, the normalized vectors $\mathbf{e}_1, \cdots, \mathbf{e}_k$ in Eq. (3.1) form an orthonormal set. As far as $\mathbf{B}$ spans the same k-dimensional subspace of $\mathbf{R}^n$ as $\mathbf{A}$, an additional vector $\mathbf{b}_{k+1}$ computed by following the procedure stated at Eq. (3.1) is also orthogonal to all the vectors included in $\mathbf{A}$ and $\mathbf{B}$. This algebraic assertion constitutes the cornerstone of the ATDCA-GS algorithm, whose pseudocode is represented in Algorithm 2.

Algorithm 1 Pseudocode of ATDCA-OSP

1: **INPUTS:** $\mathbf{Y} \in \mathbf{R}^n$ and t;
   % $\mathbf{Y}$ denotes an n-dimensional hyperspectral image with r pixels and t denotes the number of targets to be detected
2: $\mathbf{U}$ =[$\mathbf{x}_0$ | 0 |, $\cdots$, | 0];
   % $\mathbf{x}_0$ is the pixel vector with maximum length in $\mathbf{Y}$
3: **for** $i = 1$ to $t - 1$ **do**
4: $P^{\perp}_{\mathbf{U}} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$;
   % $P^{\perp}_{\mathbf{U}}$ is a vector orthogonal to the subspace spanned by the columns of $\mathbf{U}$
5: $\mathbf{v} = P^{\perp}_{\mathbf{U}}\mathbf{Y}$;
   % $\mathbf{Y}$ is projected onto the direction indicated by $P^{\perp}_{\mathbf{U}}$
6: $i = \text{argmax}_{\{1,\cdots,r\}}\mathbf{v}[:, i]$;
   % The maximum projection value is found, where r denotes the total number of pixels in the hyperspectral image and the operator ":" denotes "all elements"
7: $\mathbf{x}_i \equiv \mathbf{U}[:, i+1] = \mathbf{Y}[:, i]$;
   % The target matrix is updated
8: **end for**
9: **OUTPUT:** $\mathbf{U} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{t-1}]$;

As it can be observed from Algorithm 2, the computation of the orthogonal projector $P^{\perp}_{\mathbf{U}}$ starts by first initializing it to an arbitrary n-dimensional vector (step 6 of Algorithm 2). Then, the Gram-Schmidt orthogonal set $\mathbf{B}$ is generated from the targets already detected in the image (steps 7–10 of Algorithm 2). The main modification of this method consists of fixing the vector $\mathbf{w}$ to $[\mathbf{1}, \cdots, \mathbf{1}]^{\mathbf{T}}$ rather than generating a random vector at each iteration. As far as the underlying reason for generating a random vector is only to get a nonnull projection, this can also be achieved by fixing $\mathbf{w}$ in the way that it has been previously mentioned, which allows the reduction of the computational cost of the ATDCA-GS algorithm by avoiding the generation of random vectors. Finally, $P^{\perp}_{\mathbf{U}}$ is updated in steps 11–14 of Algorithm 2 by forcing it to be orthogonal to all the vectors in $\mathbf{B}$ and, as a consequence, to all the targets already stored in $\mathbf{U}$. At this point, it is important to emphasize that the n components of the orthogonal projector $P^{\perp}_{\mathbf{U}}$

could be initialized to any other values, since this would not affect its orthogonality with respect to the targets already extracted from the input hyperspectral image $\mathbf{Y}$. Last but not least, we emphasize that the ATDCA-GS is ideally suited for hardware implementation as it avoids the inverse operation which is very difficult to implement in hardware and obtains the same functionality with simpler operations.

Algorithm 2 Pseudocode of ATDCA-GS

INPUTS: $\mathbf{Y} \in \mathbf{R}^n$ and t;
% $\mathbf{Y}$ denotes an n-dimensional hyperspectral image with r pixels and t denotes the number of targets to be detected
$\mathbf{U}$ =[$\mathbf{x}_0$ | 0 |, $\cdots$, | 0];
% $\mathbf{x}_0$ is the pixel vector with maximum length in $\mathbf{Y}$
$\mathbf{B}$ =[0 | 0 |, $\cdots$, | 0 |];
% $\mathbf{B}$ is an auxiliary matrix for storing the orthogonal base generated by the Gram-Schmidt process
for $i = 1$ to $t - 1$ **do**
  $\mathbf{B}[:, i] = \mathbf{U}[:, i]$;
  % the i-th column of $\mathbf{B}$ is initialized with the target computed in the last iteration (here, the operator ":" denotes "all elements")
  $P^{\perp}_{\mathbf{U}} = [\mathbf{1}, \cdots, \mathbf{1}]$;
  for $j = 2$ to i **do**
    $proj_{\mathbf{B}[:,j-1]}(\mathbf{U}[:, i]) = \frac{\mathbf{U}[:,i]^T \mathbf{B}[:,j-1]}{\mathbf{B}[:,j-1]^T \mathbf{B}[:,j-1]} \mathbf{B}[:, j-1]$;
    $\mathbf{B}[:, i] = \mathbf{B}[:, i] - proj_{\mathbf{B}[:,j-1]}(\mathbf{U}[:, i])$;
    % The i-th column of $\mathbf{B}$ is updated
  end for j
  % The computation of $\mathbf{B}$ is finished for the current iteration of the main loop
  for $k = 1$ to i **do**
    $proj_{\mathbf{B}[:,k]}(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{B}[:,k]}{\mathbf{B}[:,k]^T \mathbf{B}[:,k]} \mathbf{B}[:, k]$;
    $P^{\perp}_{\mathbf{U}} = P^{\perp}_{\mathbf{U}} - proj_{\mathbf{B}[:,k]}(\mathbf{w})$;
  end for k
  % The computation of $P^{\perp}_{\mathbf{U}}$ is finished for the current iteration of the main loop
  $\mathbf{v} = P^{\perp}_{\mathbf{U}} \mathbf{Y}$;
  % $\mathbf{Y}$ is projected onto the direction indicated by $P^{\perp}_{\mathbf{U}}$
  $i = \text{argmax}_{\{1,\cdots,r\}} \mathbf{v}[:, i]$;
  % The maximum projection value is found, where r denotes the total number of pixels in the hyperspectral image
  $\mathbf{x}_i \equiv \mathbf{U}[:, i+1] = \mathbf{Y}[:, i]$;
  % The target matrix is updated
end for
OUTPUT: $\mathbf{U} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{t-1}]$;

3.1.3 GPU implementation

In this section an efficient implementation of ATDCA-GS for GPUs is described. It should be noted that a GPU implementation of the original ATDCA-OSP algorithm was presented in [58]. In the context of NVidia compute unified device architecture (CUDA)[1] adopted for the implementation, GPUs can be abstracted in terms of a stream model, under which all data sets are represented as streams (i.e., ordered data sets) [62]. The architecture of a GPU can be seen as a set of multiprocessors (MPs), where each multiprocessor is characterized by a single instruction multiple data (SIMD) architecture, i.e., in each clock cycle each processor executes the same instruction but operating on multiple data streams. Each processor has access to a local shared memory and also to local cache memories in the multiprocessor, while the multiprocessors have access to the global GPU (device) memory. Algorithms

[1] http://www.nvidia.com/object/cuda_home_new.html

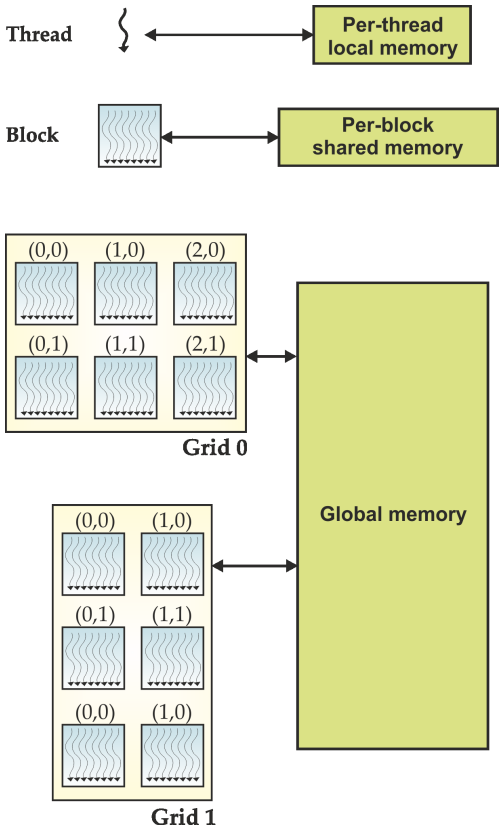


Figure 3.1: GPU parallelism at the thread, block and grid levels.

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 of 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 which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. Fig. 3.1 illustrates how the GPU parallelism can be used at the thread, block and grid levels.

Next, we describe the different steps and architecture-related optimizations carried out in the development of the GPU version of the ATDCA-GS algorithm. The first step is related with the proper arrangement of the hyperspectral data in the local GPU memory. In order to optimize accesses, bearing in mind that the ATDCA-GS algorithm uses the pixel vector as the minimum unit of computation, we store the pixel vectors of the hyperspectral image $\mathbf{Y}$ by columns. The arrangement is intended to access consecutive wavelength values in parallel by the processing kernels (coalesced accesses to memory). This means that the i-th thread of a block will access the i-th wavelength component of a pixel vector of the image. This technique is used to maximize global memory bandwidth and minimize the number of bus transactions. Once the hyperspectral image is mapped onto the GPU memory using the aforementioned strategy, a structure is created in which the number of blocks equals the number of pixel vectors 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 is now used to calculate the brightest pixel $\mathbf{x}_0$ in $\mathbf{Y}$. This kernel computes (in parallel) the dot product between each pixel vector

and its own transposed version, retaining the pixel that results in the maximum projection value.

Once the brightest pixel in $\mathbf{Y}$ has been identified, the pixel is allocated as the first column in matrix $\mathbf{U}$. In this way, we ensure that memory accesses are coalesced. The algorithm now calculates the orthogonal vectors through the Gram-Schmidt method as detailed in Algorithm 2. This operation is performed in the CPU because this method operates on a small data structure and the results can be obtained very quickly. A new kernel is created, in which the number of blocks equals the number of pixel vectors 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. This kernel is now applied to project the orthogonal vector onto each pixel in the image. An important optimization applied at this point involves the effective use of the shared memories, which act as small and very fast cache memories available for the processing elements within the same block. Allocating the data properly in these shared memories is crucial for obtaining a good performance in the GPU. In our case, these memories are used to store the most orthogonal vectors obtained at each iteration of ATDCA-GS (this is because these vectors will be accessed every time that the projection onto each pixel of the image is performed). Hence, it is crucial to store these vectors in these small cache memories in order to perform the projection operations much faster and with fewer memory accesses as compared to the case in which these vectors are stored in the main GPU memory. The maximum of all projected pixels is calculated using a separate reduction kernel which also uses shared memory to store each of the projections and obtains the new target $\mathbf{x}_1$. The algorithm now extends the target matrix as $\mathbf{U} = [\mathbf{x}_0\mathbf{x}_1]$ and repeats the same process until the desired number of targets (specified by the input parameter t) has been detected. The output of the algorithm is a set of targets $\mathbf{U} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{t-1}]$.

3.1.4 Experimental results

3.1.4.1 Hyperspectral image data

Four hyperspectral images are used in the experiments. Two of them were obtained by the NASA Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS)[2] over the Cuprite mining district in Nevada and over the World Trade Center in New York (see Fig. 3.2), just five days after the terrorist attacks of September 11, 2001. On the one hand, the Cuprite data [see Fig. 3.3(a)] was collected in the summer of 1997 and corresponds to a 350×350-pixels subset of the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data, which comprise 188 spectral data channels in the range from 400 to 2500 nm and a total size of around 50 MB. Water absorption and low SNR data channels were removed prior to the analysis. The site is well understood mineralogically, and has several exposed minerals of interest, including alunite, buddingtonite, calcite, kaolinite, and muscovite. Reference ground signatures of the above minerals [see Fig. 3.3(b)], available in the form of a USGS library. The spatial resolution is 20 meters per pixel. On the other hand, the World Trade Center data consists of 614×512 pixels, 224 spectral channels and a total size of (approximately) 140 MB. The spatial resolution is 1.7 meters per pixel. The rightmost part of Fig. 3.2 shows a USGS thermal map depicting the target locations of the thermal hot spots at the WTC area. The map is centered at the region where the towers collapsed, and the temperatures of the targets range from 700F to 1300F. These targets will be used as reference information for validation purposes in the comparison.

Another data set collected by the HYperspectral Digital Imagery Collection Experiment (HYDICE) sensor was used in experiments, which represents a subset of the well-known forest radiance data [50] consisting of 64×64 pixels and 169 spectral channels for a total size of 5.28 MB. The spatial resolution

[2] http://aviris.jpl.nasa.gov

Figure 3.2: False color composition of an AVIRIS hyperspectral image collected by NASA's Jet Propulsion Laboratory over lower Manhattan on Sept. 16, 2001 (left). Location of thermal hot spots in the fires observed in World Trade Center area, available online: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif (right).

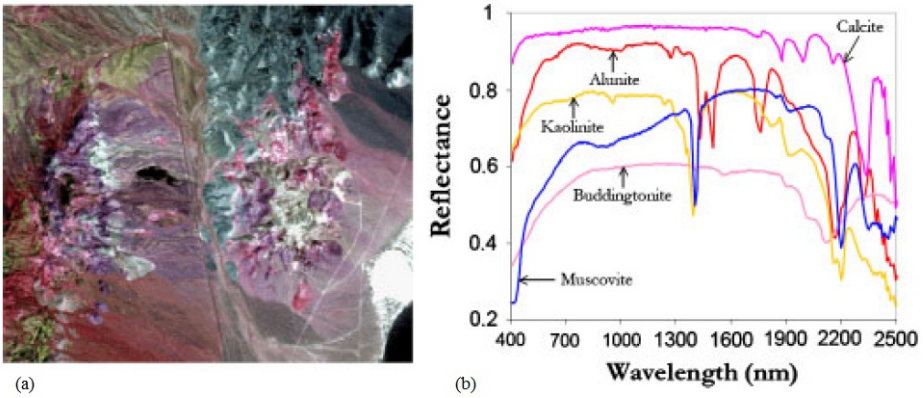


Figure 3.3: (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada and (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

is 3 meters per pixel. Finally, we have also used a well-known hyperspectral data set collected by the Reflective Optics Imaging Spectrographic System (ROSIS) over an urban area in the city of Pavia, Italy, which has been also widely used in the literature [57] and has been described in section 2.3.1.

3.1.4.2 Analysis of target detection accuracy

In this subsection the performance of ATDCA-GS and ATDCA-OSP implementations are compared using the two AVIRIS scenes for which reference data information is available. Table 3.1 shows the spectral angle distance (SAD) [50] values (in degrees) between the most similar target pixels detected by ATDCA-OSP and the pixel vectors at the known target positions, labeled from "A" to "H", in the AVIRIS World Trade Center image. The same results are reported for the ATDCA-GS. In all cases, the number of target pixels to be detected was set to $t = 30$ after calculating the virtual dimensionality

Table 3.1: Spectral angle values (in degrees) between the target pixels extracted by ATDCA-OSP and ATDCA-GS and the known ground targets in the AVIRIS World Trade Center scene.

Version	A	B	C	D	E	F	G	H
ATDCA-OSP	0.00°	14.43°	0.00°	27.38°	20.32°	7.13°	4.15°	31.27°
ATDCA-GS	0.00°	27.16°	0.00°	15.62°	27.81°	3.98°	2.72°	24.26°

Table 3.2: Spectral angle values (in degrees) between the target pixels extracted by ATDCA-OSP and ATDCA-GS and the known ground targets in the AVIRIS Cuprite scene.

Version	Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite	Average
ATDCA-OSP	4.81°	4.16°	9.52°	10.76°	5.29°	6.91°
ATDCA-GS	5.48°	4.08°	5.87°	11.14°	5.68°	6.45°

(VD) of the data [63]. As shown by Table 3.1, the ATDCA-OSP and ATDCA-GS extracted targets which were similar, spectrally, to the known reference data targets. Both versions were able to perfectly detect the targets labeled as "A" and "C", and had more difficulties in detecting other targets. In the case of targets labeled as "D" to "H", the ATDCA-GS improved the target detection results (lower SAD values in Table 3.1) with regards to the ATDCA-OSP.

Table 3.2 shows the SAD values (in degrees) between the most similar target pixels detected by the two considered versions: ATDCA-OSP and ATDCA-GS, and the pixel vectors at the known positions of the target minerals in the AVIRIS Cuprite image. In all cases, the number of target pixels to be detected was set to $t = 19$ after calculating the VD. As shown by Table 3.2, the ATDCA-GS extracted targets which were slightly more similar (on average) to the ground references than those provided by ATDCA-OSP. This indicates that the proposed Gram-Schmidt optimization does not penalize the ATDCA in terms of target detection accuracy.

3.1.4.3 Analysis of parallel performance

Two different GPU platforms have been used in the experiments. The first one is the NVidia Tesla C1060 GPU[3], and the second is the NVidia GeForce GTX 580 GPU[4]. Both GPUs are connected to an Intel core i7 920 CPU at 2.67 GHz with 8 cores, which uses a motherboard Asus P6T7 WS SuperComputer. Before analyzing the parallel performance of the proposed GPU implementation, we emphasize that parallel versions provide exactly the same results as the corresponding serial versions, executed in one of the cores of the i7 920 CPU and implemented using the `gcc` (gnu compiler default) with optimization flag –O3 to exploit data locality and avoid redundant computations. As a result, the only difference between the serial and parallel versions is the time they need to complete their calculations. The reported GPU times are the mean of ten executions in each platform (the measured times were always very similar, with differences –if any– on the order of only a few milliseconds).

Table 3.3 reports the processing times obtained for the GPU implementations of ATDCA-OSP and ATDCA-GS on the two considered GPU architectures, and for the four considered hyperspectral scenes. It should be noted that the GPU implementation of ATDCA-OSP corresponds to an improvement of the one reported in [58] (some kernels were further optimized), while the GPU implementation of ATDCA-

[3]http://www.nvidia.com/object/product_tesla_c1060_us.html
[4]http://www.nvidia.com/object/product-geforce-gtx-580-us.html

GS is the one described in this subchapter. As shown by Table 3.3, the ATDCA-GS achieved significant speedups in both GPU architectures and offered significant improvements with regards to the previously available GPU implementation of ATDCA-OSP. The slightly lower speedups achieved for the HYDICE image (5.28 MB in size) compared to those obtained for the AVIRIS World Trade Center (140 MB in size) indicate that the ATDCA-GS provides more significant acceleration factors as the amount of data to be processed is larger. The processing times achieved by the GPU implementation of ATDCA-GS are strictly in real-time for the AVIRIS data. The cross-track line scan time in AVIRIS, a push-broom instrument, is quite fast (8.3 milliseconds to collect 512 full pixel vectors). This introduces the need to process the AVIRIS World Trade Center scene (614×512 pixels and 224 spectral channels) in less than 5.09 seconds and the AVIRIS Cuprite scene (350×350 pixels and 188 spectral channels) in less than 1.98 seconds in order to achieve real-time performance. As noted in Table 3.3, all the proposed GPU implementations of ATDCA-GS are well below one second in processing time, including the loading times and the data transfer times from CPU to GPU and vice-versa. This represents a significant improvement with regards to previous GPU implementations of ATDCA [21, 58].

For illustrative purposes, Fig. 3.4 shows the percentage of the total GPU execution time consumed by memory transfers and by each CUDA kernel (obtained after profiling the ATDCA-GS implementation) along with the number of times that each kernel was invoked (in the parentheses) for the detection of $t = 30$ targets from the AVIRIS World Trade Center scene in the NVidia GeForce GTX 580 architecture, with and without the GPU architecture optimizations related with the use of shared memories and coalesced accesses described in section 3.1.3. As shown by Fig. 3.4, the GPU implementation without optimizations uses approximately 55% of the execution time for running the kernels and 45% of the time for memory transfers. On the other hand, the version with optimizations significantly decreases the percentage of time devoted to memory transfers and increases the percentage of time used for running the kernels. This indicates that the proposed implementation does not represent a straightforward parallelization effort but, instead, a careful effort to adapt the GPU architecture to the specific issues involved in hyperspectral data processing.

3.1.5 Summary and future research lines

In this subchapter, the first real-time implementation of an automatic target detection and classification algorithm (ATDCA) for hyperspectral data has been developed, implemented with Gram-Schmidt orthogonalization, on GPU architectures. The proposed implementation has been specifically tailored to specific aspects involved in hyperspectral data processing, and makes advanced use of the GPU architecture including considerations such as the arrangement of the data in the GPU local and shared memories in order to ensure coalesced memory accesses and low memory transfer times. Although the results obtained with a variety of hyperspectral images are very encouraging, GPUs are still far from being exploited in real missions due to power consumption and radiation tolerance issues to be addressed in future developments. Future work will also explore how to merge different kernels used to reduce kernel invoking time.

Table 3.3: Processing times (seconds) and speedups achieved by ATDCA-OSP and ATDCA-GS with optimizations in two different GPUs.

	AVIRIS World Trade Center		**AVIRIS Cuprite**		**ROSIS Pavia University**		**HYDICE Forest Radiance**	
	ATDCA-OSP	**ATDCA-GS**	**ATDCA-OSP**	**ATDCA-GS**	**ATDCA-OSP**	**ATDCA-GS**	**ATDCA-OSP**	**ATDCA-GS**
Serial time	512.1120	7.3230	87.9820	1.5950	20.2672	0.6460	2.2341	0.0331
Time Tesla C1060 GPU	51.4626	0.1663	9.0032	0.0560	2.2840	0.0407	0.4523	0.0045
Time GeForce GTX 580 GPU	10.5747	0.1554	1.9947	0.0456	0.5834	0.0308	0.1837	0.0035
Speedup Tesla C1060 GPU	9.95	44.03	9.77	28.48	8.87	15.87	4.94	7.36
Speedup GeForce GTX 580 GPU	48.43	47.12	44.11	35.01	34.74	20.97	12.16	9.46

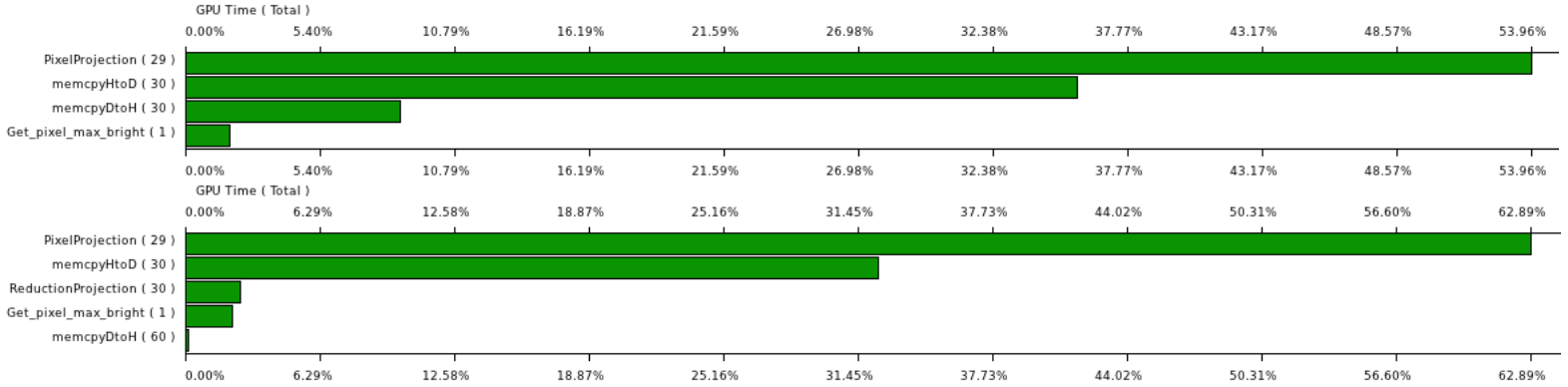


Figure 3.4: Summary plot describing the percentage of the total GPU time consumed by memory transfer operations and by the different kernels used by ATDCA-GS in the NVidia GeForce GTX 580 GPU (AVIRIS World Trade Center scene) without (top) and with architecture optimizations (bottom)

3.2 FPGA design of an automatic target detection and classification algorithm for hyperspectral image analysis

Outline - Onboard processing of remotely sensed hyperspectral data is a highly desirable goal in many applications. For this purpose, compact reconfigurable hardware modules such as field programmable gate arrays (FPGAs) are widely used. In this subchapter, several designs of an automatic target detection and classification algorithm (ATDCA) are developed for hyperspectral images. The design methodology starts from a high-level description in Matlab (or alternative C/C++) and obtains a register transfer level (RTL) description that can be ported to FPGAs. In order to validate the designs, a quantitative and comparative study is developed using two different FPGA architectures: Xilinx Virtex-5 and Altera Stratix-III. Experimental results have been obtained in the context of a real application focused on the detection of mineral components over the Cuprite mining district (Nevada), using hyperspectral data collected by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS). The experimental results indicate that the proposed designs can achieve peak frequency designs above 200 MHz in the considered FPGAs, in addition to satisfactory results in terms of target detection accuracy and parallel performance. This represents a step forward towards the design of real-time onboard implementations of hyperspectral image analysis algorithms.

3.2.1 Overview

One of the most important challenges in hyperspectral image analysis is computational complexity, which results from the need to process enormous data volumes [18]. With recent advances in reconfigurable computing, especially using field programmable gate arrays (FPGAs) [64–66], hyperspectral imaging algorithms can now be accelerated for onboard exploitation using high-performance FPGAs. One of the most critical issues when designing hardware-based (in general) and FPGA-based (in particular) implementations of hyperspectral imaging algorithms is the complexity of the design, which introduces a learning curve for algorithm developers [27, 67, 68]. Although the hardware design is usually carried out at the register-transfer level (RTL), the design can be achieved through a flow starting from a high-level Matlab or C/C++ code, thus allowing for the reutilization of available high-level versions of hyperspectral imaging algorithms. This simplifies the design flow and the adaptation of available algorithms, which is a very important aspect as we anticipate that the success of hyperspectral imaging algorithms will be soon given by their adaptivity to high performance computing platforms able to operate onboard the sensor. For instance, the Embedded Matlab module[5] allows generating efficient code in C and C++ from a high-level Matlab description. Then, tools such as Mentor Graphics' Catapult C[6] can bridge the gap between the high-level implementation and a lower-level synthesis for FPGA hardware design. These tools offer the potential to significantly help the programmer in the development of low-level hardware code with a more productive level of abstraction.

Part of this subchapter has been published in:

S. Bernabé, S. López, A. Plaza, R. Sarmiento and P. G. Rodríguez, "FPGA Design of an Automatic Target Generation Process for Hyperspectral Image Analysis", *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, Tainan (Taiwan), December 2011.

[5] http://www.mathworks.es/help/toolbox/eml/index.html
[6] http://www.mentor.com/esl/catapult/overview

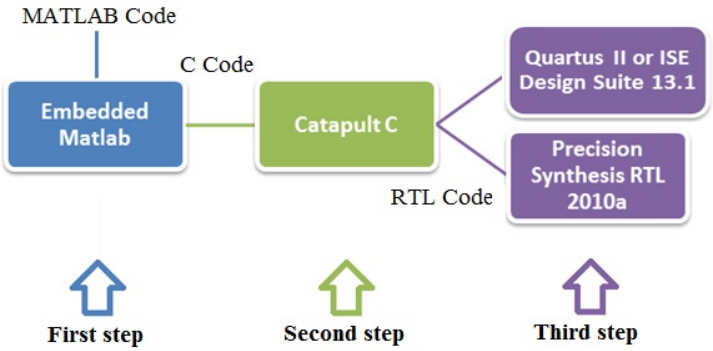


Figure 3.5: General methodology for high-level design.

In this subchapter, the utility of these tools is explored to develop several FPGA-based synthesis version of a popular technique for automatic target detection in hyperspectral images. The technique, called automatic target detection and classification algorithm (ATDCA) [51], automatically detects a set of spectrally distinct *targets* in a hyperspectral image, with the ultimate goal of characterizing the different spectral constituents that compose the scene. It is widely used in applications involving the detection and monitoring of fires, oil spills, and other types of chemical and biological agents [58]. This algorithm and its optimizations have already been described in section 3.1.2. The remainder of the subchapter is organized as follows. Section 3.2.2 describes the flow of high-level design from a Matlab code. Section 3.2.3 describes several hardware versions of the proposed implementation of ATDCA. Section 3.2.4 provides a experimental assessment of the developed versions on two different FPGA architectures: Xilinx Virtex-5 and Altera Stratix-III, using a well-known hyperspectral data set collected by AVIRIS in order to evaluate the target detection accuracy of the proposed versions. Finally, section 3.2.5 concludes with some remarks and hints at plausible future research lines.

3.2.2 Implementation flow: from high-level to low-level design

This section presents the implementation flow which starts from a high-level sequential version developed in Matlab or C/C++ code to the low-level implementation. The process is summarized in Fig. 3.5 and consists of the following steps:

1. The Embedded MATLAB module is used to generate efficient code in C/C++, so that we can create prototypes and develop embedded systems to accelerate fixed-point algorithms.

2. Catapult C is used for high-level synthesis of FPGA hardware design. From a given code in C, Catapult C produces an output for generating RTL code.

3. A design tool is used for FPGAs (such as Xilinx ISE Design suite 13.1[7] or Altera Quartus II[8]) to compile the design and obtain the frequency that can be achieved with the hardware implementation. The software package Precision RTL Synthesis is used to optimize the design and introduce advanced optimizations, obtaining a more accurate RTL description based on the FPGA card used.

[7] http://www.xilinx.com/support/documentation/dt_ise13-1.htm
[8] http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html

The methodology described in Fig. 3.5 represents the design flow from a very general perspective. The flow may be more specific if the original source code is available in Matlab. For illustrative purposes, Fig. 3.6 shows a more specific description of the flow in the case that the original code is developed in Matlab.

3.2.3 FPGA designs

Three FPGA designs have been developed for ATDCA, two of them following the flow of high-level design from Matlab and another version from C. Next, the different versions implemented in Matlab and C code are explained, describing first those using operations in floating point and later those using operations in fixed point.

3.2.3.1 Implementations using operations in floating point

For the floating point implementations, a Matlab version and another version in C have been implemented. As mentioned above, the main innovation of these versions is the replacement of the inverse function by the Gram-Schmidt orthogonalization method. The goal of these versions is to reduce the computational cost without affecting the output of the algorithm. For illustrative purposes, Fig. 3.7 provides a general scheme of the modified ATDCA. Fig. 3.8 describes the Matlab implementation, where lines 20-29 describe the process for Gram-Schmidt orthogonalization that constitutes the main modification performed in the high-level description of the algorithm. The remaining parts of the algorithm do not change and can be efficiently implemented in C code.

3.2.3.2 Implementations using operations in fixed point

In order to develop this version the Matlab toolbox for fixed-point operations was used. The size of the fixed-point variables must be defined in advance. Also we need to define in advance the sum and product formats. For illustrative purposes, Fig. 3.9 shows the initialization of a variable-point fixed generic Matlab and the initialization of the same variable with support for Embedded Matlab. It should be noted that the use of this toolbox allows for the design of a more optimal hardware configuration, as will be shown in experiments in the following section.

3.2.4 Experimental results

3.2.4.1 FPGA hardware

The selected FPGA cards are manufactured by two of the most powerful industries in this field: Xilinx and Altera. The family of Virtex-5[9] FPGAs are the first to own a 65 nm technology. These devices, manufactured in 1.0v, triple-process technology oxide, providing up to 330,000 logic cells, 1200 I/O pins, 48 low-power transistors, PowerPC 440 processor, and Ethernet MAC blocks PCIe endpoint, depending on the selected model. The 5VLX155FF1760 Xilinx Virtex-5 has been used in experiments. On the other hand, the family of Stratix III FPGAs arise from the evolution of Stratix II family, incorporating a 65 nm technology implementation capacity DDR3 533 MHz and offer yields up to 1.6 Gbps LVDS. This family of FPGAs combines high performance with the lowest power possible. In the experiments, the EP3SL200F1152C Stratix-III FPGA has been used.

[9] http://www.xilinx.com/support/documentation/

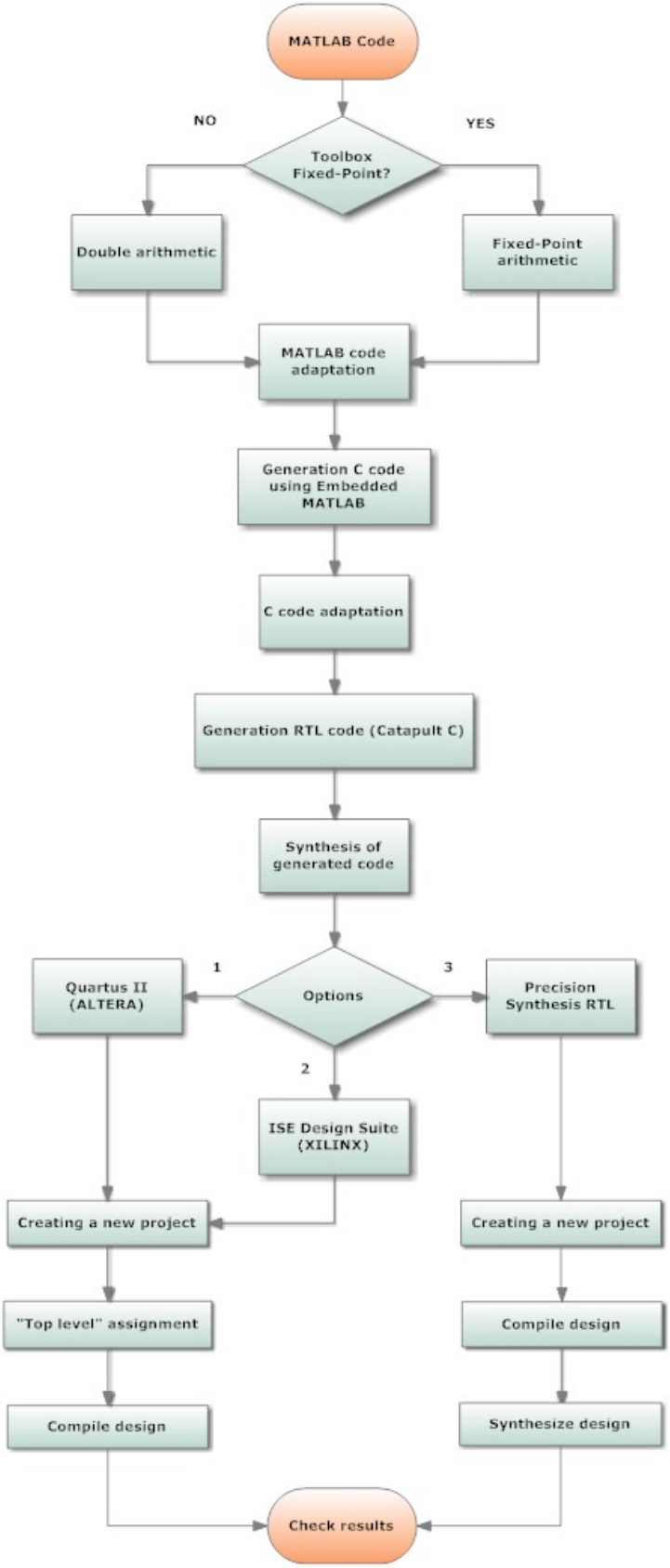


Figure 3.6: Specific methodology for high-level design using Matlab code as input.

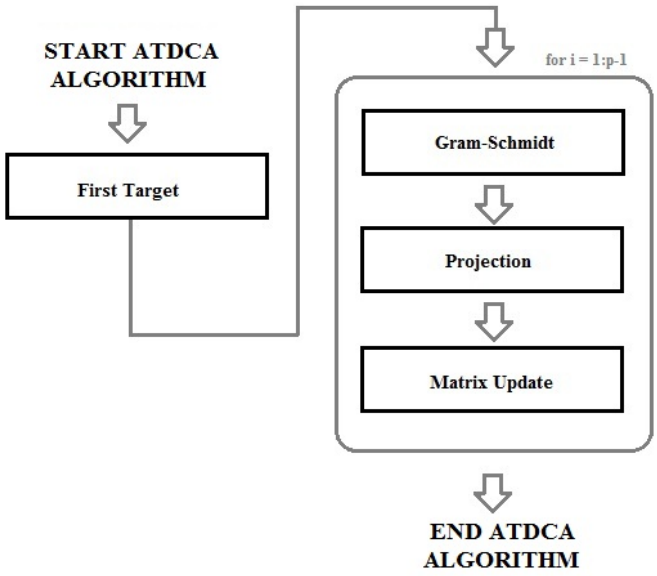


Figure 3.7: General scheme of the modified ATDCA.

3.2.4.2 Implementation results

The hardware implementation results are determined from the synthesis of Verilog code[10] obtained for the considered Virtex-5 and Stratix-III FPGAs. Both codes makes use of RTL Synthesis Precision Tool. Specifically, a comparison between the three modified versions of the ATDCA has been conducted. The first implementation is based on the use of floating point operations, starting from C code. The second implementation is also based on floating point operations but starting from a high-level design in Matlab. Finally, the third implementation uses fixed-point. The results in terms of hardware utilization on each of the FPGAs used will be presented, describing the number of registers, look-up tables (LUTs) and multiplication blocks (DSPs). On the other hand, the value of the maximum frequency that can be achieved is reported by the proposed implementations.

For this purpose, a small portion of the AVIRIS Cuprite image has been used. The considered image portion comprises 36×36 pixels and 188 spectral channels. The number of targets to be detected is $t = 5$. The results in terms of hardware utilization for this experiment are summarized in Tables 3.4, 3.5 and 3.6. These results suggest that the use of hardware resources is higher in the third implementation, mainly due to the fact that the fixed-point toolbox of Matlab requires more logic. The best performance in our experiments is obtained by the first implementation, which achieves a maximum frequency of execution of nearly 200 MHz, which is faster than that reported for the other methods. In addition, it is remarkable the good performance achieved by the Stratix III FPGA from Altera, which achieves the best overall performance results across the three tested implementations. Finally, it is worth noting that the first and third implementations are good candidates for conducting a high-level design from Matlab or from C/C++. Both implementations could be satisfactorily ported to Virtex-5 and Stratix III FPGAs, which is mainly due to the optimizations introduced by tools such as Catapult C or Precision RTL Synthesis, obtaining results with high frequencies and low latencies. These results indicate that the development of hardware-based implementations of hyperspectral imaging algorithms starting from high-level implementations (Matlab or C/C++) is a feasible goal that can be achieved through an adequate

[10] http://www.verilog.com

```
function [U,P] = ATDCA(HIM,p)
% Automatic Target Detection and Classification Algorithm.
% ---------------------------------------------------------------------------
% Input:   HIM : hyperspectral image cube [nrows x ncols x nchannels]
%          p   : desired number of endmembers to be extracted
%
% Output:  U   : set of extracted targets [nchannels x p]
%          P   : spatial coordinates of the extracted targets (positions
%                rows x cols)
%

disp(' === Start ATDCA run ===')

% The part of code that goes here does not vary substantially

% ATDCA algorithm
for i = 1:p-1
    UC = U(:,1:i);
    % Calculate GRAM-SCHMIDT method
    w=ones(nb,1);
    if(i==1)
        f(:,i)=[ones(nb-1,1);-(sum(UC(1:nb-1,1))/UC(nb,1))];
        u(:,1)=UC(:,1);
        c2(i)=(w'*u(:,1))/(u(:,1)'*u(:,1));
    else
        c1 = (u(:,1:i-1)'*UC(:,i))'./(sum(u(:,1:i-1).*u(:,1:i-1)));
        u(:,i) = UC(:,i) - sum(c1(ones(1,nb),:).*u(:,1:i-1),2);
        c2(i) = (w'*u(:,i))/(u(:,i)'*u(:,i));
        f(:,i) = w -  sum(c2(ones(1,nb),:).*u(:,1:i),2);
    end
    % The part of code that goes here does not vary substantially
end

disp(' === Eng ATDCA ===');
```

Figure 3.8: Modification of the ATDCA in Matlab code.

```
Pi_ = fi(3.1416);
Pi_ = fi(3.1416,1,64,32'SumMode','KeepLSB','ProductMode','KeepLSB', …
'ProductWordLength',128,'ProductFractionLength',64,'SumWordLength', …
128,'SumFractionLength',64);
The second initialization parameters have the following meanings:
•   3.1416: Is the numerical value that initializes the Fixed-Point.
•   1: Indicates that the variable is unsigned.
•   64: Indicates the total size of the variable bits.
•   32: Indicates the size corresponding to the decimal part.
•   'SumMode','KeepLSB': Mode sum equal to KeepLSB.
•   'ProductMode','KeepLSB': Mode multiplication equal to KeepLSB.
•   'ProductWordLength',128: Product size equal to 128.
•   'ProductFractionLength',64: Size decimal part of the product equal to 64.
•   'SumWordLength', 128: Size sum equal to 128.
•   'SumFractionLength',64: Size of the decimal part of the sum equal to 64.
```

Figure 3.9: Initialization of a variable-point fixed.

flow of design. Further experiments with additional hyperspectral scenes and hyperspectral imaging algorithms are highly desired in order to extrapolate these observations to other techniques and analysis

Table 3.4: Resources used in the Virtex-5 for the synthesis of the modified ATDCA.

Resources	Implementation 1		Implementation 2		Implementation 3	
	Units	Percentage	Units	Percentage	Units	Percentage
LUTs (out of 97280)	16925	17.40%	19938	20.50%	31138	32.01%
CLB Slices (out of 24320)	4232	17.40%	4985	20.50%	7785	32.01%
Latches (out of 97280)	6583	6.77%	12258	12.60%	20391	20.96%
DSP48Es (out of 128)	12	9.38%	9	7.03%	7	5.47%

Table 3.5: Resources used in the Stratix-III for the synthesis of the modified ATDCA.

Resources	Implementation 1		Implementation 2		Implementation 3	
	Units	Percentage	Units	Percentage	Units	Percentage
LUTs (out of 159120)	16338	10.27%	21712	13.65%	30773	19.34%
Registers (out of 159120)	6224	3.91%	13160	8.27%	21077	13.25%
# DSP blocks	12	2.08%	7	1.22%	10	1.74%

Table 3.6: Restrictions on different modifications of ATDCA.

Restriction	Implementation 1		Implementation 2		Implementation 3	
	Virtex-5	Stratix-III	Virtex-5	Stratix-III	Virtex-5	Stratix-III
Max.Freq.(MHz)	192.123	195.503	148.192	213.493	185.736	200.924
Min.Period(ns)	5.205	5.115	6.748	4.684	5.384	4.977
Latency(cycles)	517	342	4916945	310855	875	626
Latency(time(ms))	0.00269	0.00175	33.17956	14.56186	0.00471	0.00312

scenarios.

3.2.5 Summary and future research lines

This subchapter described several FPGA versions of an automatic target detection and classification algorithm (ATDCA) for hyperspectral image analysis. In our implementations, the impact of including the Gram-Schmidt orthogonalization method has been investigated for calculating the orthogonal projections calculated by this algorithm instead of using an orthogonal subspace projector that includes a very expensive matrix inverse operation. For this purpose, several versions were obtained through an RTL design flow from high-level Matlab or C code. The use of this flow has allowed us to develop efficient hardware implementations from high-level algorithmic descriptions, which is an important contribution in the hyperspectral imaging community as most of the available algorithms are designed using a high-level perspective and not all algorithms map well into hardware, hence our proposed approach allows us to preliminarily test the suitability of a certain algorithm for hardware implementation. In our experiments frequencies above 200 MHz have been reported without compromising target detection accuracy, which represents an innovative contribution in this field. In future work, we will develop real hardware implementations (instead of synthesis results) and experiment with additional algorithms and FPGA platforms to test the validity of our proposed design flow strategies in different analysis scenarios.

3.3 Cloud implementation of a full hyperspectral unmixing chain within the NASA web coverage processing service for EO-1

Outline - The launch of the NASA Earth Observing 1 (EO-1) platform in November 2000 marked the establishment of spaceborne hyperspectral technology for land imaging. The Hyperion sensor onboard EO-1 operates in the 0.4 to 2.5 micrometer spectral range, with 10 nanometer spectral resolution and 30-meter spatial resolution. Spectral unmixing has been one of the most successful approaches to analyze Hyperion data since its launch. It estimates the abundance of spectrally pure constituents (endmembers) in each observation collected by the sensor. Due to the high spectral dimensionality of Hyperion data, unmixing is a very time-consuming operation. In this subchapter, a cloud implementation of a full hyperspectral unmixing chain made up of the following steps is developed: 1) dimensionality reduction; 2) automatic endmember identification; and 3) fully constrained abundance estimation. The unmixing chain will be available online within the Web Coverage Processing Service (WCPS), an image processing framework that can run on the cloud, as part of the NASA SensorWeb suite of web services. The proposed implementation has been demonstrated using EO-1 Hyperion imagery. The experimental results with a hyperspectral scene collected over the Okavango Basin in Botswana suggest the (present and future) potential of spectral unmixing for improved exploitation of spaceborne hyperspectral data. The integration of the unmixing chain in the WCPS framework as part of the NASA SensorWeb suite of web services is just the start of an international collaboration in which many more processing algorithms will be made available to the community through this service. This subchapter is not so much focused on the theory and results of unmixing (widely demonstrated in other contributions) but about the process and added value of the proposed contribution for ground processing on the cloud and onboard migration of those algorithms to support the generation of low-latency products for new airborne/spaceborne missions.

3.3.1 Overview

The Web Coverage Processing Service (WCPS) is an image processing framework that can run on the cloud [69]. It will allow dynamic upload of processing algorithms and data collected from airborne and/or spaceborne platforms. It is the result of a three-year effort by the Earth Science Technology Office[11], Advanced Information Systems Technology (AIST)[12], with the goal of providing imagery products of increased value to the end-user at a lower cost. The core-scripting engine of the WCPS is based on LUA[13], a powerful and embeddable scripting language. The size of the raw data from remotely sensed hyperspectral sensors such as NASA's Earth Observing One (EO-1) [70], the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [71], or the Enhanced MODIS Airborne Simulator (eMAS) [72] is rapidly increasing to data sets reaching almost one Terabyte per scene. This is a major distribution and

Part of this subchapter has been published in:

P. Cappelaere, S. Sánchez, S. Bernabé, A. Scuri, D. Mandl and A. Plaza, "Cloud Implementation of a Full Hyperspectral Unmixing Chain within the NASA Web Coverage Processing Service for E0-1", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 408–418, April 2013 [JCR(2012)=2.874].

[11] http://esto.nasa.gov
[12] http://esto.nasa.gov/info_technologies_aist.html
[13] http://www.lua.org

storage problem for low-bandwidth users as well as storage for users interested in time-series spanning many years. On the cloud, the WCPS can leverage scalable computing capabilities of virtual machines, quickly process large amounts of data where it is stored and deliver finished product to the end-user at an extremely low cost.

In other words, WCPS allows scientists to develop their own custom algorithms and execute them directly against existing data sets on the cloud, a feature that could have a dramatic impact to the new decadal missions for Earth observation such as the hyperspectral infrared imager (HyspIRI) [73]. Similar efforts such as ESA's BEAM project[14] exist, but the WCPS discriminator is in its future capability of being embedded as part of an airborne or spaceborne mission. This opens up possibilities for the development of products on the cloud and their dynamic upload to the flight target such as eMAS on an ER-2, AVIRIS/MASTER on a GlobalHawk, or HyspIRI. The WCPS is the latest Open Geospatial Consortium (OGC) Web Service of the NASA SensorWeb. A standard RESTful and secure API is available for automation. That API is used to integrate the WCPS with the internal workflow engine and can also be used for integration with customer-specific services. Automated processing capabilities are key to providing advanced products in real-time directly to end-users around the world. From a satellite perspective, downlinking the raw data is time-consuming. In a life-threatening situation, a simple flood extent map of 3 Megabytes may be computed onboard and downlinked right to the end-user as an ad-hoc low-latency product with high social benefit [70]. These ad-hoc algorithms would have been tested on the cloud and uplinked on-demand dynamically to the satellite.

Following the aforementioned ideas, this subchapter describes the unmixing chain algorithm that will be available online with the WCPS as part of the NASA SensorWeb suite of web services. Despite the fact that the satellite hyperspectral instruments provide hundreds of narrow spectral channels, the spatial resolution of these instruments is still in the range of 30 to 60 meters per pixel. As a result, most of the pixels collected by EO-1 and the new generation imaging spectrometers such as HyspIRI will be likely mixed in nature [17]. Such an unmixing chain algorithm is therefore critical for proper identification of Earth components. This subchapter will present an approach to execute this computer-intensive algorithm on the cloud running on multiple virtual machines.

Spectral unmixing aims at estimating the abundance of pure spectral components (called endmembers) in each mixed pixel [19,74]. During the past years, many algorithms and models have been developed for endmember identification and abundance estimation in remotely sensed hyperspectral images [75, 76], thus making spectral unmixing a hot topic in the hyperspectral imaging literature. However, the extremely high dimensionality and complexity of hyperspectral images makes the unmixing process a very time-consuming one [18]. This is particularly the case for instruments such as EO-1 Hyperion, which provide a significant coverage of the Earth with multi-temporal capabilities. In these scenarios, high performance computing, now available on the cloud, becomes highly desirable in order to accelerate hyperspectral-related calculations [77]. This is because the cloud can provide a highly dynamic and adaptive environment for distributed execution of processing algorithms while, at the same time, obtaining most of the advantages of cloud implementations with flexibility and high availability. These aspects are of crucial importance for remotely sensed hyperspectral data exploitation.

To illustrate these benefits, a virtual machine implementation of a full hyperspectral unmixing chain is presented within the NASA WCPS for EO-1 on a commercial cloud at Joyent. The WCPS has full-access to the data to be processed as well as the spectral unmixing algorithm (among many others). This enables the dynamic processing of the data on the cloud without having to download it. Future systems

[14] http://www.brockmann-consult.de/cms/web/beam

will automatically store the newly collected data sets into the cloud as soon as they are acquired, so that they can be made widely available to the scientific community together with the processing algorithms with no need to download huge data sets. This is a truly cloud-oriented environment, with the advantage of increased availability of the data and flexibility in the processing. More specifically, the considered unmixing chain consists of the following processing modules:

1. *Dimensionality reduction.* For this purpose, principal component analysis (PCA) [76] is used to reduce the input hyperspectral data set to a proper subspace.

2. *Endmember extraction.* This is accomplished by using the N-FINDR approach [78], one of the most widely used and successfully applied methods for automatically determining endmembers in hyperspectral image data without using a priori information. The algorithm attempts to automatically find the simplex of maximum volume that can be inscribed within the hyperspectral data set.

3. *Abundance estimation.* Once a set of endmembers has been automatically extracted from the input hyperspectral data using N-FINDR, their abundance in each pixel of the scene is estimated by assuming that the fractional abundances of endmembers in a given pixel must add up to one and cannot be negative [52].

The proposed implementation has been demonstrated using the NASA Earth Observing One (EO-1)[15] Hyperion imager, which operates in the 0.4 to 2.5 micrometer spectral range, with 10 nanometer spectral resolution and 30-meter spatial resolution. The experimental results suggest the (present and future) potential of spectral unmixing for improved exploitation of spaceborne hyperspectral data.

The remainder of the subchapter is structured as follows. Section 3.3.2 describes the NASA SensorWeb suite of web services. Section 3.3.3 describes the WCPS framework and some of its internal aspects. Section 3.3.4 describes in detail the full hyperspectral unmixing chain considered in this subchapter. Section 3.3.5 provides WPCS scripting examples including those related with the aforementioned unmixing chain. Section 3.3.6 describes the experimental results obtained for a hyperspectral scene collected by EO-1's Hyperion over the Okavango Basin in Botswana. Section 3.3.7 concludes with some remarks and hints at plausible future research.

3.3.2 The NASA SensorWeb initiative

The NASA SensorWeb was born to reduce the cost of EO-1 operations by using automation. This effort was a significant success for the mission. The second phase of the NASA SensorWeb focused on non-scientific end-users and social benefits that could be derived from data acquired by many NASA or commercial assets, including spaceborne instruments such as EO-1, AVIRIS or the Moderate Resolution Imaging Spectrometer (MODIS) [1].

The capture of expert knowledge into processing algorithms is essential. As science is migrating towards application science, the support and services of end-users are becoming an important portion of the NASA SensorWeb value chain. This value chain concept is illustrated in Fig. 3.10. Visualizing this concept is important as it demonstrates the science value providing eventual social benefits to end users as a result of SensorWeb enablement. The NASA SensorWeb value chain is itself part of a much larger system (see Fig. 3.11). It now encompasses suppliers such as Canadian Space Agency, the European Space Agency (ESA), commercial data providers, value-added distributers such as CATHALAC in

[15] http://eo1.gsfc.nasa.gov

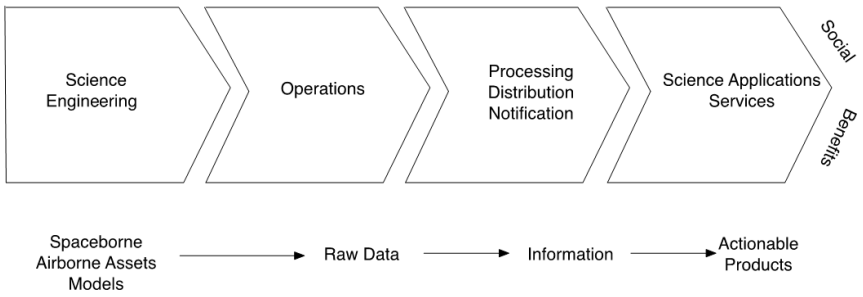


Figure 3.10: NASA SensorWeb value chain (borrowed from Michael Porter)

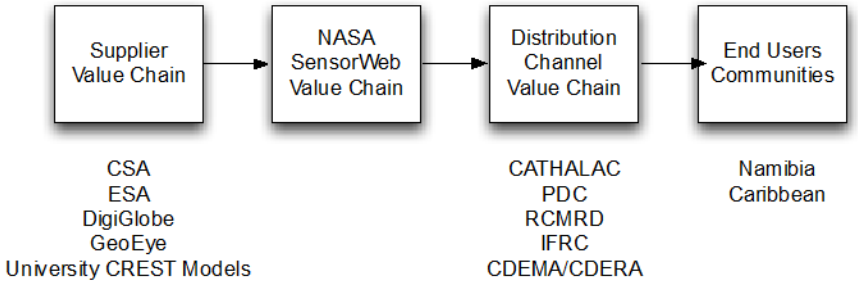


Figure 3.11: An illustration of NASA SensorWeb larger value system

Panama, the Regional Center of Mapping and Resource Development (RCMRD) in Nairobi, the Pacific Disaster Center (PDC) in Maui, the Caribbean Disaster Emergency Management Agency (CDEMA) and Response Agency (CDERA), the International Federation of the Red Cross and Red Crescent, and eventually reaches users all the way to Namibia or in the Caribbean. The diagram (Fig. 3.11) puts in perspective the current scope of the program and its importance as NASA's participation to the international Group on Earth Observation.

3.3.3 The web coverage processing service (WCPS) framework

Data Processing on the cloud is the main focus of the WCPS. The core components of the WCPS are comprised of:

1. LUA, an embeddable scripting Engine[16] developed by a team at the Pontifical Catholic University of Rio de Janeiro (PUC-RIO), used for online games and adopted as a new template engine for Wikipedia.
2. The IM Imaging Library, also developed and managed by PUC-RIO.
3. A C++ Interface Wrapper developed by Vightel Corporation[17], MD, USA.
4. The Flight Interface to the NASA Core Flight Executive (cFE)[18].
5. Node.js, a JavaScript Platform built on the Google chrome engine and Express, a Node.js web application framework.

[16]http://www.lua.org/
[17]http://www.manta.com/c/mm3zwwc/vightel-corporation
[18]http://code.nasa.gov/project/core-flight-executive-cfe

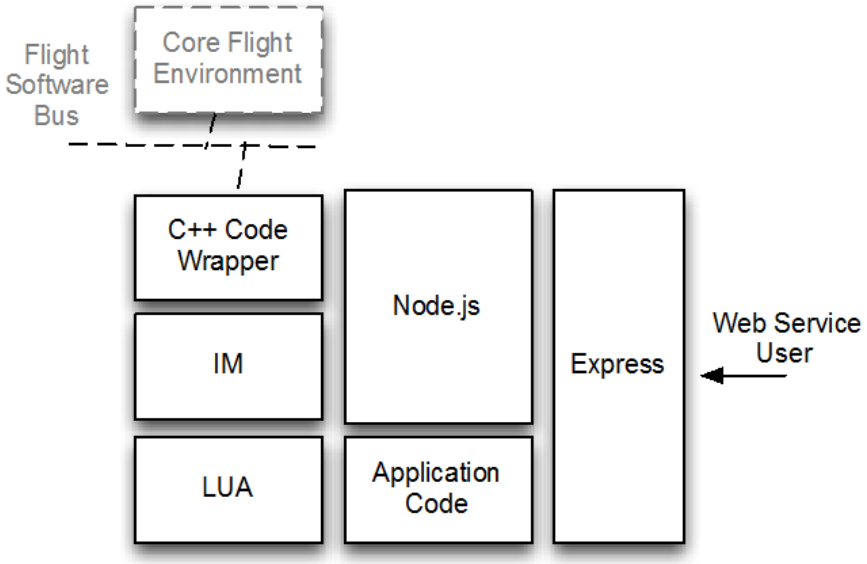


Figure 3.12: The Web Coverage Processing System (WPCS) Flight/Ground Component Architecture

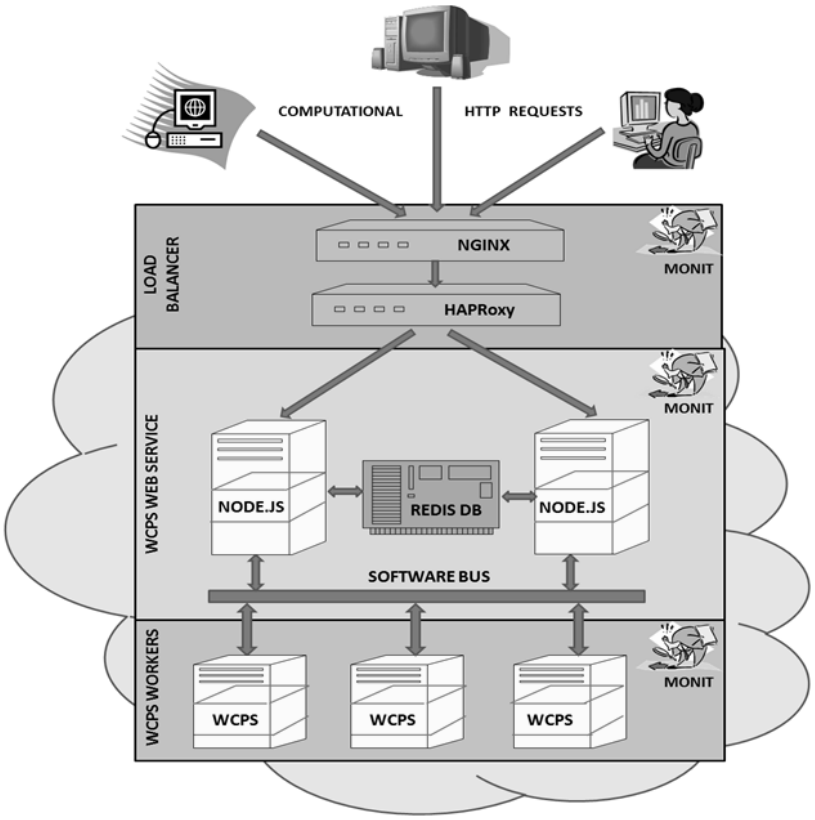


Figure 3.13: Cloud implementation of the Web Coverage Processing System (WPCS)

The WCPS (Fig. 3.12) is also used as a prototype for RESTful/secure standards developed by NASA in coordination with the Open Geospatial Consortium (OGC). It seamlessly integrates with the

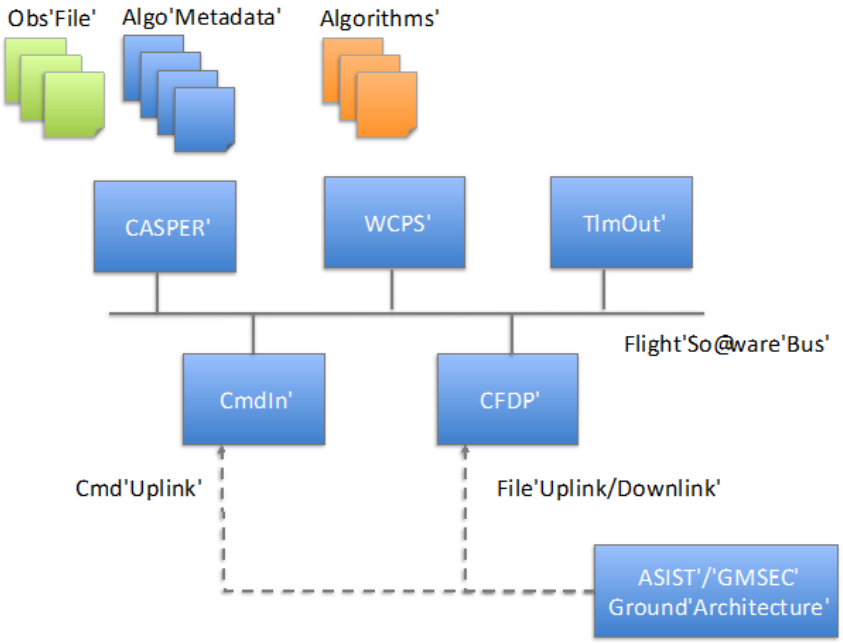


Figure 3.14: Current EO-1 flight architecture

EO-1 Sensor Planning Service (SPS), the Campaign Manager or Workflow Chaining Service (WfCS), the Notification Service and GeoTorrent distribution services. Users from all over the world are required to pass a two-factor authentication using OpenID/OAuth hybrid protocol in order to start any image-processing request. This is provided by a validation and protection service from Symantec (previously Verisign).

The cloud implementation of WPCS is described in Fig. 3.13. To leverage the availability of virtual machines that can be instantiated on the fly, the WCPS is broken down into workers waiting on a software bus for imaging tasks generated by end-users. The front-end web service is clustered behind a load balancer (called HAProxy) and high performance HTTP Server and reverse proxy (called NGINX). A single REDIS key-value store is used to manage user profiles and other miscellaneous resources. MONIT[19] is used to manage and monitor the various processes and recover from errors or crashes.

In parallel, NASA's Goddard Space Flight Center (GSFC) in Maryland has several on-going research efforts to ensure that the software will actually performed well in a flight environment. Specifically, we are using a commercial TileraPro64 processor board with 64 cores as a stand-in for the DOD-developed rad-hard MAESTRO that is more likely to be used for space missions. The NASA cFE has been ported to that environment. The WCPS is simply another application sitting on the flight software bus in a very similar configuration to the ground configuration. The current testbed also includes the NASA Jet Propulsion Laboratory's CASPER[20] application for automated planning and scheduling. This application is actually part of the current EO-1 flight software architecture, described in Fig. 3.14.

[19] http://mmonit.com
[20] http://casper.jpl.nasa.gov

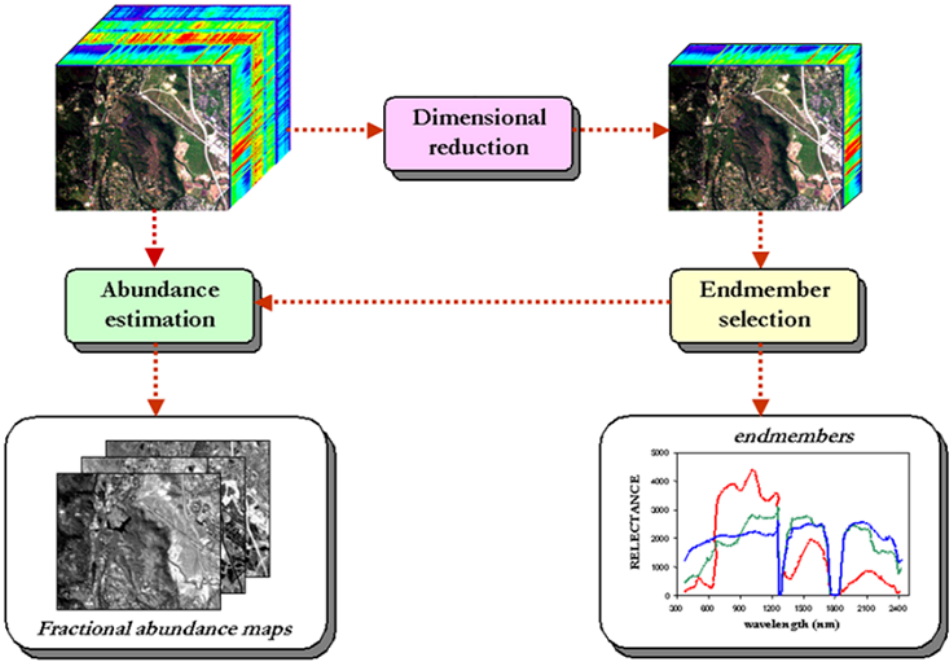


Figure 3.15: Spectral unmixing chain implemented in this subchapter in WCPS

3.3.4 Full spectral unmixing chain

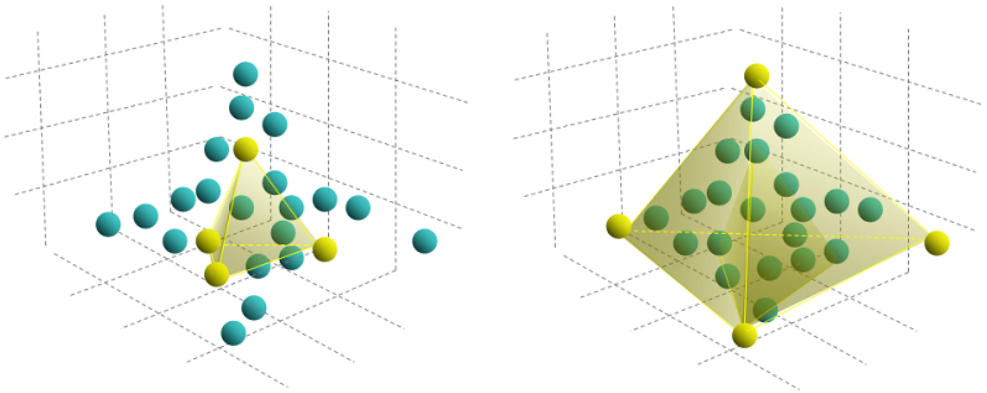
In this section, the full spectral unmixing chain implemented in this subchapter is described in detail. It consists of the following steps: dimensionality reduction, endmember extraction and abundance estimation. The chain is graphically illustrated in Fig. 3.15, and the different steps of the chain are explained in more details below.

3.3.4.1 Dimensional reduction

In this subchapter, the dimensional reduction step is performed using principal component analysis (PCA), a popular tool for feature extraction in different areas including remote sensing. The implementation of PCA adopted in this subchapter is the one described in [79]. This technique is an approximation to PCA from which principal components can be derived in sequential fashion.

3.3.4.2 Endmember selection

For the endmember identification part, we rely on the well-known N-FINDR algorithm [78], which is a standard for the hyperspectral imaging community. 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 3.16(a)]. Every pixel in the image must be evaluated in order to refine the estimate of endmembers, looking for the set of pixels that maximizes the volume of the simplex defined by selected endmembers. 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 until there are no more endmember replacements [see Fig. 3.16(b)].



(a) (b)

Figure 3.16: The N-FINDR algorithm illustrated in a 3-D representation of a hyperspectral data cube in which each point represents a pixel vector in the 3-D spectral space. (a) Initial random initialization of the algorithm. (b) Final result of the algorithm after convergence.

3.3.4.3 Abundance estimation

Finally, abundance estimation is carried out using fully constrained least squares spectral unmixing (FCLSU). Once a set of p endmembers has been extracted with the N-FINDR algorithm, their correspondent abundance fractions in a specific pixel vector of the scene can be estimated (in least squares sense) by a simple unconstrained expression [52]. However, it should be noted that the fractional abundance estimations obtained this way do not satisfy the abundance sum-to-one and abundance non-negativity constraints expected for the linear model to work properly. As indicated in [52], a non-negative constrained least squares algorithm can be used to obtain a solution to the problem with non-negativity constraint in abundance estimation, while a fully constrained estimate can also be obtained from the previous estimate.

3.3.5 WCPS scripting examples

Before describing the experimental results obtained after porting the full spectral unmixing chain to WCPS, we provide in this section some scripting examples indicating how end-users can interact with this framework to run the considered algorithms. It is important to emphasize that WCPS supports many multispectral and hyperspectral instruments such as the Advanced Land Imager (ALI) and Hyperion flying on the EO-1 spacecraft [70]. Basic band manipulations, ratios, and thresholding operations are supported. Other processing techniques include calculations related with Normalized Differential Vegetation (NDVI), Normalized Difference Snow Index (NDSI), Photochemical Reflectance index (PRI), Landsat band ratios, tesseled cap transformations, vertical bands destreaking, and so forth [80]. Algorithm 3 shows an example of a WCPS script for computing the normalized NDVI as described in [80].

Atmospheric correction is another feature provided by the WCPS. The EO-1 processing now automatically includes atmospheric correction of ALI and Hyperion L1G data. Users have the capability to re-process the data with different parameters than the ones automatically selected by the operational system. FLAASH [81] and ATREM [82] packages are currently used for atmospheric correction. Specifically, a fast, C version of FLAASH with lookup tables is being used. Very recently, pansharpening capabilities have also been added to WCPS. The current algorithm uses a version of the Intensity-Hue-

```
- - ==========================================
- - title: ndvi
- - description: Normalized Difference Vegetation Index (Deering, 1978)
- - author: Jim Tucker
- - openid:
- - date: 2010-07-09
- - version: 1.0
- - tags: hyperion_l1r_ac, ndvi, 45, 32
- - duration: 10s
- - ==========================================

local b45 = channel(45)
local b32 = channel(32)
local rgba = normalized_difference_ratio( b45, b32, 'ndvi' )

- - Flip image and write it
local flip = im.ProcessFlipNew( rgba )

- - Load color lookup table as an image
local legend = LoadColorPaletteImage( 'ndvi' )

- - Double its size
local legend2x = im.ImageCreateBased( legend, legend:Width()*2, legend:Height()*2, leg-
end:ColorSpace(), 0 )

im.ProcessResize( legend, legend2x, 0 )

- - Insert it
im.ProcessInsert( legend2x, flip, flip, 19, 511 )

- - Write the file
write_file( flip )

- - ==========================================
```

Figure 3.17: Algorithm 3. Example of a WCPS script intended to compute the normalized NDVI as described in [1].

Saturation that allows us to fuse the 10m ALI Pan band and three other multi-spectral channels at 30m-resolution to generate a high-resolution composite visible for example. This has currently been used operationally for a Namibia Flood Pilot. For users in Namibia, Internet bandwidth is an issue. It is extremely difficult to download very large files. For that particular case, the WCPS can actually tile the finished product into keyhole markup language (KML) super-overlays that can be displayed in Google Earth. We are using Python scripts (from MapTiler) controlling the Geospatial Data Abstraction Library (GDAL).

More complex algorithms can also be supported by the WCPS such as the spectral angle mapper [17]. A user-defined signature can get uploaded to the system and detected in a particular scene. Alternatively, the system could detect a full set of p endmembers in a hyperspectral scene using the N-FINDR [78] algorithm, followed by FCLSU for abundance estimation [52]. The unmixing chain has been integrated with the WCPS and will be evaluated in the following section using a 2001 scene acquired by EO-1 in

```
- - ========================================
- - title: unmixing_chain
- - description: Spectral UnMixing Chain
- - author: Sergio Sanchez, Sergio Bernabe, Antonio Plaza
- - openid:
- - date: 2012-01-16
- - version: 1.0
- - tags: hyperion_lr_ac, unmixing
- - ========================================

local selected_channels = {
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

- - Model Selection:
- - unconstrained: The abundances are unconstrained and can become any numeric
- - value.
- - constrained: The sum of abundances equals 1.
- - fully_constrained: The sum of abundances equals 1 and abundance values cannot be
- - less than zero. This is the default unmixing algorithm.

local model = "fully_constrained"

local iterations = 400

- - number of requested endmembers
local endmembers = 7

- - color lookup table to apply to abundance maps
local clut = "prism"

spectral_unmixing( selected_channels, iterations, model, endmembers, clut )

- - ========================================
```

Figure 3.18: Algorithm 4. Example of a WCPS script intended to compute a fully constrained spectral unmixing chain

the vicinity of Chief's Island Okavango Basin in Botswana. From a user perspective, the WPCS script that executes the full spectral unmixing chain described in section 3.3.4 is illustrated in Algorithm 4.

As shown by Algorithm 4, the WCPS script for running the hyperspectral unmixing chain can be applied to atmospherically corrected data. It first defines the set of channels that will be selected for the processing of the considered hyperspectral scene. Here, noisy data channels and water absorption

data channels are removed, retaining a total of 145 spectral channels for experiments. The endmembers are extracted with N-FINDR, and the maximum numbers to be identified are also specified in the script (in this case, p=7). Three different types of unmixing models are available in the script: unconstrained (meaning that the abundance sum-to-one and non-negativity constraints are not imposed when conducting the abundance estimation step); constrained (meaning that only the sum-to-one is imposed when conducting the abundance estimation); and fully constrained (meaning that both the sum-to-one and non-negativity constraints are imposed when conducting the abundance estimation). The default model used in experiments is the fully constrained one. Another parameter that needs to be set in the script is the local number of iterations, which refers to the number of iterations executed by the iterative implementation of the PCA-based dimensionality reduction module that needs to be run prior to the N-FINDR endmember selection. Finally, a color lookup table is applied to improve visualization of the output of the script, which highlights the fully constrained fractional abundance maps estimated for the input hyperspectral scene. As shown by Fig. 3.18, the execution of the unmixing chain in WCPS is performed in modular and easy-to-use fashion.

3.3.6 Experimental results

The WCPS implementation of our considered spectral unmixing chain has been preliminarily tested using a 2001 scene acquired by EO-1 in the vicinity of Chief's Island Okavango Basin in Botswana. These data, along with reference data, are available online[21]. The investigations on this scene have been centered on the seasonal flooding of the Okavango Delta. The extreme inaccessibility of the area makes it an ideal candidate for the use of remote sensing data to map the annual flooding and land cover in the region. This is the world's largest inland delta. It is fed by the Okavango River, which originates in Angola's western highlands. The catchment area for the Okavango River lies in three countries (Angola, Namibia, and Botswana) and has a total area of approximately 325,000 km^2. The Delta extends 250 km along its radial axis and extends over 22,000 km^2 in area[22].

A field campaign, conducted in March 2001 by the University of Texas Center for Space Research (CSR) in conjunction with the University of Botswana Harry Oppenheimer Okavango Research Center (HOORC), focused on a portion of the Delta known as Chief's Island. One of the main focus of this study was to characterize the annual flooding by mapping the spatial patterns within the lower Delta as well as to identify small-scale responses of vegetation. Of particular importance is also the analysis of fire scar areas, meaning the scar or sign left by the fire in some regions after burning. The EO-1 Hyperion data set considered in the experiments consisted of 1476 $\times$ 256 pixels (with 242 spectral channels) and with 14 different land-cover types consisting of seasonal swamps, occasional swamps, and drier woodlands located in the distal portion of the delta. Uncalibrated and noisy data channels that cover water absorption features were removed, resulting in 145 features (as indicated in the script shown in Fig. 3.18). The land-cover classes in this study were chosen to reflect the impact of flooding on vegetation in the study area.

For illustrative purposes, Fig. 3.19(a) shows a pseudo-RGB image of the considered scene. Fig. 3.19(b-f) shows some of the abundance maps obtained after processing the scene with the fully constrained unmixing chain using p=18 (the number of endmembers was estimated using the virtual dimensionality concept in [63]. The maps in Fig. 3.19(b-e) are in agreement with the classification results obtained for the same scene in [83]. A distinguishing feature of the results presented in Fig. 3.19 with regards to

[21] http://www.csr.utexas.edu/hyperspectral/data/Botswana

[22] http://eo1.gsfc.nasa.gov/new/validationreport/Technology/Documents/Tech.Val.Report/Science_Summary_Crawford.pdf

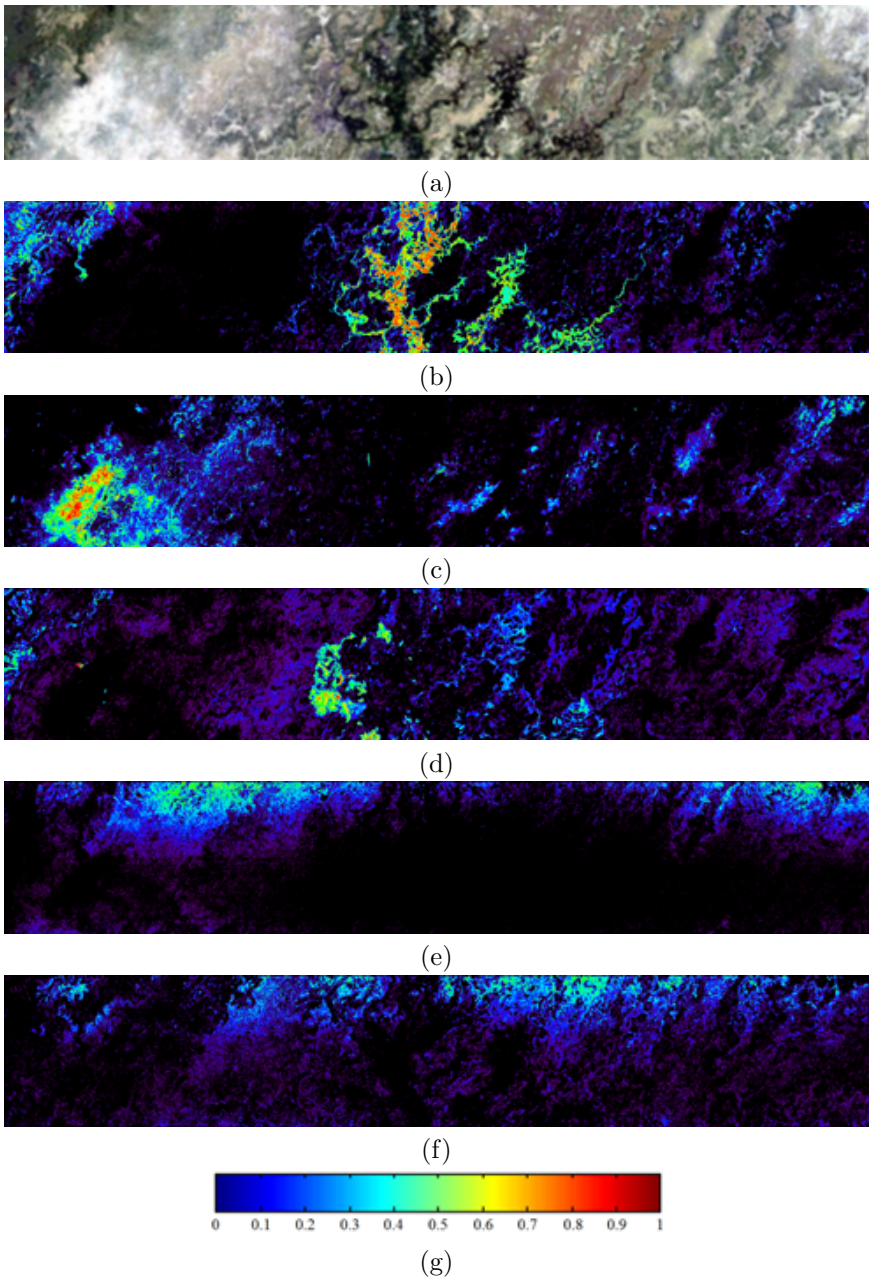


Figure 3.19: (a) Pseudo-RGB color image of the Hyperion Botswana scene. (b-f) Some of the abundance maps estimated by the WCPS implementation of the full spectral unmixing chain represented using the color map in (g).

those reported in [83] is that the sub-pixel fractions of each of the considered classes can be estimated in the experimental results reported in this subchapter.

On the other hand, Table 3.7 shows the spectral angle scores between some of the endmembers derived by the considered unmixing chain and their corresponding reference spectra, obtained by averaging the pixels belonging to each reference ground class in the online data. The interval in which the spectral values are comprised is $[0°, 90°]$. Hence, the closest the angles in Table 3.7 to $0°$, the higher the measured

Table 3.7: Spectral angle values (in degrees) between the endmembers extracted by the considered unmixing chain from the EO-1 Hyperion Botswana scene and their corresponding reference spectra, obtained by averaging the pixels belonging to each class in the reference data available online for this scene.

Water	Soils	Fire scar	Woodlands	Floodplain
8.16°	2.68°	1.58°	1.65°	5.82°

spectral similarity. As shown by Table 3.7, the spectral similarity between the extracted endmembers and the average spectra in the reference ground classes identified in the field campaign conducted over the study area is very high, indicating a good agreement between the endmembers extracted from the satellite data and the reference classes identified on the ground.

Despite the encouraging results obtained, further experiments with additional Hyperion scenes are required in order to fully substantiate the contributions introduced by the novel developments presented in this subchapter. In fact, a major goal in our current research is to illustrate the benefits that can be obtained by transitioning theoretical algorithms (such as the considered hyperspectral unmixing chain) to the science applications world using the WCPS framework. Additional benefits will be obtained during flight demonstrations, as the proposed framework is also implemented in the form of a flight test-bed. This is in fact the main societal benefit to the community that we intend to demonstrate in this contribution.

To conclude this section, we emphasize that the integration of the unmixing chain in the WCPS framework as part of the NASA SensorWeb suite of web services is just the start of an international collaboration in which many more processing algorithms will be made available to the community through this service. It is our hope that more scientists and researchers contribute to the NASA WCPS, an open image processing framework that allows dynamic upload of processing algorithms (in addition to the unmixing chain that we addressed in this contribution) and data collected from airborne and/or spaceborne platforms. As a result, this work is not so much focused on the theory and results of unmixing (which have been widely demonstrated in other contributions) but also on the process and value of the proposed contribution for ground processing on the cloud, as well as onboard generation of low-latency products for new airborne and spaceborne missions.

It is our feeling that the results reported in this section are sufficient to demonstrate the expected contributions and added value resulting from having a well-consolidated approach for hyperspectral image analysis (from band arithmetic and ratios to spectral unmixing) available as a web service from the NASA SensorWeb. This OGC suite of standardized API and services offers the potential to expand its use and consolidate its role as one of the most successful services for processing and distributing actionable products to the end-user community. In effect, it will realize the societal benefit goal of applying Science and Technology to the benefit of the people. Several pilots such as in Namibia and the Caribbean currently demonstrate this value for disaster management. Most importantly, the described technological development is not circumscribed to EO-1 Hyperion but also to future missions such as HyspIRI. In this regard, we would like to emphasize that the tools used in the development are general and consolidated enough to facilitate the integration on global missions or missions with different resolutions and sensor capabilities. However, aspects of standardization and efficient implementation will be of great importance in order to be able to extrapolate the developed cloud implementation to process data collected from other instruments. It is also our feeling that important synergies exist in the data

processing chains of EO-1 Hyperion and HyspIRI (the former has been widely used as a demonstrator for the latter [73]) that will greatly facilitate the desired integration.

3.3.7 Summary and future research lines

In this subchapter, we have illustrated how a full hyperspectral unmixing chain made up of the following steps: 1) dimensionality reduction; 2) automatic endmember identification; and 3) fully constrained abundance estimation can be integrated with the Web Coverage Processing Service (WCPS), an image processing framework that can run on the cloud, as part of the NASA SensorWeb suite of web services. The ultimate goal has been to show that, with the support of the proposed cloud implementation, the WCPS can help quickly process large amounts of data and deliver finished products (in this case, obtained through spectral unmixing) to the end-user at an extremely low cost. The experimental results with a hyperspectral scene collected over the Okavango Basin in Botswana suggest the (present and future) potential of spectral unmixing for improved exploitation of spaceborne hyperspectral data by providing unmixing results which are in agreement with previous classification studies conducted for the same scene.

As a future research line, we will experiment with real-time implementations of the considered unmixing chain on other different kind of hardware devices for onboard processing, such as a commercial TileraPro64 multi-core processor board, which actually performed well in a flight environment. This will be achieved by using the adaptive environment for super-compiling with optimized parallelism (AESOP). Other specialized hardware platforms such as field-programmable gate arrays (FPGAs) [28] or commodity graphics processing units (GPUs) [21] will be also investigated. For instance, the full unmixing chain reported in this subchapter has already been ported onto NVidia GPUs [84], and the N-FINDR algorithm (along with other endmember selection algorithms [68] and abundance estimation [85] algorithms) have been successfully ported to FPGA platforms for real-time exploitation. In the future we are also planning on exploring additional mechanisms to perform the dimensionality reduction step needed by the considered unmixing chain, e.g., using the wavelet transform which has been already successfully been used for this purpose in the literature [86].

Chapter 4

Performance Comparison of Efficient Algorithms on Parallel Architectures. Case of Study: Unmixing Problem

Outline - One of the main problems in the analysis of remotely sensed hyperspectral data cubes is the presence of mixed pixels, which arise when the spatial resolution of the sensor is not able to separate spectrally distinct materials. Due to this reason, spectral unmixing has become one of the most important tasks for hyperspectral data exploitation. However, unmixing algorithms can be computationally very expensive, a fact that compromises their use in applications under real-time constraints. For this purpose, in this chapter two efficient implementations of a full hyperspectral unmixing chain are developed on two different kinds of high performance computing architectures: graphics processing units (GPUs) and multi-core processors. The proposed full unmixing chain is composed for three stages: (i) estimation of the number of pure spectral signatures or *endmembers*, (ii) automatic identification of the estimated endmembers, and (iii) estimation of the fractional abundance of each endmember in each pixel of the scene. The two computing platforms used in this work are inter-compared in the context of hyperspectral unmixing applications. The GPU implementation of the proposed methodology has been implemented using CUDA and the `cuBLAS` library, and tested on two different GPU architectures: NVidia GeForce GTX 580 and NVidia Tesla C1060. It provides real-time unmixing performance in two different analysis scenarios using hyperspectral data collected by NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) over the Cuprite mining district in Nevada and the World Trade Center complex in New York City. The multi-core implementation, developed using the applications program interface (API) `OpenMP` and the Intel Math Kernel Library (MKL) used for matrix multiplications, achieved near real-time performance in the same scenarios. A comparison of both architectures in terms of performance, cost and mission payload considerations is given based on the results obtained in the two considered data analysis scenarios.

Part of this chapter has been published in:

S. Bernabé, S. Sánchez, A. Plaza, S. López, 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*, vol. 6, no. 3, pp. 1386–1398, June 2013 [JCR(2012)=2.874].

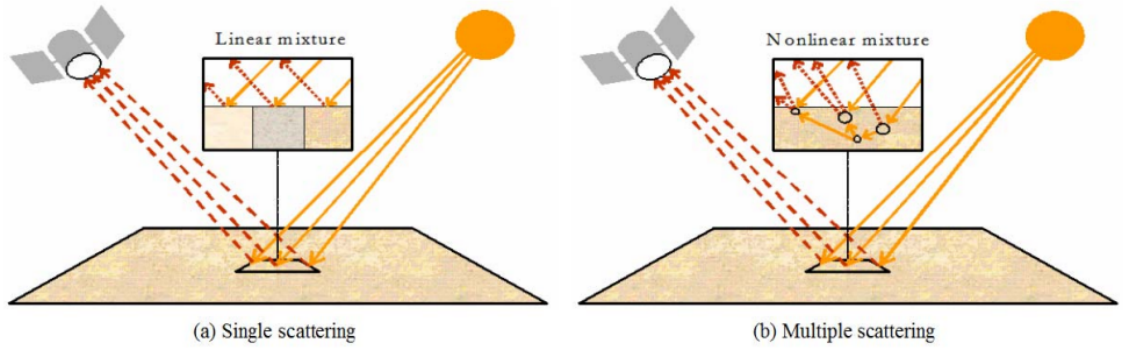


Figure 4.1: Linear (a) versus nonlinear (b) mixture models in remotely sensed hyperspectral imaging.

4.1 Overview

Hyperspectral imaging instruments are capable of collecting hundreds of images, corresponding to different wavelength channels, for the same area on the surface of the Earth [87]. For instance, NASA is continuously gathering imagery data with Earth observation instruments such as the Jet Propulsion Laboratory's AVIRIS, which is able to record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 micrometers) of reflected light in an area 2 to 12 kilometers wide and several kilometers long, using 224 spectral channels [71]. One of the main problems in the analysis of hyperspectral data cubes is the presence of mixed pixels [7, 19], which arise when the spatial resolution of the sensor is not fine enough to separate spectrally distinct materials. In this case, several spectrally pure signatures (endmembers) are combined into the same (mixed) pixel. Spectral unmixing [17, 26] involves the separation of a pixel spectrum into its endmember spectra, and the estimation of the abundance value for each endmember [52, 88]. A popular approach for this purpose in the literature has been linear spectral unmixing, which assumes that the endmember substances are sitting side-by-side within the field of view of the imaging instrument [see Fig. 4.1(a)]. On the other hand, the nonlinear mixture model [89–91] assumes nonlinear interactions between endmember substances [see Fig. 4.1(b)]. In practice, the linear model is more flexible and can be easily adapted to different analysis scenarios. Let $\mathbf{y}$ be a pixel vector given by a collection of values at different wavelengths. In the context of linear spectral unmixing, such vector can be modeled as:

$$\mathbf{y} \approx \mathbf{M}\alpha + \mathbf{n} = \sum_{i=1}^{p} \mathbf{e}_i \alpha_i + \mathbf{n}, \tag{4.1}$$

where $\mathbf{M} = \{\mathbf{e}_i\}_{i=1}^{p}$ 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. The spectral unmixing chain considered in this work comprises three steps (see Fig. 4.2): A) estimation of the number of pure spectral signatures (*endmembers*), p, in the hyperspectral scene; B) identifying a collection of $\mathbf{M} = \{\mathbf{e}_i\}_{i=1}^{p}$ endmembers, and C) estimating the abundances, in which the fractional coverage of each endmember is estimated for each pixel. The estimation error can be computed by reconstructing the original image (using the extracted endmembers and the derived abundances) and comparing the reconstructed image with the original one.

In recent years, several techniques have been proposed to solve the aforementioned problem under the linear mixture model assumption (see [53, 55, 78, 92–94], among several others), but all of them are quite

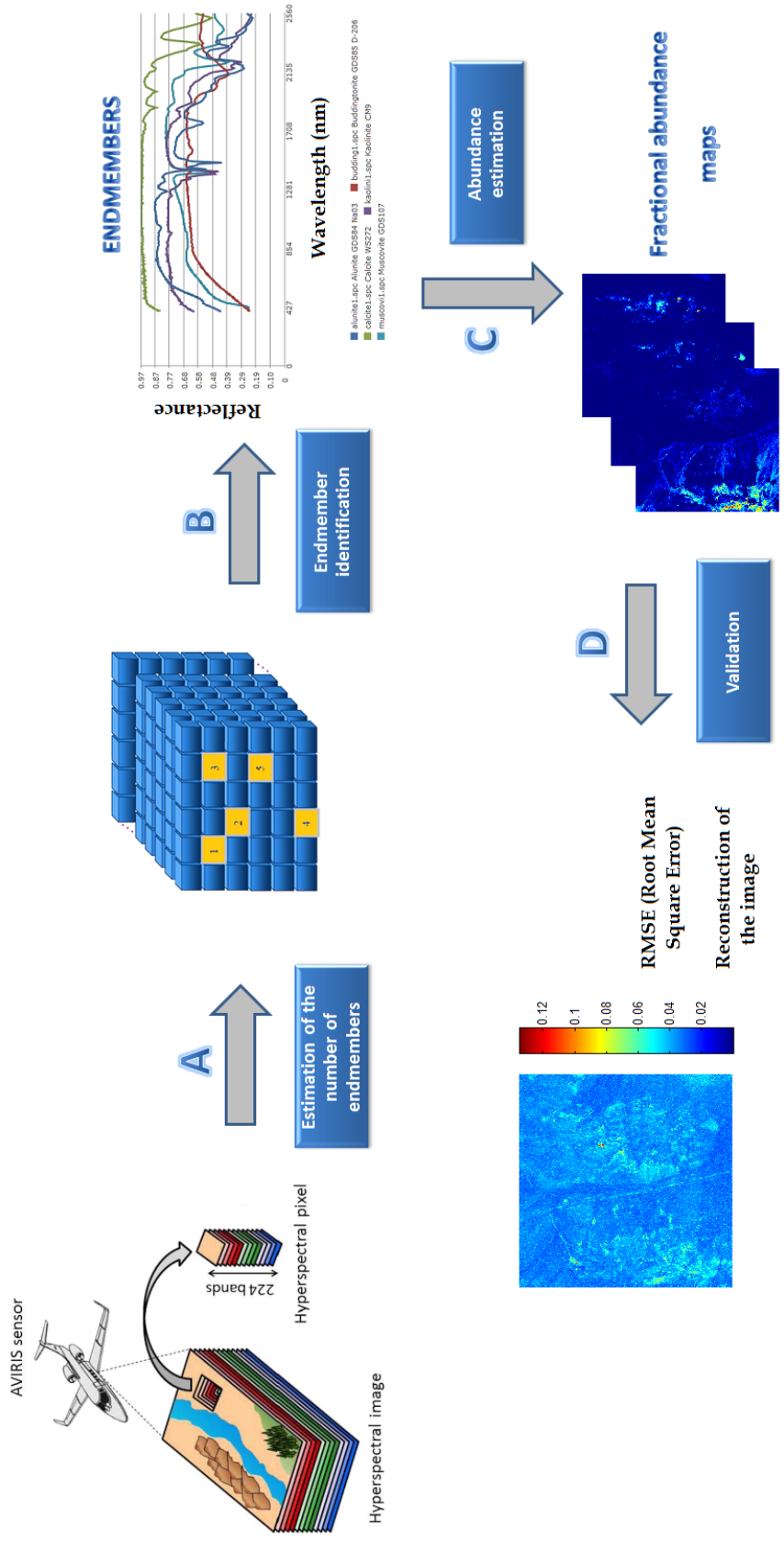


Figure 4.2: Block diagram illustrating the full hyperspectral unmixing chain considered in this work.

expensive in computational terms. Although these techniques map nicely to high performance computing systems such as commodity clusters [24, 25], these systems are difficult to adapt to onboard processing requirements introduced by applications with real-time constraints such as wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination [57,95]. In those cases, low-weight integrated components such as field programmable gate arrays (FPGAs) [27–29], which offer a good compromise in terms of mission payload, have revealed as a feasible option, but one that generally requires a significant effort from the design and programmability point of view. Possible alternatives are multi-core processors [30–32] and commodity graphics processing units (GPUs) [33, 34], which offer highly relevant computational power at low cost, this offering the opportunity to bridge the gap towards real-time analysis of remotely sensed hyperspectral data [60, 96, 97].

In this chapter, two computationally efficient implementations of a full hyperspectral unmixing chain are developed on GPUs and multi-core processors. Although previous work has discussed the implementation of unmixing algorithms on GPUs [18], the implementation of a full hyperspectral unmixing chain on multi-core processors has not been discussed in previous contributions, to the best of our knowledge. The two considered platforms are inter-compared in the context of hyperspectral unmixing applications. In our comparisons, we have selected both domestic (e.g., multi-core processor i7 and NVidia GeForce GTX 580 GPU) and professional platforms (e.g., multi-core Xeon processor and NVidia Tesla C1060 GPU). The study reveals that GPUs and multi-core processors can provide real-time unmixing performance. The implementations on GPUs have been carried out using NVidia CUDA and the `cuBLAS` library[1], an implementation of `BLAS` (basic linear algebra subprograms) on top of NVidia CUDA, while the multi-core implementations have been developed using the API `OpenMP`[2] and Intel's Math Kernel Library (MKL)[3].

The remainder of the chapter is organized as follows. Section 4.2 describes the different modules that conform to the proposed unmixing chain. It is important to emphasize that our GPU and serial ATDCA-GS versions have been described in section 3.1.3. Sections 4.3 and 4.4 describe the GPU and multi-core implementations of these modules, respectively. Section 4.5 presents an experimental evaluation of the proposed implementations in terms of both unmixing accuracy and parallel performance, using two different hyperspectral scenes with reference data and collected by AVIRIS. Finally, section 4.6 concludes the chapter with some remarks and hints at plausible future research lines.

4.2 Unmixing chain algorithms

4.2.1 Virtual dimensionality (VD) algorithm for estimation of the number of endmembers

Let us denote by $\mathbf{Y} \equiv [y_1, y_2, \cdots, y_N]$ a hyperspectral image with N pixel vectors, each with L spectral channels. The VD first calculates the eigenvalues of the covariance matrix $\mathbf{K}_{L\times L} = 1/\mathrm{N}(\mathbf{Y}-\overline{\mathbf{Y}})^T(\mathbf{Y}-\overline{\mathbf{Y}})$ 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, for each of the spectral channels in the original hyperspectral image $\mathbf{Y}$. If a distinct spectral signature makes a contribution to the eigenvalue-represented signal energy in one spectral channel, then its associated correlation eigenvalue will be greater than its corresponding covariance-eigenvalue in this particular data channel. Otherwise, the correlation eigenvalue would be very close to the covariance-eigenvalue, in which case only noise would be present in this particular

[1] http://developer.nvidia.com/cuBLAS
[2] http://openmp.org
[3] http://software.intel.com/en-us/intel-mkl

data channel. By applying this concept, a Neyman-Pearson detector [63] is introduced to formulate the issue of whether a distinct signature is present or not in each of the spectral channels of $\mathbf{Y}$ as a binary hypothesis testing problem, where a so-called Neyman-Pearson detector is generated to serve as a decision maker based on a prescribed P_F (i.e., false alarm probability). In light of this interpretation, the issue of determining an appropriate estimation $\widehat{p}$ for the number of endmembers is further simplified and reduced to a specific value of P_F that is presented by the Neyman-Pearson detector.

4.2.2 Unconstrained least-squares (UCLS) algorithm for abundance estimation

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

$$\alpha = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y}. \tag{4.2}$$

Two additional constrains can be imposed into the model described in (4.2), these are the abundance non-negativity constraint (ANC), i.e., $\alpha_i \geq 0$, and the abundance sum-to-one constraint (ASC), i.e., $\sum_{i=1}^{p} \alpha_i = 1$. However, in this work we focus on the unconstrained estimation only as it is much faster and it has been shown in practice to provide satisfactory results if the model endmembers are properly selected.

4.3 GPU implementation

GPUs can be abstracted in terms of a stream model, under which all data sets are represented as streams (i.e., ordered data sets). Fig. 4.3 shows the architecture of a GPU, which can be seen as a set of multiprocessors (MPs). Each multiprocessor is characterized by a single instruction multiple data (SIMD) architecture, i.e., in each clock cycle each processor executes the same instruction but operating on multiple data streams. Each processor has access to a local shared memory and also to local cache memories in the multiprocessor, while the multiprocessors have access to the global GPU (device) memory. 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 of synchronization. The kernels can perform a kind of batch processing arranged in the form of a grid of blocks (see Fig. 4.4), where each block is composed by a group of threads which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. As a result, there are different levels of memory in the GPU for the thread, block and grid concepts (see Fig. 3.1, in section 3.1.3). There is also a maximum number of threads that a block can contain but the number of threads that can be concurrently executed is much larger (several blocks executed by the same kernel can be managed concurrently, at the expense of reducing the cooperation between threads since the threads in different blocks of the same grid cannot synchronize with the other threads). With the above ideas in mind, our GPU implementation of the hyperspectral unmixing chain comprises three stages: 1) GPU implementation of VD; 2) GPU implementation of ATDCA-GS; and 3) GPU implementation of UCLS. In the following, we describe the GPU implementations of VD and UCLS, as the GPU implementation of ATDCA-GS has already been described in section 3.1.3 of this thesis.

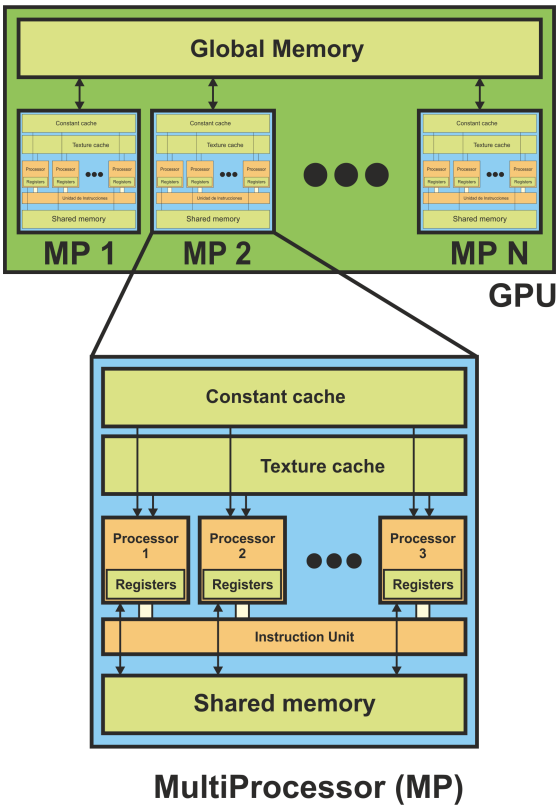


Figure 4.3: Schematic overview of a GPU architecture, which can been seen as a set of multiprocessors (MPs).

4.3.1 GPU implementation of VD

Once the full hyperspectral image $\mathbf{Y}$ is loaded pixel by pixel from disk to the main memory of the GPU, the first step is to calculate the covariance matrix $\mathbf{K}_{L\times L}$. For this purpose, we need to calculate the mean value $\overline{\mathbf{Y}}$ of each data channel of the image and subtract this mean value from all the pixels in the same channel. To perform this calculation in the GPU, we use a kernel called *mean_pixel* configured with as many blocks as the number of data channels L in the hyperspectral image. 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 to the same data channel. Once this process is completed, another thread divides the computed value by the number of pixels in the original image, N, and the overall mean value is obtained. The resulting mean values of each data channel $\overline{Y}$ are stored in a structure as they will be needed for the calculation of the covariance matrix $\mathbf{K}_{L\times L}$ in the GPU by means of a matrix multiplication operation $(\mathbf{Y}-\overline{\mathbf{Y}})^T(\mathbf{Y}-\overline{\mathbf{Y}})$. This operation is performed using the `cuBLAS` library. Specifically, the `cublasSgemm` function of `cuBLAS` is used. The next step is to calculate the correlation matrix $\mathbf{R}_{L\times L}$ in the GPU. To achieve this, we use a kernel *correlation* which launches as many threads as elements in $\mathbf{R}_{L\times L}$, 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.

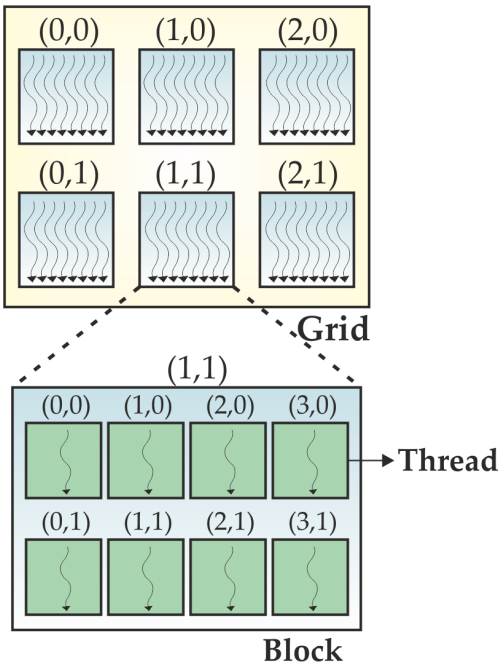


Figure 4.4: Batch processing in the GPU: grids of blocks of threads, where each block is composed by a group of threads.

4.3.2 GPU implementation of UCLS

The GPU implementation of UCLS can be summarized by the following steps:

1. The first step is to calculate the operation $(\mathbf{M}^T\mathbf{M})$, where $\mathbf{M} = \{\mathbf{e}_i\}_{i=1}^{p}$ is formed by the p endmembers extracted by the ATDCA-GS. This is performed by a kernel called `MtxM`. The inverse of this operation is calculated in the CPU mainly due to two reasons: i) its computation is relatively fast, and ii) the inverse operation remains the same throughout the whole execution of the code. A new kernel called `Mxinv` is now used to multiply $\mathbf{M}$ by its inverse.

2. The result calculated in the previous step is now multiplied by each pixel $\mathbf{y}$ in the hyperspectral image, thus obtaining a set of abundance vectors $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_p]$, each containing the fractional abundances of the p endmembers in $\mathbf{M}$. This is accomplished in the GPU by means of a specific kernel, called `get_abundances`, which produces p abundance maps.

4.4 Multi-core implementation

A multi-core processor is a single CPU with two or more independent processors (called *cores*), which are the units that read and execute program instructions. The multiple cores can run multiple instructions at the same time, increasing the overall speed for programs amenable to parallel computing. In our implementation of the considered unmixing chain, we used `OpenMP` which is an API intended to explicity address multithreaded, shared-memory parallelism. In `OpenMP` the users specify the regions in the code that are suitable for parallel implementation. The user also specifies necessary synchronization operations, such as locks or barriers, to ensure correct execution of the parallel region. At runtime the threads are executed in different processors but sharing the same memory and address space. In

the following, we briefly summarize the main techniques used in the multi-core implementation of the considered unmixing chain:

- In our unmixing chain, we have several matrix multiplications. Mainly in the VD algorithm, we have a multiplication of high dimensionality in the *covariance* operation (the most expensive one of the algorithm). Our idea is based on the work presented in [98], where the matrix multiplication routine is included using two levels of parallelism (`OpenMP+BLAS`). The first level of parallelism is set using the API `OpenMP` and the second level is achieved by invoking the *dgemm* routine of a multithreading implementation of `BLAS` (the MKL library of optimized math routines has been used in the experiments).

- One of the main techniques adopted in the `OpenMP` implementation of the considered unmixing chain is the use of locking routines. For instance, these routines are used in the ATDCA-GS algorithm to calculate the brightest pixel in the scene and the maximum projection operation.

- Another strategy adopted in the `OpenMP` implementation of the considered unmixing chain is the use of `parallel for` directives, which indicate to the compiler that the structured block of code should be executed in parallel on multiple threads. Each thread will execute the same instruction stream, however not necessarily the same set of instructions. These directives are used with the locking routines and to implement the different steps for the UCLS algorithm in section 4.3.2.

4.5 Experimental results

4.5.1 Analysis of algorithm precision

The number of endmembers to be extracted from the AVIRIS Cuprite image was estimated as $p = 19$ after calculating the virtual dimensionality (VD) [50], which is the first stage in our proposed unmixing chain. Table 3.2 shows the spectral angles (in degrees) between the most similar endmembers extracted by the ATDCA-GS and the reference USGS spectral signatures available for this scene. The range of values for the spectral angle is [0°, 90°]. As shown by Table 3.2, the endmembers extracted by the ATDCA-GS algorithm are very similar, spectrally, to the USGS reference signatures, despite the potential variations (due to possible interferers still remaining after the atmospheric correction process) between the ground signatures and the airborne data. For illustrative purposes, the fractional abundance maps obtained for the same representative minerals in the Cuprite mining district are displayed in Fig. 4.5(a-e). Since no reference information is available regarding the true abundance fractions of minerals in the AVIRIS Cuprite data, no quantitative experiments were conducted although the obtained mineral maps exhibit similar correlation with regards to previously published maps[4].

In any case, the results of spectral unmixing can also be evaluated in terms of the quality of the reconstruction of the original data set using the extracted endmembers, the estimated fractional abundances, and the linear mixture model. These results have been discussed in a previous work [21]. In this case, the metric employed to evaluate the goodness of the reconstruction is the root mean square error (RMSE) obtained after comparing the original scene with the reconstructed one. This metric is based on the assumption that a set of high-quality endmembers (and their corresponding estimated abundance fractions) may allow reconstruction of the original scene with higher precision compared to a set of low-quality endmembers. In this case, the original scene is used as a reference to measure the

[4] http://speclab.cr.usgs.gov/cuprite.html

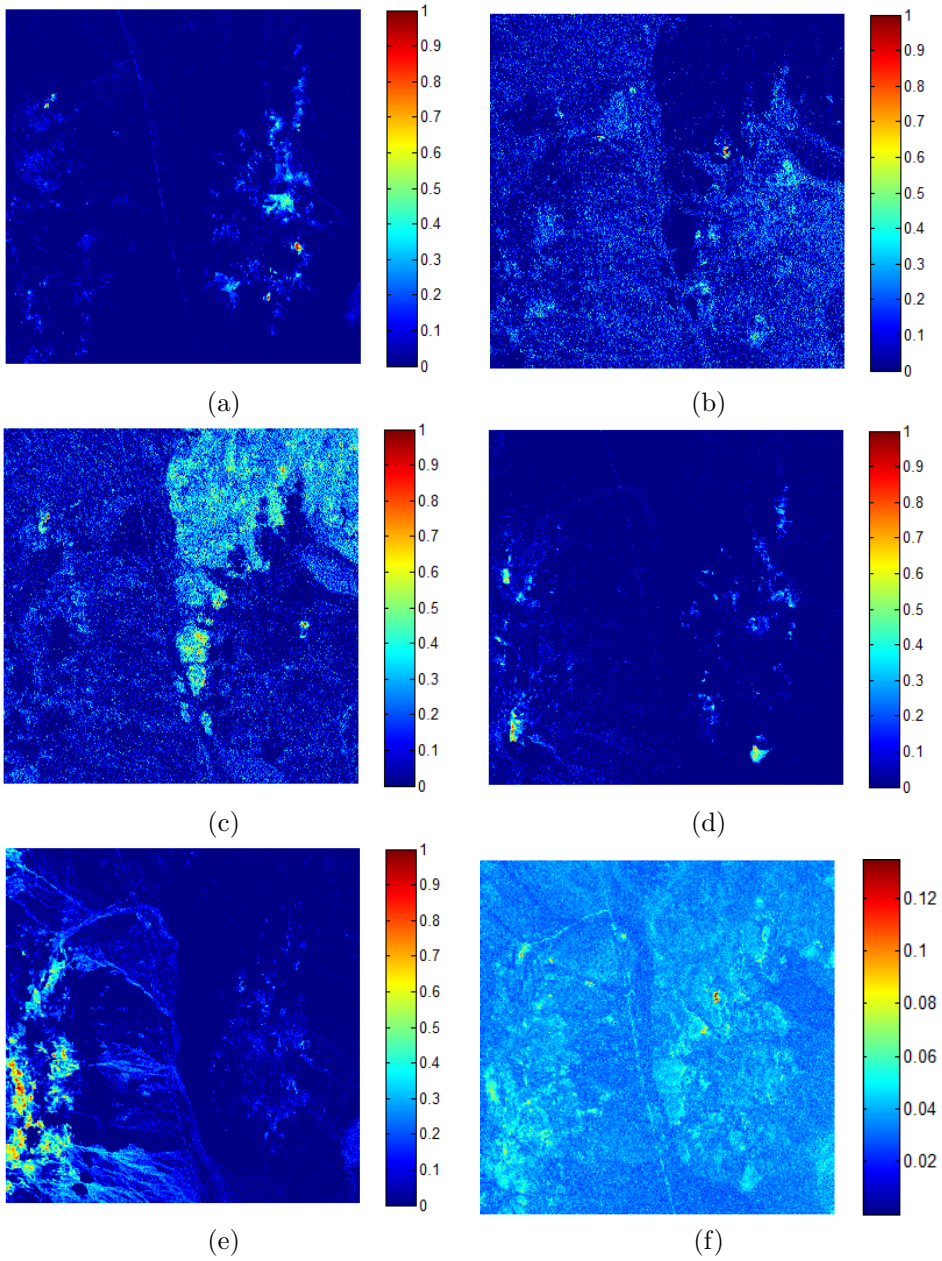


Figure 4.5: Abundance maps extracted from the Cuprite scene for different minerals: (a) Alunite. (b) Budinggtonite. (c) Calcite. (d) Kaolinite. (e) Muscovite. (f) Per-pixel RMSE obtained in the reconstruction process of the AVIRIS Cuprite scene using $p = 19$ endmembers (the overall RMSE in this case was 0.0361).

fidelity of the reconstructed version on a per-pixel basis. For illustrative purposes, Fig. 4.5(f) graphically represents the per-pixel RMSE obtained in the reconstruction process of the AVIRIS Cuprite scene. The RMSE map in Fig. 4.5(f) generally reveals a good spatial distribution of the error, although some anomalous endmembers appear to be missing. In any event, the per-pixel RMSE values are quite low, indicating a good overall compromise in the reconstruction of the scene.

A similar experiment was also conducted for the AVIRIS WTC scene. Table 3.1 shows the spectral

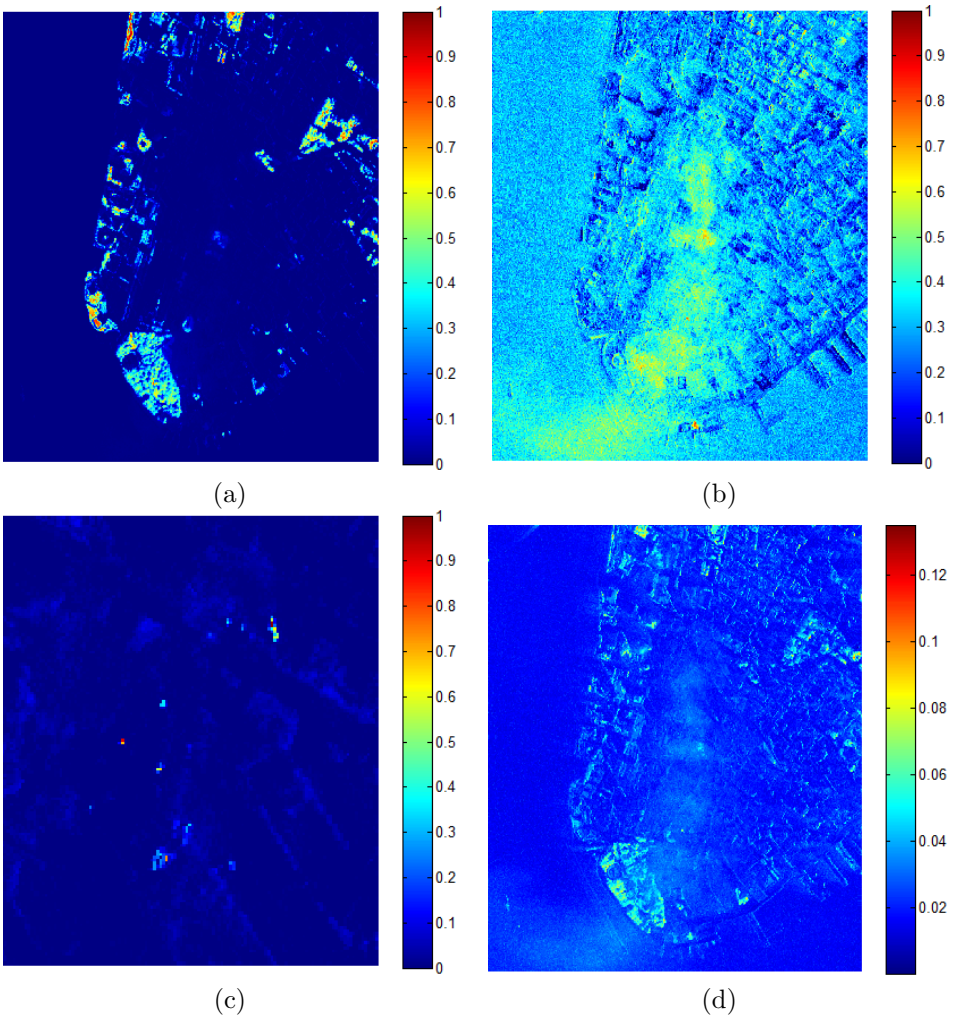


Figure 4.6: Abundance maps extracted from the WTC scene for different targets: (a) Vegetation. (b) Smoke. (c) Fire. (d) Per-pixel RMSE obtained in the reconstruction process of the AVIRIS WTC scene using $p = 31$ endmembers (the overall RMSE in this case was 0.0216).

angles (in degrees) between the most similar endmember pixels detected by ATDCA-GS and the pixel vectors at the known target positions in the scene, labeled from 'A' to 'H' in the rightmost part of Fig. 3.2. The number of endmembers to be detected was $p = 31$ after calculating the virtual dimensionality (VD) of the hyperspectral data. As shown by Table 3.1, the ATDCA-GS extracted endmembers which were very similar, spectrally, to be known reference pixels in Fig. 3.2 (this method was able to perfectly detect the pixels labeled as 'A' and 'C', and had more difficulties in detecting very small targets). On the other hand, we could observe that the fractional abundance maps of vegetation, smoke and fire [see Fig. 4.6(a-c)] revealed features that cannot be easily appreciated in the false color composition displayed in Fig. 3.2. For illustrative purposes, the RMSE reconstruction map for this scene is also displayed in Fig. 4.6(d).

4.5.2 Analysis of parallel performance

The proposed full hyperspectral unmixing chain has been tested on four different platforms (two GPUs and two multi-core processors):

- The first GPU (denoted hereinafter as GPU1) is the NVidia Tesla C1060, which features 240 processor cores operating at 1.296 GHz, with single precision floating point performance of 933 Gflops, double precision floating point performance of 78 Gflops, total dedicated memory of 4 GB, 800 MHz memory (with 512-bit GDDR3 interface) and memory bandwidth of 102 GB/s[5].
- The second GPU (denoted hereinafter as GPU2) used is the NVidia GeForce GTX 580, which features 512 processor cores operating at 1.544 GHz, with single precision floating point performance of 1,354 Gflops, double precision floating point performance of 198 Gflops, total dedicated memory of 1,536 MB, 2,004 MHz memory (with 384-bit GDDR5 interface) and memory bandwidth of 192.4 GB/s[6].
- The two aforementioned GPUs are connected to a multi-core Intel i7 920 CPU (denoted hereinafter as MC1) at 2.67 GHz with 4 physical cores and 6 GB of DDR3 RAM memory, which uses a motherboard Asus P6T7 WS SuperComputer.
- Finally, another multi-core system (denoted hereinafter as MC2) is also used in the experiments. The system is made up of two Quad Core Intel Xeon at 2.53 GHz with 12 physical cores and 24 GB of DDR3 RAM memory, which uses a motherboard Supermicro X8DTT-H and it is mounted on a bullx R424-E2.

For illustrative purposes, Table 4.1 provides an indication of processor performance (including memory bandwidth and floating point performance) for both multi-core systems (MC1 and MC2) used in the experiments. These measures have been obtained using `Geekbench`[7] (version 2.4.0), a widely used benchmarking tool which provides a comprehensive set of benchmarks engineered to quickly and accurately measure processor and memory performance. Specifically, floating point performance has been evaluated using two `Geekbench` benchmarks based on the following calculations: vector dot product (single-core scalar, multi-core scalar, single-core vector and multi-core vector), and matrix LU decomposition (single-core scalar and multi-core scalar). On the other hand, memory bandwidth has been evaluated using two `Geekbench` benchmarks based on the following calculations: stream copy (single-core scalar and single-core vector) and stream add (single-core scalar and single-core vector).

Before describing the parallel performance results, it is important to emphasize that our GPU and multi-core versions provide exactly the same results as the serial versions of the implemented algorithms, using the `gcc` (gnu compiler default) with optimization flags `-O3` (for the single-core version) and `-fopenmp` (flag used for the multi-core version) to exploit data locality and avoid redundant computations. Hence, the only difference between the serial and parallel algorithms is the time they need to complete their calculations. The serial algorithms were executed in one of the available cores, while the multi-core versions were executed in all the available cores. For each experiment, ten runs were performed and the mean values were reported (these times were always very similar, with differences on the order of a few milliseconds only).

[5] http://www.nvidia.com/object/product_tesla_c1060_us.html
[6] http://www.nvidia.com/object/product-geforce-gtx-580-us.html
[7] http://www.primatelabs.com/geekbench/

Table 4.1: Processor performance (including memory bandwidth and floating point performance) for the two multi-core systems (MC1 and MC2) used in the experiments, measured using different Geekbench benchmarks.

Benchmark	**Description**	**MC1**	**MC2**
Vector dot product (measures floating point performance in Gflops)	single-core scalar	1.53	1.39
	multi-core scalar	7.24	16.4
	single-core vector	4.83	4.34
	multi-core vector	20.7	51.5
Matrix LU decomposition (measures floating point performance in Gflops)	single-core scalar	2.04	1.82
	multi-core scalar	4.41	11.3
Stream copy (measures memory bandwidth in GB/sec)	single-core scalar	5.96	5.42
	single-core vector	8.59	6.80
Stream add (measures memory bandwidth in GB/sec)	single-core scalar	7.85	6.49
	single-core vector	8.45	7.11

Table 4.2: Processing times (in seconds) and speedups achieved for the parallel unmixing chain in two different platforms: multi-core and GPU, tested with the AVIRIS Cuprite scene.

	Initialization	**VD**	**ATDCA-GS**	**UCLS**	**Writing**	**Total**
Serial time	0.121	5.541	1.331	1.051	0.009	8.053
Parallel time GPU1	0.269	0.246	0.049	0.067	0.009	0.640
Parallel time GPU2	0.281	0.241	0.024	0.034	0.011	0.590
Parallel time MC1	0.126	0.924	0.516	0.277	0.010	1.853
Parallel time MC2	0.098	1.066	1.055	0.197	0.053	2.468
Speedup (GPU1)	–	22.48	27.26	15.74	–	12.58
Speedup (GPU2)	–	23.00	55.28	30.62	–	13.64
Speedup (MC1)	–	6.00	2.58	3.79	–	4.35
Speedup (MC2)	–	5.20	1.26	5.35	–	3.26

Table 4.3: Processing times (in seconds) and speedups achieved for the parallel unmixing chain in two different platforms: multi-core and GPU, tested with the AVIRIS WTC scene.

	Initialization	**VD**	**ATDCA-GS**	**UCLS**	**Writing**	**Total**
Serial time	0.364	20.149	9.979	10.314	0.036	40.842
Parallel time GPU1	0.522	0.711	0.202	0.280	0.039	1.755
Parallel time GPU2	0.535	0.499	0.109	0.133	0.037	1.313
Parallel time MC1	0.370	3.258	3.549	2.759	0.037	9.973
Parallel time MC2	0.281	3.782	4.612	1.620	0.206	10.501
Speedup (GPU1)	–	28.34	49.28	36.79	–	23.28
Speedup (GPU2)	–	40.37	91.49	77.66	–	31.12
Speedup (MC1)	–	6.18	2.81	3.74	–	4.10
Speedup (MC2)	–	5.33	2.16	6.37	–	3.89

Tables 4.2 and 4.3 summarize the timing results and speedups measured after processing two hyperspectral images on the considered GPU and multi-core platforms. It should be noted that the cross-track line scan time in AVIRIS, a push-broom instrument [71], is quite fast (8.3 milliseconds to collect 512 full pixel vectors). This introduces the need to process the considered AVIRIS Cuprite scene (350×350 pixels and 188 spectral data channels) in less than 1.985 seconds to fully achieve real-time performance. Similarly, the AVIRIS WTC scene needs to be processed in less than 5.096 seconds in order

to achieve real-time performance. As shown by Table 4.2, the AVIRIS Cuprite scene could be processed in real-time using GPU1, GPU2 and MC1. On the other hand, Table 4.3 reveals that the AVIRIS WTC scene could only be processed in real-time in the GPU platforms.

For illustrative purposes, Fig. 4.7 shows the percentage of the total GPU execution time consumed by memory transfers and by each CUDA kernel (obtained after profiling the full implementation of the spectral unmixing chain) during the processing of the AVIRIS World Trade Center scene in the two considered GPUs. As shown by Fig. 4.7, the implementation on the GeForce GTX 580 GPU uses approximately 85% of the execution time for executing the kernels and 15% of the time for memory transfers. On the other hand, the implementation on the Tesla C1060 GPU uses about 90% of the total GPU time for executing the kernels and only 10% of the time for memory transfers. This indicates that memory transfers are not a bottleneck for the proposed GPU implementations.

On the other hand, Fig. 4.8 compares the processing times measured in the two considered multi-core processors when the number of physical cores available was varied. With the increase of the number of cores, the processing time in MC1 significantly decreases and leads to real-time processing results, as illustrated in Fig. 4.8(a). However, real-time processing performance could never be achieved in MC2. This is due to the scalability problems observed when increasing the number of cores. Specifically, it can be seen in Figs. 4.8(a) and 4.8(b) that there was no significant difference between using 4 or 12 cores in this platform. This suggests that our multi-core implementation can be optimized to increase its scalability to a high number of cores.

Although the speedups obtained in all cases for the multi-core platforms are not very high (particularly for MC2), we believe that this kind of systems may provide a good alternative to GPUs for onboard processing of remote sensing data. This is particularly due to the fact that the power consumption of GPUs is quite high, an observation that may compromise mission payload and energy requirements. In this regard, multi-core platforms are evolving very quickly and it is expected that systems with hundreds of cores will be soon available, thus offering the possibility to replace many-core systems such as GPUs as the default platforms for high performance computing in many applications. At present, GPUs offer such possibility of massively parallel processing and we have illustrated in this work that their computational power can be readily exploited for providing real-time performance in an important exploitation-based application for hyperspectral data such as spectral unmixing.

4.6 Summary and future research lines

In this chapter, computationally efficient implementations of a full hyperspectral unmixing chain have been developed on multi-core processors and GPU platforms. Both platforms can boost the computational performance of the considered unmixing chain, using relatively inexpensive hardware. The performance of the proposed implementations has been evaluated (in terms of the quality of the solutions provided and their parallel performance) in the context of two analysis scenarios, using data sets collected by the AVIRIS instrument. The experimental results reported in this chapter indicate that remotely sensed hyperspectral imaging can greatly benefit from the development of efficient implementations of unmixing algorithms in specialized hardware devices for better exploitation of high-dimensional data sets. In this case, real-time performance could be obtained using any of the considered GPU devices and one of the considered multi-core environments. GPUs compared with multi-core processors present more advantages because these devices offer few onboard restrictions in terms of cost and size, and these are important parameters when defining mission payload in remote sensing missions. In this regard, our contribution bridges the gap towards real-time unmixing of remotely sensed hyperspectral images in

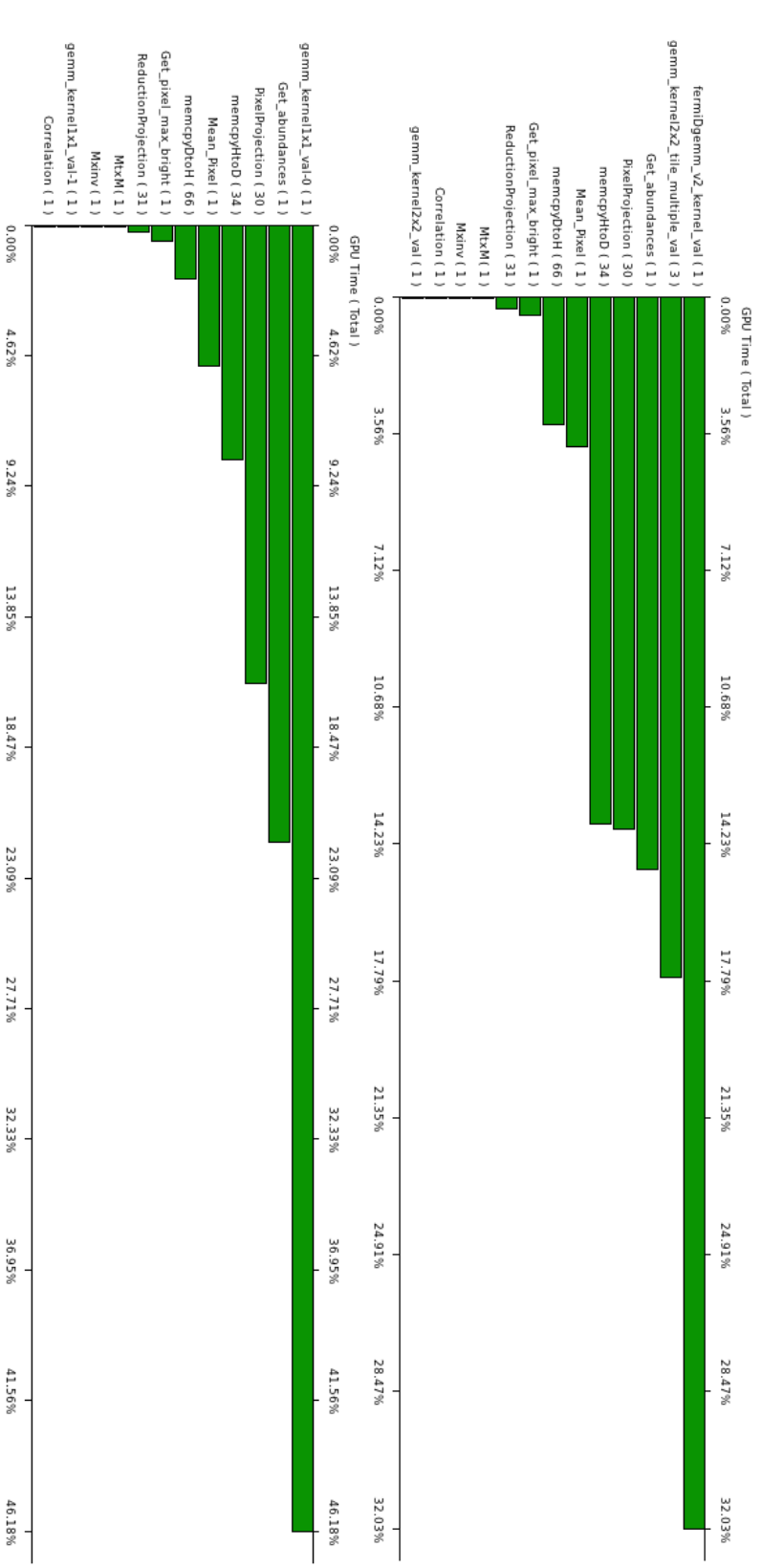


Figure 4.7: Summary plot describing the percentage of the total GPU time consumed by memory transfer operations and by the different kernels used by the NVidia GeForce GTX 580 GPU (top) and the Tesla C1060 GPU (bottom) in the unmixing of the AVIRIS WTC image.

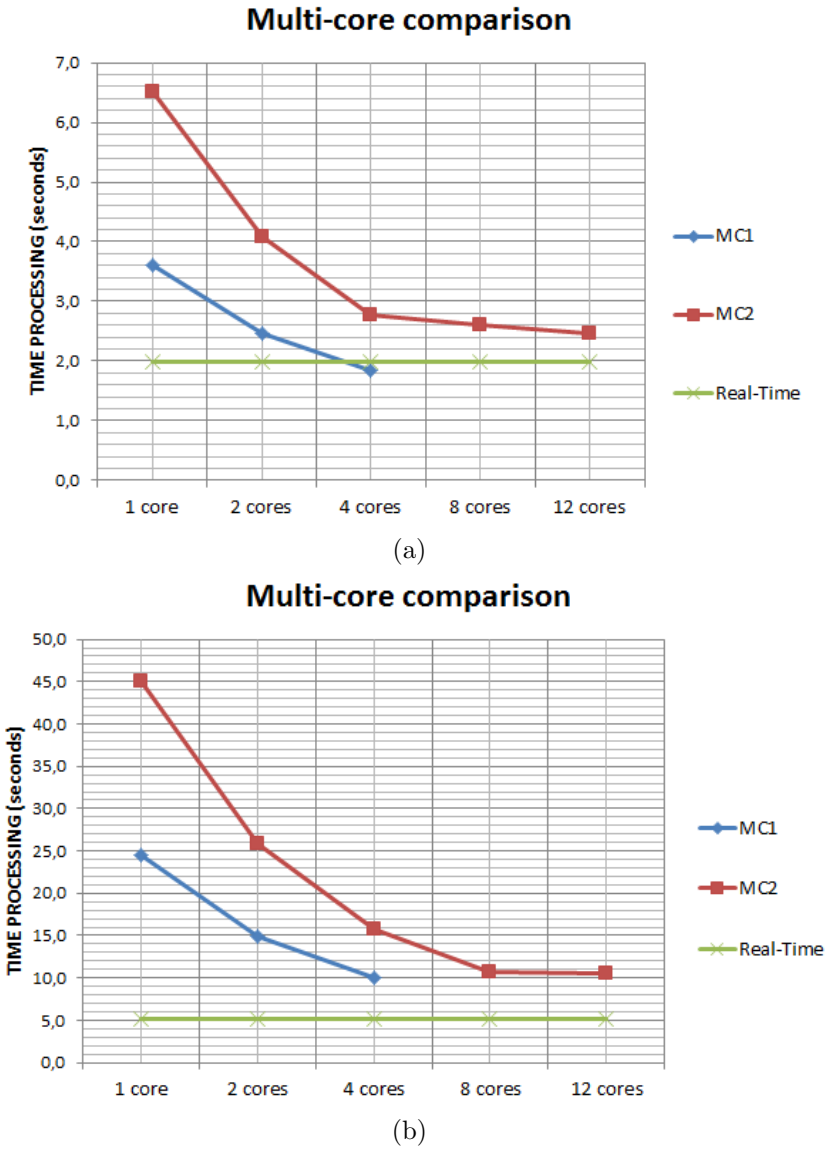


Figure 4.8: Comparison of time processing between the two considered multi-core processors varying the number of cores using: (a) AVIRIS Cuprite scene. (b) AVIRIS WTC scene.

GPUs and multi-core processors.

Although the results reported in this chapter are very encouraging, GPUs and multi-core processors are still rarely exploited in real missions due to power consumption and radiation tolerance issues, despite improvements in these directions are expected in upcoming years. Currently we are also experimenting with FPGAs and other spectral unmixing algorithms, i.e., a fully constrained linear spectral unmixing algorithm, in order to be able to adapt the proposed algorithms to hardware devices that can be mounted onboard hyperspectral imaging instruments after space qualification by international agencies. We are also investigating the use of OpenCL[8] as a computing standard for multi-core architectures that allows writing code for both GPUs and CPUs, and which has recently also seen a growing interest for FPGAs.

[8]http://www.khronos.org/opencl

Chapter 5

Systems for Information Extraction and Classification of Remotely Sensed Imagery

5.1 A new parallel tool for classification of remotely sensed imagery

Outline - In this subchapter, a new tool for classification of remotely sensed images is described. Our processing chain is based on three main parts: (1) pre-processing, performed using morphological profiles which model both the spatial (high resolution) and the spectral (color) information available from the scenes; (2) classification, which can be performed in unsupervised fashion using two well-known clustering techniques (ISODATA and k-means) or in supervised fashion, using a maximum likelihood classifier; and (3) post-processing, using a spatial-based technique based on a moving a window which defines a neighborhood around each pixel which is used to refine the initial classification by majority voting, taking in mind the spatial context around the classified pixel. The processing chain has been integrated into a desktop application which allows processing of satellite images available from Google Maps engine and developed using Java and the SwingX-WS library. A general framework for parallel implementation of the processing chain has also been developed and specifically tested on graphics processing units (GPUs), achieving speedups in the order of 30× regard to the serial version of same chain implemented in C language.

5.1.1 Overview

The wealth of satellite imagery available from many different sources has opened the appealing perspective of performing remote sensing image classification and retrieval tasks [3, 4] at a large scale. For instance, Google Maps engine[1] now provides high-resolution satellite images from many locations

Part of this subchapter has been published in::

S. Bernabé, A. Plaza, P. R. Marpu and J. A. Benediktsson, "A New Parallel Tool for Classification of Remotely Sensed Imagery", *Computers & Geosciences*, vol. 46, pp. 208–218, September 2012 [JCR(2012)=1.834].

[1] http://maps.google.com

around the Earth, and there have been significant efforts to develop an application programming interface (API)[2] and other external libraries such as SwingX-WS[3] to facilitate the processing of remotely sensed data available from Google Maps engine. The combination of an interactive mapping and satellite imagery tool such as Google Maps with advanced image classification and retrieval features [50, 99, 100] has the potential to significantly expand the functionalities of the tool and also to allow end-users to extract relevant information from a massive and widely available database of satellite images (the Google Maps service is free for non-commercial use)[4]. It should be noted that the current version of Google Maps does not allow using maps outside a web-based application (except with a link to Google Maps). Here we use Google Maps purely as example to demonstrate that if we have a data repository we can use the tool we propose. We are currently exploring the possibility of large scale processing with data such as from Google Maps repositories. By using the Google Maps API or external libraries such as SwingX-WS, it is possible to embed the full Google Maps site into an external website application. Other similar services currently available comprise Yahoo Maps[5], Microsoft Bing Maps[6] and OpenStreetMap[7]. The characteristics of Yahoo Maps and Microsoft Bing Maps are similar to those available in Google Maps (although the spatial resolution of the satellite imagery available in Google Maps is generally higher than the resolution of the image data available in those systems). On the other hand, the OpenStreetMap follows a different approach. It is a collaborative project aimed at creating a free editable map of the world. Its design was inspired by sites such as Wikipedia[8]. In comparison, the Google Maps service offers important competitive advantages [101], such as the availability of high resolution satellite imagery, the smoothness in the navigation and interaction with the system, and adaptivity for general-purpose desktop applications. Regardless of its competitive advantages, the possibility to perform unsupervised or supervised classification of satellite images [7, 102, 103] is not available in Google Maps. However, image classification is widely recognized as one of the most powerful approaches in order to extract information from satellite imagery [6, 36, 104, 105].

In this subchapter, we describe a new tool for classification of remotely sensed imagery which has been tested with different types of data, including satellite images from the Google Maps engine and also remotely sensed hyperspectral images. Our processing chain is based on three main parts: (1) pre-processing, performed using morphological profiles [36, 99] which model both the spatial (high resolution) and the spectral (color) information available from the scenes using mathematical morphology concepts [5, 6]; (2) classification, which can be performed in unsupervised fashion using two well-known clustering techniques: ISODATA [106] and k-means [107] or in supervised fashion, using a maximum likelihood classifier [108]; and (3) post-processing, using a simple spatial-based technique based on majority voting which adopts a similar strategy followed by techniques such as Markov random fields [109]. The processing chain has been implemented in the C programming language and integrated into our proposed tool, developed using Java and the SwingX-WS library.

In order to cope with the computational requirements introduced by the processing of large areas at high spatial resolution [57], a parallel framework for the processing chain is provided [24,27] with a specific implementation for commodity graphic processing units (GPUs). These specialized hardware cards are nowadays widely available due to the advent of video-game industry, and can now be programmed in

[2] http://code.google.com/apis/maps/index.html
[3] https://swingx-ws.dev.java.net
[4] http://code.google.com/apis/maps/terms.html
[5] http://maps.yahoo.com
[6] http://www.bing.com/maps
[7] http://www.openstreetmap.org
[8] http://www.wikipedia.org

general purpose fashion [33, 58, 96]. The accuracy of our proposed parallel processing chain is validated by analyzing the consensus of the classification results obtained with regard to those provided by other implementations of the considered unsupervised and supervised classifiers in commercial software (ITT Visual Information Solutions ENVI[9]).

The remainder of the subchapter is organized as follows. Section 5.1.2 briefly describes the programming libraries used to develop the proposed system and the procedure for image retrieval from Google MapsTM engine. Section 5.1.3 describes the processing chain used for processing satellite images. Section 5.1.4 describes a parallel implementation of the considered processing chain for GPUs. Section 5.1.5 describes an experimental evaluation of the proposed system which has been conducted by comparing the agreement between the obtained classification results with those provided by commercial software, such as ITT Visual Information Solutions ENVI package. It also includes an application case study in which the proposed system is used for the classification of hyperspectral data collected over the University of Pavia, Italy, using a data set with reference information about different urban classes. Finally, section 5.1.6 concludes the subchapter with some remarks and hints at plausible future research.

5.1.2 Libraries

In order to develop our system, we resorted to the `SwingX-WS` library which provides advanced capabilities for the creation of desktop applications. `SwingX-WS` attempts to simplify the use of web services (in the broad sense) by providing APIs that reside on top of existing libraries (such as Google Maps API). It contains a set of Java beans (i.e., reusable software components developed in Java) which are able to interact with web services and, at the same time, can be embedded into standard desktop applications (which fits very well our desired functionality). The initial Java beans included in `SwingX-WS` library provide support for advanced Google web services such as searching news, video, images, and financial data, as well as a generic tile-based mapping component that was adopted as a baseline for the design of our desktop application. Specifically, our `SwingX-WS` beans have been designed with graphical configuration in mind and work quite well inside of a Java beans-aware editor such as Java NetBeans[10]. The bean that we particularly exploited in the development of our desktop application is `JXMapViewer`, a generic viewer for tile-based map servers. In our proposed system, a client (user) first sends a request to the web server via `SwingX-WS`. The web server interacts with an application server that provides the map server functionality. Specifically, the application server interacts directly with a database from which the extracted information is provided. Finally, the newly developed desktop application performs data processing using a specific chain that will be described in the next section, using the `JXMapViewer` component and the other additional `SwingX-WS` and `SwingX` modules to manage the visualization of the obtained classification results.

In addition to `JXMapViewer`, other additional `SwingX-WS` and `SwingX` modules were used in the development of our desktop application. In the following, a brief overview of these additional modules is provided:

- **`JXMapKit`**. The `JXMapKit` module comprises a pair of `JXMapViewers` modules which allow including some desired functionalities in the newly developed desktop application, including zoom buttons, a zoom slider, and a mini-map in the lower right corner showing an overview of the map.
- **`TileFactoryInfo`**. The `TileFactoryInfo` is in charge of encapsulating all information specific to the managing of the images provided by the map server in the form of tiles. This includes the

[9] http://www.ittvis.com/language/en-us/productsservices/envi.aspx
[10] http://netbeans.org

functionality that allows the desktop application to load the map tiles from a certain location, and also to control the size and zoom level of the tiles. Any map server can be used by installing a customized `TileFactoryInfo`, but in our specific desktop application we only use this module to manage tiles derived from Google Maps server.

- **`DefaultTileFactory`**. Creates a new instance of `DefaultTileFactory` using the specified `TileFactoryInfo`. This module has also been used to form wide satellite images by composing a mosaic of tiles provided by means of `TileFactoryInfo`, thus incorporating the functionality to download large image areas at the desired zoom level from Google Maps engine to the newly developed desktop application.

- **`GeoPosition`**. This module provides the geographical latitude and longitude associated to each pixel of the image tiles returned by `DefaultTileFactory`, and further allows locating the image tile which comprises a certain geographic location.

- **`Painter`**. This module allows software developers to be able to customize the background painting of a `JXPanel`, which has been used in our desktop application to visualize the images provided by Google Maps engine. Since many components within `SwingX` extend the `JXPanel` module, the developer can implement custom painting. In our developments, this module has been specifically used to superimpose the classification result provided by the different considered algorithms on the maps provided by Google Maps engine, including additional functionalities such as controlling the degree of transparency when carrying out such superimposition.

- **`CompoundPainter`**. `Painters` can be combined together by using a component called `CompoundPainter`, which uses an array to store several `Painters`, and the order in which they should be painted. This functionality has been used in the newly developed desktop application to establish several layers by means of which the obtained classification results can be superimposed on the satellite images provided by Google Maps with different levels of transparency.

5.1.3 Processing chain

Fig. 5.1 shows a flowchart of the processing chain considered here. In the following, we describe the different parts of the processing chain.

5.1.3.1 Pre-processing using morphological profiles

One of the most widely used methods in the literature for analyzing spatial structure in image scenes is mathematical morphology [5]. It is based on processing the original image using a so-called structuring element (SE), which acts as a probe for extracting or suppressing specific structures of the image objects, checking that each position of the SE fits within those objects. Based on these ideas, two fundamental operators are defined in mathematical morphology, namely *erosion* and *dilation* [6]. The erosion operation provides an output image which shows where the SE fits the objects. On the other hand, the application of the dilation operator to an image produces an output image, which shows where the SE hits the objects in the image. All other mathematical morphology operations can be expressed in terms of erosion and dilation. For instance, the notion behind the *opening* operator is to dilate an eroded image in order to recover as much as possible of the eroded image. In contrast, the *closing* operator erodes a dilated image so as to recover the initial shape of image structures that have been dilated.

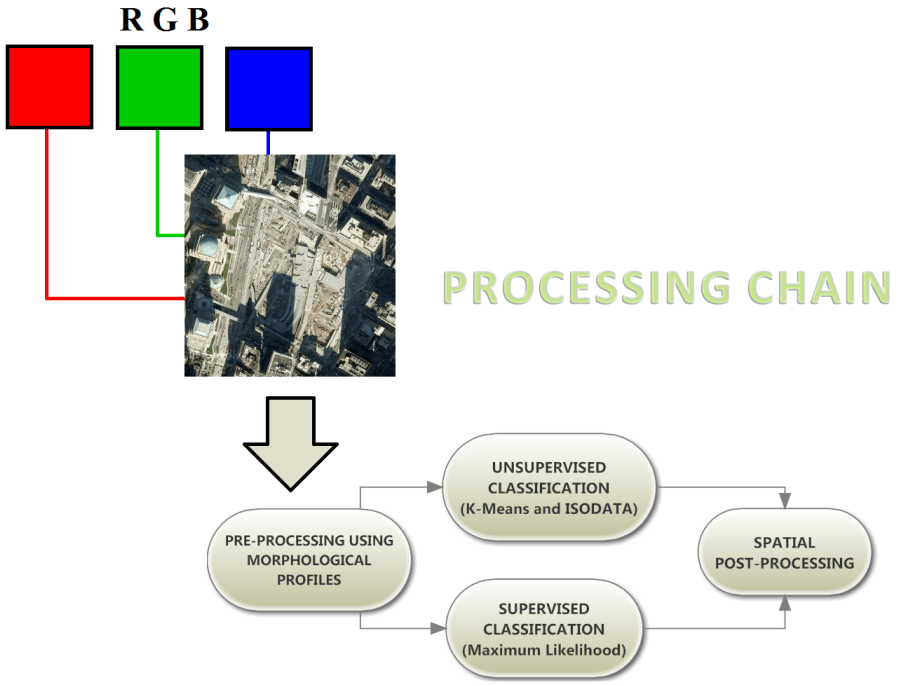


Figure 5.1: Flowchart describing the processing chain considered in this subchapter.

The concept of morphological profile (MP) was originally developed [99] from the principle of granulometry [6], and relies on opening and closing by reconstruction operations, a special class of morphological transformations that do not introduce discontinuities and which preserve the shapes observed in input images. While conventional opening and closing remove the parts of the objects that are smaller than the SE, opening and closing by reconstruction either completely removes the features or retains them as a whole. The MP is composed of the *opening profile* (OP), which consists of an ensemble of opening by reconstruction of increasing size, and of the *closing profile* (CP), performed using the dual operation [6]. For a given SE, geodesic opening and closing allows one to know the size or shape of the objects present in the image: those which are deleted are smaller than the SE, while those which are preserved are bigger. To determine the shape or size of all elements present in an image, it is necessary to use a range of different SE sizes. This assumption leads to the formal definition of the MP:

$$\mathrm{MP}(x,y) = \{\mathrm{CP}_k(x,y), \ldots, f(x,y), \ldots, \mathrm{OP}_k(x,y)\}\,, \tag{5.1}$$

where $f(x,y)$ denotes the pixel in the spatial coordinates (x,y) of the original image. However, the above formulation refers to a single channel image and, therefore, the color information available in Google Maps satellite images is not considered. A simple approach to deal with this problem is to build the MP on each of the individual data channels. This approach, called *extended morphological profile* (EMP), can be seen as a single stacked vector to be used as a collection of features for classification purposes. Following the previous notation used in Eq. (5.1), the EMP at the pixel with spatial location (x,y) can be simply represented by:

$$\mathrm{MP}_{ext}(x,y) = \{\mathrm{MP}_R(x,y), \mathrm{MP}_G(x,y), \mathrm{MP}_B(x,y)\}\,, \tag{5.2}$$

where MP_R, MP_G and MP_B respectively denote the MP calculated on the red, green and blue channel. The resulting EMP can be used as an enhanced feature vector which extends over the information contained in the original pixel (using spatial and spectral information) for subsequent classification purposes.

5.1.3.2 Unsupervised and supervised classification

Unsupervised classification is conducted using clustering techniques, which aim at grouping pixels together in feature space so that pixels belonging to the same cluster present similar properties [100]. The first unsupervised clustering technique used in the implementation is the well-known ISODATA algorithm [106], a squared-error clustering method. The algorithm starts with a random initial partition of the available pixels in the original image f into c candidate clusters. It then iteratively optimizes this initial partition so that, on each iteration i, a partition $\mathrm{P}^i = \{\mathrm{P}^i_1, \mathrm{P}^i_2, \cdots, \mathrm{P}^i_c\}$ of the image f into c clusters is computed, where $\mathrm{P}^i_k = \{f^i_{j,k} \in \Re^n, j = 1, 2, \cdots, m^i_k\}$ contains the set of pixels in f belonging to the k-th component on the iteration i, with m^i_k denoting the number of pixels in P^i_k and $k = 1, 2, \cdots, c$. For clarity, the spatial coordinates (x, y) associated to the pixels in the original image f have been removed from the formulation above since the clustering is conducted in the spectral (feature) space, and not in the spatial domain as it was the case with morphological operations. With this notation in mind, in which the spatial coordinates associated to the pixels have been omitted for simplicity, the squared error for a given partition P^i of the original image f into c clusters is defined as:

$$e^2(\mathrm{P}^i) = \sum_{k=1}^{c} \sum_{j=1}^{m^i_k} \| f_{j,k} - \mu_k \|, \tag{5.3}$$

where μ_k is the centroid of the k-th cluster. The squared error in Eq. (5.3) is reduced at each iteration, until a convergence criterion is achieved.

Another clustering algorithm has been considered in the implementation: the well-known k-means algorithm [107]. Its concept is similar to that of ISODATA. Specifically, the goal of k-means is to determine a set of c points, called centers, so as to minimize the mean squared distance from each pixel in f (or in MP_{ext}) to its nearest center. The algorithm is based on the observation that the optimal placement of a center is at the centroid μ_k of the k-th cluster. It starts with a random initial placement. At each stage, the algorithm moves every center point to the centroid of the set of pixel vectors for which the center is a nearest neighbor, according to the Euclidean distance [100], and then updates the neighborhood by recomputing the distance from each pixel vector to its nearest center. These steps are repeated until the algorithm converges to a point that is a minimum for the distortion [107]. A relevant issue for both the ISODATA and the k-means algorithms is how to set the number of clusters c in advance. In our work, this choice is left to the end-user, who can adjust the quality of the clustering by interactively setting this parameter.

Once a segmentation [110] of the original image f (or its EMP, denoted by MP_{ext}) has been achieved via unsupervised clustering, a supervised procedure can be applied to classify other different areas based on the training sites selected in a different spatial location. In our tool, this can be accomplished using the well-known maximum likelihood (ML) classifier [100], which relies on the following discriminant function:

$$\mathrm{ML}(x, y) = -\ln|\gamma_i| - (f(x, y) - \mu_i)^T \gamma_i (f(x, y) - \mu_i), i = 1, \cdots, c, \tag{5.4}$$

where $f(x, y)$ is the pixel vector with spatial coordinates (x, y), μ_i is the mean vector for class i, with $i = 1, \cdots, c$, and γ_i is the covariance matrix of class i. Alternatively, in (5.4) the pixel, $f(x, y)$, can be replaced by its associated profile, $\mathrm{MP}_{ext}(x, y)$ if morphological pre-processing is performed to the original image f prior to classification purposes.

5.1.3.3 Spatial post-processing

A spatial post-processing module [105] has been implemented in our chain in order to refine the outcome of the classification conducted in the previous subsection by simply sliding a square neighborhood window centered in each classified pixel and applying a majority voting procedure in which the central pixel is assigned to the most predominant class in the neighborhood window. In our tool, this simple spatial post-processing step is completely optional. The size of the spatial windows considered in the implementation of spatial post-processing can be completely configured by the end-user, where typical window sizes range from 3×3 to 15×15 pixels.

5.1.4 Parallel implementation

A general parallel framework for the considered processing chain can be developed by means of a standard data partitioning strategy, in which the original image is decomposed into different partitions according to a spatial-domain decomposition framework, meaning that each pixel (vector) is never partitioned across different processing units of the parallel system [24, 57]. In the following we describe the specific parallel implementation conducted in this subchapter, which have been carried out for GPUs. These systems 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, taking one or more streams as inputs and producing one or more streams as outputs [62]. Thereby, data-level parallelism is exposed to hardware, and kernels can be concurrently applied without any sort of synchronization. The kernels can perform a kind of batch processing arranged in the form of a grid of blocks, as displayed in Fig. 5.2(a), where each block is composed by a group of threads which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. This figure also displays how each kernel is executed as a grid of blocks of threads. On the other hand, Fig. 5.2(b) shows the execution model in the GPU, which can be seen as a set of multiprocessors. In each clock cycle each processor of the multiprocessor executes the same instruction but operating on multiple data streams. Each processor has access to a local shared memory and also to local cache memories in the multiprocessor, while the multiprocessors have access to the global GPU (device) memory.

The steps of the processing chain in Fig. 5.1 that were implemented in the GPU comprise the pre-processing of the original image using morphological profiles, the unsupervised classification using the k-means clustering algorithm, and the final spatial post-processing. The parallel algorithms were implemented using NVidia CUDA.

5.1.4.1 GPU implementation of pre-processing using morphological profiles

In order to implement the morphological pre-processing step in parallel, the first issue that needs to be addressed is how to map the original image onto the memory of the GPU. In the implementation, we assume that the size of the image f to be processed fits into the GPU memory. Otherwise, the image is partitioned into multiple spatial-domain tiles, with scratch borders to avoid communicating pixels in the border of partitions. In this case, the parallelization strategy can be simply addressed as

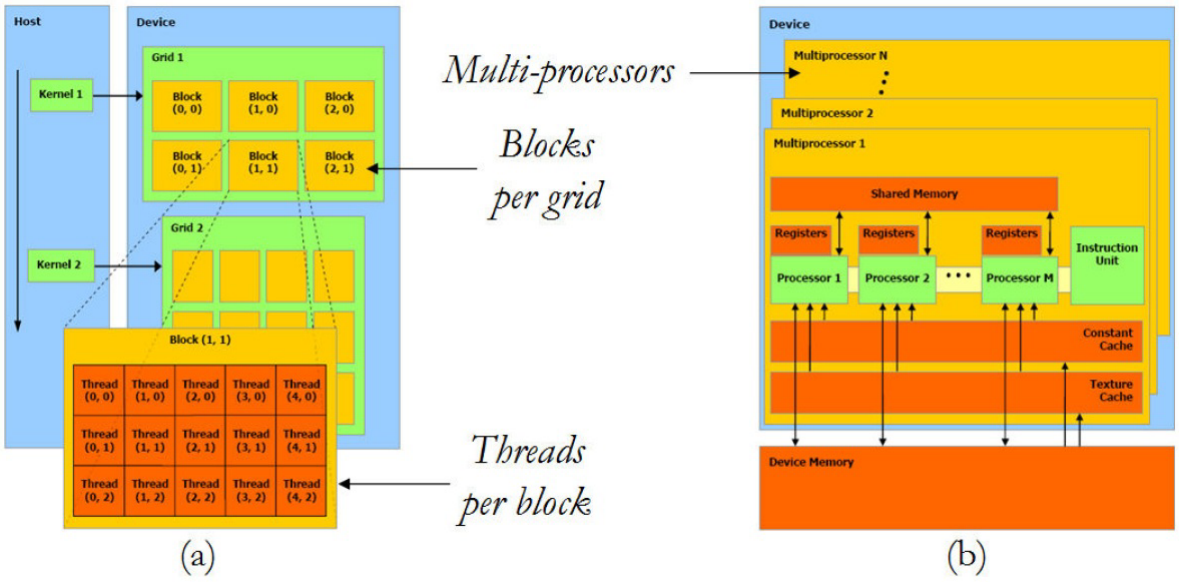


Figure 5.2: Schematic overview of a GPU architecture. (a) Threads, blocks and grids. (b) Execution model in the GPU.

described in [24], in which a step-by-step description on how to implement a border-handling strategy with redundant information is given. We now describe the kernels used for efficient implementation of morphological pre-processing on the GPU:

- *Erosion and dilation.* This kernel calculates the morphological erosion and dilation operations by defining a grid with a number of blocks equal to the number of pixels in the original image f, and with a number of threads defined by the number of pixels in the window defined by the morphological SE and centered around each pixel $f(x, y)$. This situation is graphically illustrated by an example in Fig. 5.3, in which NM $\times$ NL denotes the total number of pixels and TV denotes the number of threads allocated to each processing window. Internally, the threads of each block calculate the minimum or the maximum value of the pixels in the window to complete the morphological dilation and erosion operation, respectively, and the kernel is applied to the three components of the original image f: red, green and blue.

- *Opening and closing by reconstruction.* This kernel builds opening and closing by reconstruction operations by combining the output of erosion and dilation to form standard opening and closing filters, and then iterates k times in order to obtain the opening and closing by reconstruction operations OP_k and CL_k. A similar structure of grids, blocks and threads with regards to the one used in Fig. 5.3 is adopted.

- *Extended morphological profile.* This kernel simply combines the output provided by the previous stage and the original values in the three channels of the original image f to obtain the morphological profile MP and the extended profile MP_{ext} at each pixel. This is done by combining the output provided by opening and closing by reconstruction for each of the individual channels. As shown by the description of the kernels in this section, the calculation of morphological profiles is an embarrassingly parallel calculation since the calculations for each processing window can be completed independently of those associated to the other windows. This kind of parallelism guided

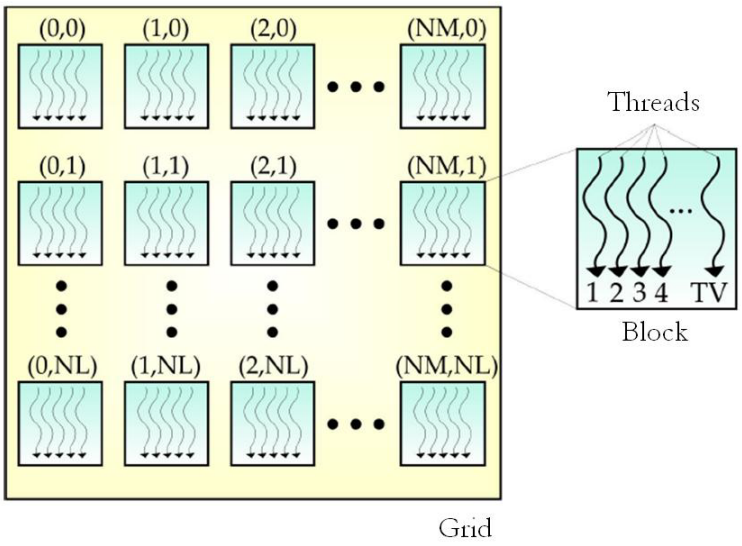


Figure 5.3: Parallel strategy adopted in the GPU implementation of morphological profiles: a grid is defined with as many blocks as pixels in the original image, and each block manages a number of threads equal to the number of individual operations which are performed within a processing window. $\mathrm{NM} \times \mathrm{NL}$ denotes the total number of pixels and TV denotes the number of threads allocated to each window.

by data and without inter-processor communications fits perfectly the GPU architecture depicted in Fig. 5.2.

5.1.4.2 GPU implementation of the k-means unsupervised clustering algorithm

The k-means algorithm calculates the Euclidean distance from each pixel vector (which can be the original information in the image f or the outcome of the morphological pre-processing stage, MP_{ext}) to the closest cluster center (where the cluster centers are initialized randomly). This operation is computed in a single CUDA kernel, in which the calculation of the distance of each pixel of the image was assigned to an independent processing thread. As a result, in this implementation we define a grid with as many blocks as pixels are present in the original image f. Internally, the threads of each block calculate the Euclidean distance between the local pixel and each of the c classes, where the value of c is determined in advance. In this way, each thread calculates the distance between the values of each pixel and the nearest cluster center.

It should be noted that, in this kernel, the processing of each pixel is performed in a vector-based fashion (i.e., the distance is calculated by considering all the components allocated to each pixel) as opposed to the processing in the previous stage (morphological pre-processing) in which the components of the original image are processed in channel-by-channel fashion. For illustrative purposes, Fig. 5.4 shows a simplified version of the CUDA kernel that performs this operation. The main purpose with Fig. 5.4 is to provide an idea of the complexity of writing a kernel in CUDA by illustrating one of the kernels used in our implementation. As shown by Fig. 5.4, porting an available C code into CUDA is relatively simple as the code is similar to a standard C implementation but encapsulated in the form of a kernel. Once this operation is completed, the centers of each cluster are recomputed and new pixels reassigned to each of the clusters until convergence. This part has not been implemented in the GPU, mainly because, for a small number of clusters, this computation can be performed in the CPU without

```
__constant__ centroid constData[SIZE_DATA];

__global__ void KMeans_kernel(pixel *g_idata, centroid *g_centroids, int numClusters,
                              unsigned long numElements) {
    unsigned long valindex = blockIdx.x*blockDim.x + threadIdx.x; //id. thread
    int k, myCentroid;
    unsigned long minDistance=0xFFFFFFFF;

    for(k=0;k<numClusters;k++){
        if(abs((long)(g_idata[valindex].value - constData[k].value))<minDistance){
            minDistance=abs((long)(g_idata[valindex].value - constData[k].value));
            myCentroid=k;
        }
        g_idata[valindex].centroid=myCentroid;
        __syncthreads();
    }
}
```

Figure 5.4: Simplified version of the CUDA kernel developed for the implementation of the k-means algorithm in GPUs.

representing a dominant factor in the overall time of the solution.

5.1.4.3 GPU implementation of the spatial post-processing module

Finally, the spatial post-processing module can be simply implemented in the GPU following a similar strategy to the one adopted for the implementation of morphological operations. The CUDA kernel which implements this operation simply defines a grid with a number of blocks equal to the number of pixels in the original image f, and with a number of threads defined by the number of pixels in the post-processing window centered around each pixel $f(x, y)$, as illustrated in Fig. 5.3.

We emphasize that the GPU implementation only covers a part of the considered processing chain, as the ISODATA algorithm for unsupervised classification and the ML algorithm for supervised classification have not been implemented in the GPU as of yet. However, with the GPU version of k-means and the pre- and post-processing modules described in this section, a partial GPU implementation of the processing chain is available and will be evaluated in terms of parallel performance in section 5.1.5.

To conclude this section, Fig. 5.5 shows different views of the developed tool. The tool allows selecting an area to be classified, obtaining classification results both in unsupervised and supervised fashion, retaining the classified area at different zoom levels, and other functionalities such as spatial pre- and post-processing, managing of the resulting classification and extracted satellite images, loading/storing of results via file logs which can be saved in a database, automatic positioning in any latitude and longitude coordinates in the entire Google Maps database, overlaying of classification results with different views (satellite, map, hybrid), etc.

5.1.5 Experimental validation

In this section, an experimental validation of the developed system is performed using satellite images obtained from Google Maps across different locations. The validation is conducted by means of the following experiments:

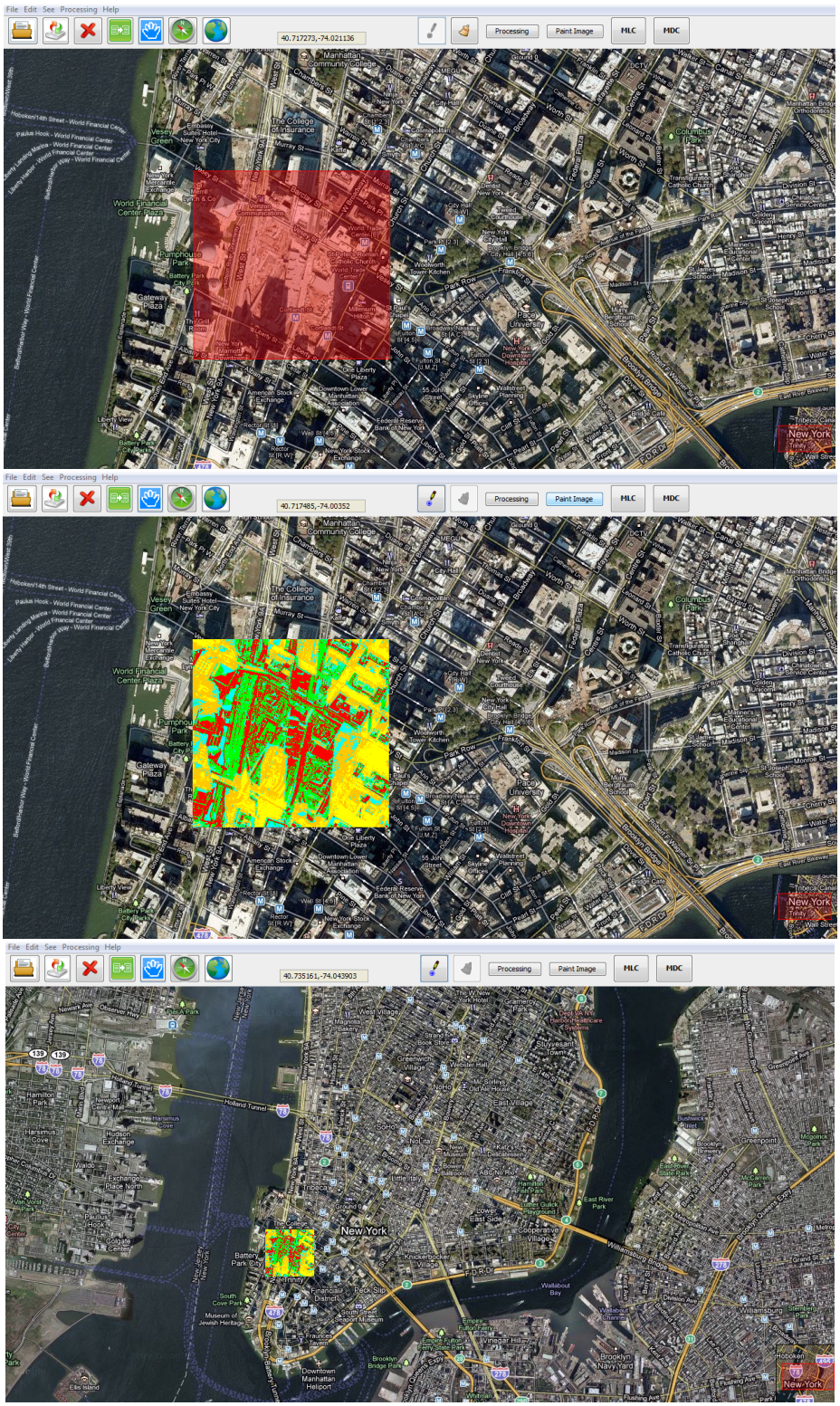


Figure 5.5: Different views of the developed tool. Selection of an area to be classified in New York City (top). Unsupervised classification result provided by our parallel implementation of k-means algorithm superimposed on the tool (middle). Zoom reduction retaining the classified area (bottom).

1. In the first experiment, the impact of using morphological profiles versus using the original satellite image is evaluated in the considered processing chain. This experiment is based on visual assessment of the classification obtained for a satellite image collected over the Nile river in Egypt (which presents spatial correlation between the observed image features and hence can be used to analyze the impact of morphological pre-processing).

2. In the second experiment, an experimental validation of the k-means and ISODATA unsupervised classification algorithms is conducted by selecting a satellite image over the World Trade Center in New York city (an area which represents a challenging classification scenario due to the presence of complex urban features). The validation has been conducted by evaluating the agreement between the classification results provided by our implemented versions of k-means and ISODATA with regards to those available in the well-known ENVI commercial software package distributed by ITT Visual Information Solutions. In our tests, exactly the same parameters are adopted when running the implementations as those available in the ENVI package.

3. In the third experiment a detailed validation of the supervised ML classification algorithm is conducted using the same image in the previous experiment. Again, the obtained results are validated by the ML classifier by evaluating the agreement with the implementation available in ENVI using different satellite images collected over the World Trade Center area in New York City. Again, exactly the same parameters are used when running the proposed implementation and those available in ENVI.

4. In the fourth experiment, the computational performance of the developed GPU implementation of the processing chain (which comprises morphological pre-processing, k-means clustering and spatial post-processing) is analyzed with regard to its CPU (serial) version. For this purpose, two GPUs from NVidia are used: the GeForce 9400M GPU[11] and the Tesla C1060 GPU[12].

5. In the fifth experiment the performance of the proposed processing chain is analyzed in the context of an urban classification application using a remotely sensed hyperspectral data set collected by the reflective optics spectrographic imaging system (ROSIS). The flight was operated by the Deutschen Zentrum for Luft und Raumfahrt (DLR, the German Aerospace Agency) in the framework of the HySens project, managed and sponsored by the European Union. The considered scene was collected over the University of Pavia in Italy and comprises reference information about urban classes that will be used to assess the performance of the proposed system with a different type of remotely sensed data, thus showing the generality of the developed tool.

5.1.5.1 Experiment 1: impact of using morphological profiles

For this first experiment, a satellite image collected over a stretch of the Nile river in Cairo, Egypt has been selected. The spatial resolution of the image is approximately 10 meters per pixel. Fig. 5.6(a) shows the original image, while Figs. 5.6(b) and 5.6(c) respectively show the result of applying the proposed processing chain with and without using morphological profiles for pre-processing (using $k = 14$). In both cases, the k-means algorithm (implemented with $c = 5$) is used to provide the final classification maps, and no spatial post-processing is performed. Finally, Fig. 5.6(d) shows the classification result obtained by the k-means algorithm available in ENVI software for the original image, using also $c = 5$. It

[11] http://www.nvidia.com/object/product_geforce_9400m_g_us.html
[12] http://www.nvidia.com/object/product_tesla_c1060_us.html

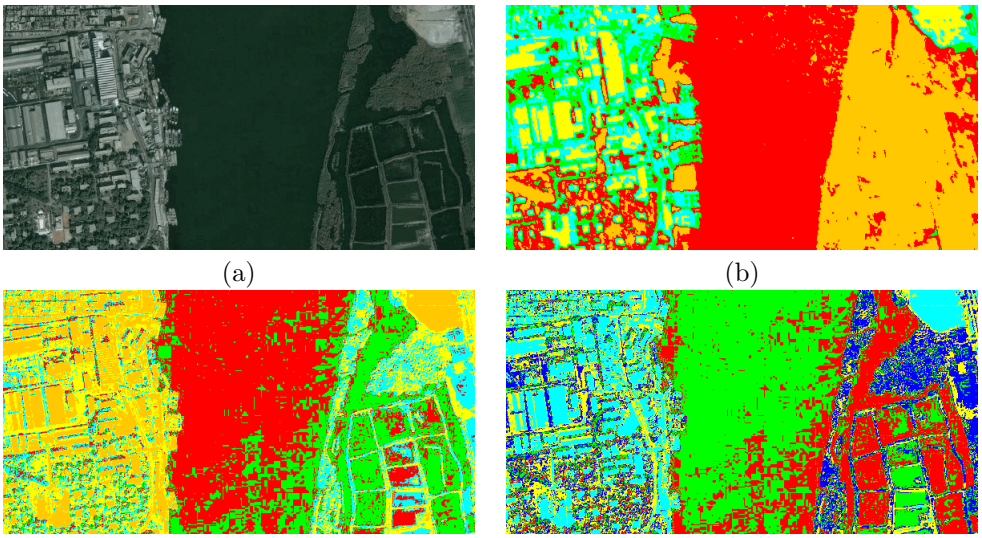


Figure 5.6: (a) Satellite image collected over a stretch of the Nile river in Cairo, Egypt. (b) Classification using the processing chain with morphological profiles and k-means. (c) Classification using the processing chain (on the original image) with k-means. (d) Classification using ENVI's implementation of k-means.

should be noted that the results displayed in Figs. 5.6(b-c) were obtained using our GPU implementation of the processing chain (the results of the GPU version are exactly the same as those obtained for the serial version implemented in C language). As shown in Fig. 5.6, the classification output obtained after applying morphological profiles [see Fig. 5.6(b)] provides a better characterization of spatial features such as the river or the urban areas located at the leftmost bank of the river than the classification output obtained after applying the k-means algorithm to the original satellite image [see Fig. 5.6(c)], which exhibits outliers in homogeneous classes such as the river. This example reveals that the rich spatial information (extracted through morphological profiles) can be used in combination with spectral information to discriminate objects better than using spectral information alone.

As shown by Fig. 5.6, the color labels obtained in the different classification results are different, but the classification maps provided by the processing chain applied to the original satellite image in Fig. 5.6(c) and by ENVI's implementation of k-means in Fig. 5.6(d) are very similar. In both cases, the parameters for both algorithms have been set to exactly the same values, with the number of clusters set empirically to $c = 5$. Table 5.1 reports the classification agreement (in percentage) measured after comparing our processing chain result, based on k-means classification, with the one obtained by ENVI (assuming the latter as the reference). In other words, the agreement reflects the pixel-based similarity of the map produced using our implementation with regards to the map obtained using ENVI, which is considered as reference data in this experiment. As shown by Table 5.1, the agreement between both maps (in terms of individual classes and also from a global point of view) is very high. This is also confirmed by the confusion matrix displayed in Table 5.2 [111]. Overall, this experiment reveals that morphological pre-processing provides a better characterization of the features in the original image, and that the k-means classifier implemented in our tool performs in very similar terms as the one available in ENVI software.

Table 5.1: Percentage of agreement (in terms of individual classes and also from a global point of view) after comparing the classification map in Fig. 5.6(c) with the classification map in Fig. 5.6(d).

Soil #1 (blue)	**Water #1 (green)**	**Urban (orange)**	**Water #2 (red)**	**Soil #2 (yellow)**	**Overall agreement**
63.89	96.98	99.74	89.01	94.59	88.47

Table 5.2: Confusion matrix obtained after comparing the classification map in Fig. 5.6(c) produced by the tool (with the k-means algorithm) for the Nile river image with the classification map in Fig. 5.6(d) produced by ENVI.

Class	**Soil #1 (dark blue)**	**Water #1 (red)**	**Urban (blue)**	**Water #2 (green)**	**Soil #2 (yellow)**
Soil #1 (blue)	17,550	0	0	7,270	0
Water #1 (green)	0	34,316	0	8	0
Urban (orange)	0	0	25,265	0	1,335
Water #2 (red)	10	1,069	0	58,928	0
Soil #2 (yellow)	9,908	0	65	0	23,332

5.1.5.2 Experiment 2: comparison of unsupervised classification algorithms

In this experiment a satellite image collected over the World Trade Center (WTC) area in New York City is used [see Fig. 5.7(a)]. The resolution of the image is quite high, with approximately 5 meters per pixel. Figs. 5.7(b-c) respectively show the unsupervised classification result obtained by the processing chain, with and without morphological pre-processing, using the k-means algorithm. Figs. 5.7(d-e) respectively show the unsupervised classification result obtained by the processing chain, with and without morphological pre-processing, using the ISODATA algorithm. Finally, Figs. 5.7(f-g) respectively show the unsupervised classification result obtained using ENVI's implementation of k-means and ISODATA algorithms. In all cases, we empirically used $k = 14$ and $c = 5$ as input parameters, and no spatial post-processing was applied.

As shown by Fig. 5.7, the classification results provided by k-means and ISODATA are very similar. Also, although the color class labels for the implementations are different, the classification maps provided by our implementations (without morphological pre-processing) and the ones obtained using ENVI are very similar. In both cases, the algorithm parameters have been set to exactly the same values. Table 5.3 reports the classification agreement (in percentage) measured after comparing our k-means and ISODATA classification maps with the ones obtained by ENVI (assuming the latter ones as the reference in the comparisons). As shown by Table 5.3, the agreement between the maps is always very high. The confusion matrices for k-means and ISODATA are respectively given in Tables 5.4 and 5.5. Overall, this experiment indicates that the two clustering algorithms implemented in the proposed desktop tool perform in similar terms as those available in ENVI software.

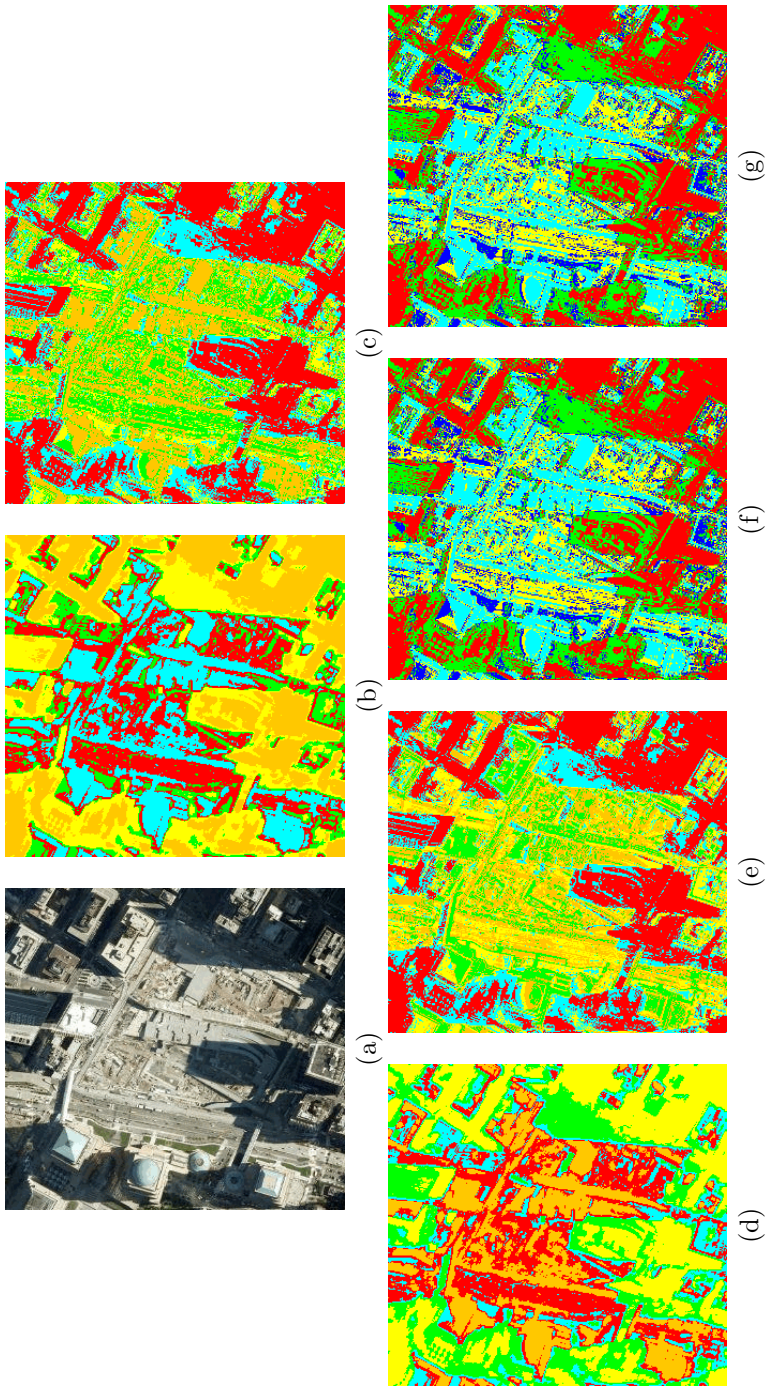


Figure 5.7: (a) Satellite image collected over the World Trade Center area in New York City. (b) Classification using our processing chain with morphological profiles and k-means. (c) Classification using our processing chain (on the original image) with k-means. (d) Classification using our processing chain with morphological profiles and ISODATA. (e) Classification using our processing chain (on the original image) with ISODATA. (f) Classification using ENVI's k-means. (g) Classification using ENVI's ISODATA.

Table 5.3: Percentage of agreement (in terms of individual classes and also from a global point of view) after comparing the k-means classification map in Fig. 5.7(c) with the classification map in Fig. 5.7(f), and after comparing the ISODATA classification map in Fig. 5.7(e) with the classification map in Fig. 5.7(g).

Clustering Algorithm	**Shadows #1 (blue)**	**Shadows #2 (red)**	**Urban areas #1 (yellow)**	**Urban areas #2 (green)**	**Urban areas #3 (orange)**	**Overall agreement**
k-means	78.26	95.49	85.56	90.65	97.39	89.47
ISODATA	85.85	77.43	91.59	92.08	97.39	88.87

Table 5.4: Confusion matrix obtained after comparing the classification map in Fig. 5.7(c) produced by our tool (with the k-means algorithm) for the World Trade Center image with the classification map in Fig. 5.7(f) produced by ENVI.

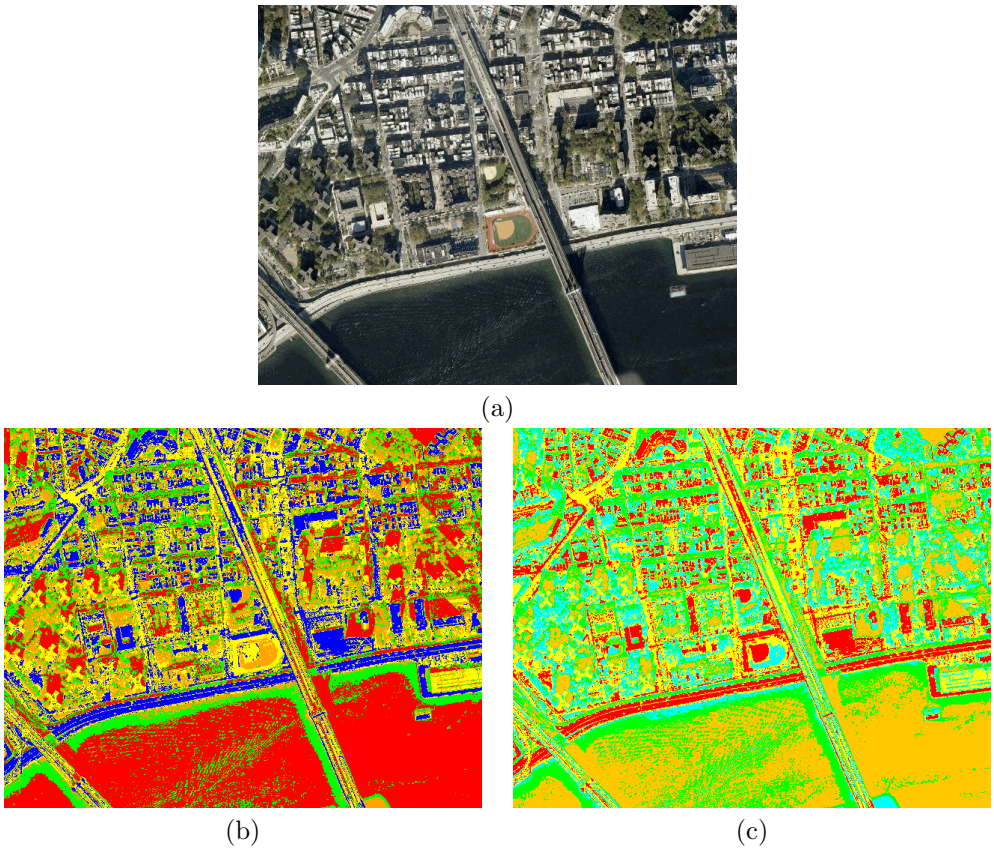
Class	**Shadows #1 (green)**	**Shadows #2 (red)**	**Urban areas #1 (dark blue)**	**Urban areas #2 (yellow)**	**Urban areas #3 (blue)**
Shadows #1 (blue)	15,142	1,192	845	0	0
Shadows #2 (red)	3,835	25,246	0	0	0
Urban areas #1 (yellow)	371	0	10,171	306	0
Urban areas #2 (green)	0	0	872	15,937	527
Urban areas #3 (orange)	0	0	0	1,338	19,635

5.1.5.3 Experiment 3: supervised classification algorithm

In this third experiment, a larger image over a different location in New York City is selected [see Fig. 5.8(a)]. In order to classify this scene in supervised fashion using the ML classifier, the areas resulting from an unsupervised classification (using k-means) of a different zone in New York City were selected, and used those areas to train the ML classifier using $c = 5$ and applying it to the original image. The resulting classification map is displayed in Fig. 5.8(b). For illustrative purposes, the classification map

Table 5.5: Confusion matrix obtained after comparing the classification map in Fig. 5.7(e) produced by our tool (with the ISODATA algorithm) for the World Trade Center image with the classification map in Fig. 5.7(g) produced by ENVI.

Class	**Shadows #1 (green)**	**Shadows #2 (red)**	**Urban areas #1 (dark blue)**	**Urban areas #2 (yellow)**	**Urban areas #3 (blue)**
Shadows #1 (blue)	20,472	0	0	0	0
Shadows #2 (red)	5,966	16,611	0	0	16
Urban areas #1 (yellow)	0	0	19,635	1,338	0
Urban areas #2 (green)	0	0	527	16,189	984
Urban areas #3 (orange)	0	2,737	0	54	10,888



(a) (b) (c)

Figure 5.8: (a) Satellite image collected over the World Trade Center area in New York City. (b) Supervised classification result provided by our implementation of ML [trained with the classification in Fig. 5.7(d)]. (c) Supervised classification result provided by ENVI's implementation of ML [trained with the classification in Fig. 5.7(f)]

achieved by the ML algorithm available in ENVI is displayed in Fig. 5.8(c). In order to obtain this map, the ML algorithm in ENVI was trained using exactly the same areas and with the same parameters as those used in our implementation. Again, Fig. 5.8 reveals that, although the color labels for the proposed implementation and the one provided by ENVI are different, the classification maps are very similar. For illustrative purposes, Table 5.6 reports the classification agreement (in percentage) measured after comparing the ML-based classification map with the one obtained by ENVI (assuming the latter as the reference). The associated confusion matrix is given in Table 5.7. These results indicate a high degree of similarity between the proposed implementation of ML and the one available in ENVI software. This experiment reveals that the proposed implementation allows a successful integration of the ML supervised technique for automatic classification of satellite images derived from Google Maps via the developed tool.

5.1.5.4 Experiment 4: performance of the GPU implementation

In this experiment the computational performance of the GPU implementation of the processing chain (including morphological pre-processing, k-means clustering and spatial post-processing) was analyzed. The implementation carried out in NVidia CUDA is compared with regard to its CPU (serial) version, implemented in C language. In this work, two different CPU-GPU configurations were used. In the

Table 5.6: Percentage of agreement (in terms of individual classes and also from a global point of view) after comparing the ML classification map in Fig. 5.8(b) with the classification map in Fig. 5.8(c).

Vegetation #1 (green)	**Vegetation #2 (blue)**	**Water (orange)**	**Urban areas #1 (red)**	**Urban areas #2 (yellow)**	**Overall agreement**
71.71	85.60	76.33	94.13	98.07	85.17

Table 5.7: Confusion matrix obtained after comparing the classification map in Fig. 5.8(b) produced by our tool (with the ML algorithm) for the World Trade Center image with the classification map in Fig. 5.8(c) produced by ENVI.

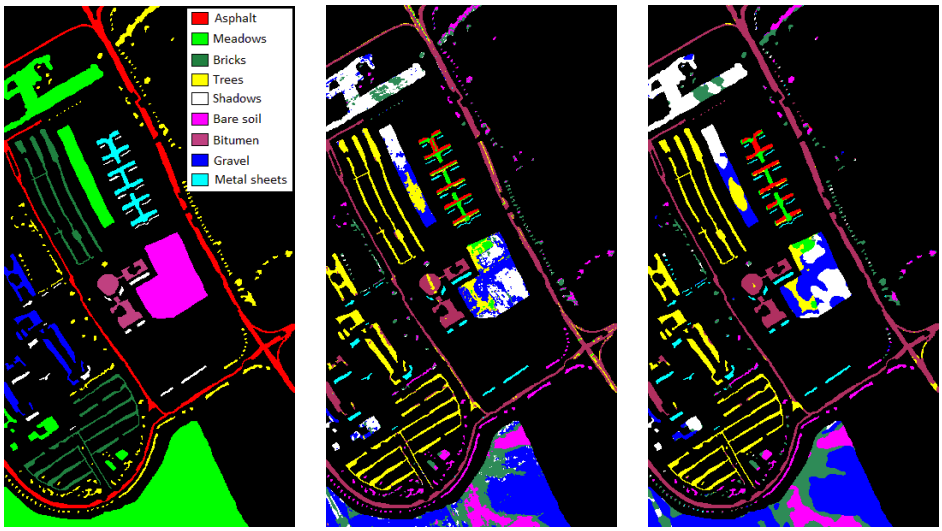
Class	**Vegetation #1 (green)**	**Vegetation #2 (oragne)**	**Water (red)**	**Urban areas #1 (dark blue)**	**Urban areas #2 (yellow)**
Vegetation #1 (green)	42,056	0	23,999	0	0
Vegetation #2 (blue)	13,902	45,234	0	0	390
Water (orange)	1,116	0	80,194	0	0
Urban areas #1 (red)	0	0	0	43,590	1,125
Urban areas #2 (yellow)	0	7,502	0	9,409	55,456

first one, an Intel Core i7 920 CPU was used with the Ubuntu 9.04 Linux operating system and the NVidia Tesla C1060 GPU. In the second one, an Intel Core 2 Duo P8700 2.53 GHz CPU was used with the Windows 7 operating system and an NVidia GeForce 9400 M GPU. Table 5.8 shows the execution times achieved for each of the CPU-GPU configurations used, as well as the speedups achieved for different image sizes and number of clusters, c. In all cases, the construction of morphological profiles with $k = 14$ and spatial post-processing with a window size of 5×5 pixels have been used. The reason for considering a fixed number of iterations k for the calculation of morphological profiles and also a fixed size of the spatial post-processing window is that in our experiments it has been observed that the GPU implementation of these two steps scales linearly with the number of GPU cores used, hence the number of clusters used c is more important as the GPU implementation does not scale linearly with this parameter as the GPU implementation of k-means is more complex. In other words, the speedups observed for the morphological processing and post-processing can be considered linear, while the speedups reported in Table 5.8 are mainly originated by the clustering stage which was also the most computationally expensive, taking approximately two thirds of the overall processing time while the morphological processing and post-processing took approximately one third of the overall computation in the considered experiments.

An important observation from Table 5.8 is that, as the image size and number of clusters c are increased, the speedup achieved by the GPUs tends to be more significant. For instance, the implementation in the Tesla C1060 GPU achieved a speedup of about 37x with regards to the CPU version for 1024×1024 image size and $c = 128$. However, the implementation in the GeForce 9400M GPU saturates for certain image sizes, achieving a speedup of about 10x in the best case. Overall, this experiment reveals that GPU architectures provide a source of computational power that can be readily used to accelerate the considered processing chain, which mainly consists of regular operations that map

Table 5.8: Processing times (in seconds) and speedups achieved with regards to the corresponding CPU (recalculating the centers of each cluster and reassigning new pixels to each of the clusters) for different GPU implementations of the considered processing chain (using different image sizes and number of clusters, c).

Parameters considered		NVidia GeForce 9400M			NVidia Tesla C1060		
Image size (pixels)	Number of clusters, c	Time CPU	Time GPU	Speedup	Time CPU	Time GPU	Speedup
512×512	5	0.026	0.252	3.26x	0.016	0.145	5.67x
512×512	64	0.042	0.496	7.60x	0.020	0.210	17.95x
512×512	128	0.064	0.764	10.29x	0.028	0.268	29.33x
1024×1024	64	0.263	3.582	6.18x	0.048	0.715	30.97x
1024×1024	128	0.356	4.376	8.76x	0.067	1.044	36.69x



(a) (b) (c)

Figure 5.9: (a) Reference map containing 9 mutually exclusive land-cover classes. (b) Classification map obtained after applying our unsupervised k-means algorithm with $c = 9$. (c) Classification map obtained after applying spatial post-processing (using a processing window of 7×7 pixels) over the classification result obtained in (b).

nicely to this kind of architecture.

5.1.5.5 Experiment 5: analysis of hyperspectral data

In this experiment the performance of the considered processing chain is analyzed using remotely sensed hyperspectral data. The scene used for experiments was collected by the ROSIS optical sensor over an urban area centered at the University of Pavia, Italy. This image has been described in section 2.3.1. Fig. 5.9(a) shows the reference data available for the scene, which comprises nine reference classes of interest including urban features (asphalt, gravel, metal sheets, bitumen and bricks) as well as vegetation features (meadows, trees), bare soil and shadows. The classification map obtained after applying the proposed unsupervised k-means algorithm with $c = 9$ (i.e., the number of classes in the reference map) is displayed in Fig. 5.9(b). Finally, Fig. 5.9(c) shows the result obtained after applying spatial post-processing (using a processing window of 7×7 pixels) over the classification result obtained in Fig. 5.9(b). Due

Table 5.9: Confusion matrix obtained after comparing the classification map in Fig. 5.9(b) produced by the proposed tool (with the k-means algorithm) for the ROSIS University of Pavia scene with the reference data in Fig. 5.9(a).

Class	**Bare soil (Magenta)**	**Asphalt (Red)**	**Meadows (Green)**	**Trees (Yellow)**	**Metal sheets (Cyan)**	**Self-blocking bricks (Sea Green)**	**Bitumen (Coral)**	**Gravel (Blue)**	**Shadow (White)**
Bare soil (Green)	290	51	0	4	561	0	0	2	0
Asphalt (Blue)	2,160	19	8,173	20	0	31	0	2	0
Meadows (Sea Green)	120	0	3,848	990	0	0	0	0	0
Trees (Magenta)	0	0	2,116	1,951	0	0	0	0	0
Metal sheets (Red)	0	1	0	5	649	0	0	0	0
Self-blocking bricks (White)	1,654	8	4,011	92	0	9	1	1	0
Bitumen (Coral)	17	6,002	5	0	5	172	1,248	476	0
Gravel (Yellow)	788	540	496	0	130	3,470	81	1,617	0
Shadow (Cyan)	0	10	0	2	0	0	0	1	947

Table 5.10: Confusion matrix obtained after comparing the classification map in Fig. 5.9(c) produced by the proposed tool (with the k-means algorithm) for the ROSIS University of Pavia scene with the reference data in Fig. 5.9(a).

Class	**Bare soil (Magenta)**	**Asphalt (Red)**	**Meadows (Green)**	**Trees (Yellow)**	**Metal sheets (Cyan)**	**Self-blocking bricks (Sea Green)**	**Bitumen (Coral)**	**Gravel (Blue)**	**Shadow (White)**
Bare soil (Green)	308	0	0	4	479	0	0	0	1
Asphalt (Blue)	2,226	1	8,337	62	0	1	0	0	0
Meadows (Sea Green)	19	0	3,799	1,163	0	1	0	0	0
Trees (Magenta)	0	0	2,141	1,478	0	0	0	0	0
Metal sheets (Red)	0	0	0	8	780	0	0	0	0
Self-blocking bricks (White)	1,768	0	3,905	139	0	3	0	0	0
Bitumen (Coral)	0	6,412	0	90	0	66	1,321	377	1
Gravel (Yellow)	708	218	467	24	86	3,611	9	1,722	1
Shadow (Cyan)	0	0	0	96	0	0	0	0	944

to the high dimensionality of hyperspectral images, in this experiment no morphological pre-processing was applied and instead, the full spectral information available from the scene were processed. For illustrative purposes, Table 5.9 shows the confusion matrix obtained after comparing the classification result in Fig. 5.9(b) i.e., clustering without spatial post-processing – with the reference data in Fig. 5.9(a), while Table 5.10 shows the confusion matrix obtained after comparing the classification result in Fig. 5.9(c) – i.e., clustering with spatial post-processing – with the reference data in Fig. 5.9(a). In both cases, class confusion can be observed for some of the classes resulting from the unsupervised nature of the proposed processing chain and the complexity of the scene. However, this example reveals the potential of the proposed tool to perform classification of different types of remotely sensed data. The inclusion of more advanced supervised classifiers, which are being planned as a future extension of this work, should significantly improve the obtained classification results by using a small percentage of the reference samples for training purposes.

5.1.6 Summary and future research lines

This subchapter has described a new desktop application for unsupervised and supervised classification of remotely sensed images. The system has been developed using the Java programming language with calls `SwingX-WS` library. The experimental results, conducted by comparing the obtained classification results with those provided by commercial products such as the popular ENVI software package, reveal that the proposed tool provides classification maps of high similarity with regard to those provided by ENVI for the same satellite imagery, but with the possibility to perform classification of any image portion available in Google Maps engine, both in unsupervised and supervised fashion, and in a computationally efficient form, through the exploitation of the fine granularity of parallelism offered by GPU architectures. The proposed tool has also been demonstrated with remotely sensed hyperspectral data.

In future developments, the plan is to incorporate additional feature extraction techniques such as attribute morphological profiles, and also other supervised classifiers such as random forests and support vector machines (SVMs) [112]. Also, it is of interest to extend the developed tool with the incorporation of content-based image retrieval (CBIR) functionalities.

5.2 A web-based system for classification of remote sensing data

Outline - The availability of satellite imagery has expanded over the past few years, and the possibility to perform fast processing of massive databases comprising this kind of imagery data has opened ground-breaking perspectives in many different fields. This subchapter describes a web-based system[13], which allows an inexperienced user to perform unsupervised classification of satellite/airborne images. The processing chain adopted in this work has been implemented in C language and integrated in our proposed tool, developed with HTML5, JavaScript, Php, AJAX and other web programming languages. Image acquisition with the applications programmer interface (API) is fast and efficient. An important added functionality of the developed tool is its capacity to exploit a remote server to speed up the processing of large satellite/airborne images at different zoom levels. The ability to process images at different zoom levels allows the tool an improved interaction with the user, who is able to supervise the final result. The previous functionalities are necessary to use efficient techniques for the classification of images and the incorporation of content-based image retrieval (CBIR). Several experimental validation types of the classification results with the proposed system are performed by comparing the classification accuracy of the proposed chain by means of techniques available in the well-known Environment for Visualizing Images (ENVI) software package.

5.2.1 Overview

Remote sensing image analysis and interpretation have become key approaches that rely on the availability of web mapping services and programs. This resourceful increase has led to the exponential growth of the user community for satellite/airborne images, not long ago only accessible by government intelligence agencies [2, 3]. In particular, the wealth of satellite/airborne imagery available from Google Maps, which now provides high-resolution images from many locations around the Earth[14], has opened the appealing perspective of performing classification and retrieval tasks via the Google Maps application programming interface (API).

The combination of an easily searchable mapping and satellite/airborne imagery tool such as Google Maps, with advanced image classification and retrieval features [47], can expand the functionalities of the tool and also allow end-users to extract relevant information from a massive and widely available database of satellite/airborne images. The Google Maps service is free for non-commercial use. It should be noted that the current version of Google Maps does not allow using maps outside a web-based application (except with a link to Google Maps). Here Google Maps is used purely as an example to demonstrate that if we have a data repository we can use the tool we propose, and the logo and Google Maps terms of service[15] are always in place. The characteristics of Yahoo Maps are similar to Google Maps (though the spatial resolution of the satellite/airborne imagery in Yahoo Maps is generally lower than Google Maps). OpenStreetMap is a collaborative project aimed at creating a free editable map of

Part of this subchapter has been published in::

A. Ferrán, S. Bernabé, P. G. Rodríguez and A. Plaza, "A Web-Based System for Classification of Remote Sensing Data", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 4, pp. 1934–1948, August 2013 [JCR(2012)=2.874].

[13]Available online: http://hypergim.ceta-ciemat.es
[14]http://code.google.com/apis/maps/index.html
[15]https://developers.google.com/maps/terms?hl=en

Table 5.11: Comparison between the main functionalities of Google Maps, Yahoo Maps and OpenStreetMap

	Google Maps	Yahoo Maps	OpenStreetMap
Without restrictions of use	No, up to a certain limit	Yes	Yes
Hybrid satellite view	Yes	Yes, low visibility	No
High resolution imagery	Yes	No	No
Zooming levels	Very high quality	High quality	Medium quality
Error correction	Low	Low	Very high
Smoothness in navigation	Very high	High	High
Adaptivity for desktop and web applications	High	High	High

the world, a design inspired by sites such as Wikipedia[16]. For illustrative purposes, Table 5.11 shows a comparison between the main functionalities of the previous map servers.

As shown by Table 5.11, Google Maps offers important competitive advantages, such as the availability of high resolution satellite/airborne imagery, the smoothness in the navigation and interaction with the system, the availability of a hybrid satellite view which can be integrated with other views (e.g., maps view), and adaptivity for general-purpose web applications. It should be noted that other open standards for geospatial content such as those included within the open geospatial consortium (OCG) cannot currently provide complete world coverage at high spatial resolution as it is the case of Google Maps. This is why it has been decided to use Google Maps service as a baseline for our system. However, our system has been designed in a completely modular and open way, which allows for the incorporation of alternative data sources and software implementations in future developments. On the other hand, a feature which is currently lacking in Google Maps is the unsupervised or supervised classification of satellite/airborne images at different zoom levels [7, 8], even though image classification is widely recognized as one of the most powerful approaches in order to extract information from satellite/airborne imagery [9–11]. The incorporation of such a function into Google Maps would allow relevant information withdrawal from a massive, widely available database of satellite/airborne images and the possibility to perform content-based image retrieval (CBIR) tasks [15], which are of great interest for the exploitation of this and other databases satellite/airborne images.

In this subchapter, we describe a web-based system (which represents a follow-up of our previous work in [113] and presented in the previous subchapter) that allows an inexperienced user to perform an unsupervised classification of satellite/airborne images obtained via Google Maps. Specifically, our web-based system incorporates a fully unsupervised processing chain based on two well-known clustering techniques: ISODATA [106] and k-means [107], followed by spatial post-processing based on majority voting [105]. The processing chain has been implemented in C language and integrated into our proposed tool using open standards and free software tools including HTML5, JavaScript, Php, AJAX, and other web-based programming languages.

In our previous work [113], the tool was implemented as a desktop system and developed in JAVA. The main drawbacks resolved were:

- The image acquisition with the API (only compatible with web applications) is now much faster and efficient because we obtain the image directly, a full mosaic compared with the library swingX-WS, in which we had to manually create the mosaic. In all cases (following the Google Maps terms

[16]http://www.wikipedia.org

of service), we do not remove the watermarks, which are visible when using the API. In our tests we have experienced that the watermarks generally do not affect the classification results from a general point of view.

- A very important added functionality of our newly developed tool is the fact that it exploits a remote server to speed up the processing of large images at different zoom levels versus the previously available desktop system, in which large images are very slow to be processed despite the use of GPU computing capabilities. In the future we are planning to incorporate high performance computing functionalities [18,24,57,114,115] to the remote server also using GPU technologies [95,97,116–120].

- The ability to combine and change the color of the class labels and process images at different zoom levels allows the tool an improved interaction with the user who is now able to supervise the final result.

- Last but not least, we highlight the flexibility of our proposed system since it can be easily extended to other map servers and software platforms. In this regard, our system has been designed in a way that it would easily allow replacement of map servers and software implementations following a highly modular design.

The remainder of the subchapter is organized as follows. Section 5.2.2 describes the system architecture, including relevant aspects such as the map, server and client layers. Section 5.2.3 describes the processing chain implemented by the proposed methodology, including aspects such as the image acquisition process, the graphical user interface (GUI) that allows end-users to interact with the proposed system, the image processing algorithms implemented, and the procedure adopted for data saving and end product distribution to the users. An experimental validation of the classification results obtained by the proposed system is performed in section 5.2.4 by comparing the classification accuracy of the proposed chain in terms of the techniques available in the well-known Environment for Visualizing Images (ENVI) software package[17]. Finally, section 5.2.5 concludes the subchapter with some remarks and hints at possible future research. The previous functionalities are necessary to use efficient techniques for image classification and the incorporation of content-based image retrieval (CBIR), which are main goals in both systems. Regarding the availability of other client/server system designs for remote sensing data processing, we are aware of developments in the CBIR domain such as the KIM[18] system (developed by the European Space Agency). However, we are not aware of similar systems being able to provide fast, scalable and advanced data processing of remotely sensed imagery with high spatial resolution, such as those provided by our application.

5.2.2 System architecture

This section describes the architecture of the system, displayed in Fig. 5.10. It is a web application comprised of several layers or modules. Each module serves a different purpose, and the technology adopted for the development of the system is based on open standards and free software. A combination of these has been used for the development of the system.

As shown by the architecture model described in Fig. 5.10, the proposed system can be described from a high level viewpoint using three different layers, which are completely independent from each other. Due to the adopted modular design, any of the layers can be replaced. Also, the system is fully

[17] http://www.exelisvis.com/language/en-us/productsservices/envi.aspx
[18] http://rssportal.esa.int/deepenandlearn/tiki-index.php?page=KIM+Project

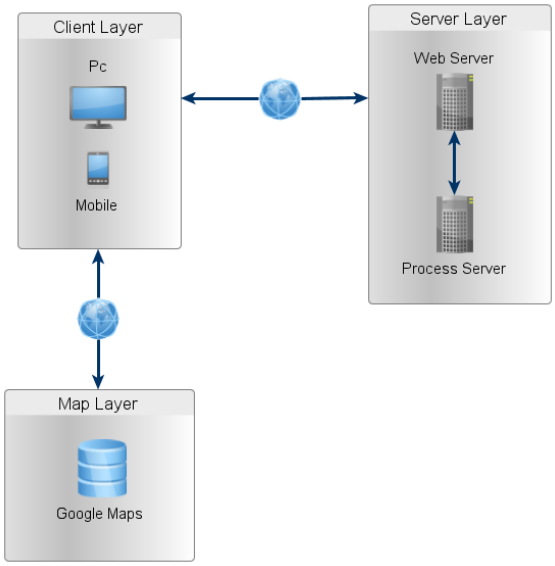


Figure 5.10: Architecture of the proposed system expressed in the form of different modular layers.

scalable, allowing for the incorporation of additional layers and additional data processing algorithms that can be included in the form of components into our framework. This is done in user-transparent fashion, and the modular design of processing algorithms in the form of components (which receive data as input and provide processing results) allows for the incorporation of additional components in scalable fashion and without the need to modify the proposed system, only the pool of data processing algorithms in the compute server. Moreover, the design of the system in the form of layers (map, client and server) allows for the incorporation of additional resources in each of the layers without modification of the system. A good example is the possibility to incorporate additional compute resources such as GPUs, or even additional map servers, which are fully supported by our current implementation of the system.

The communication between two layers is carried out over the Internet via the hypertext transfer protocol (HTTP)[19]. As a result, the system performance will depend largely (as expected) on the available bandwidth. Both the map layer (currently provided by Google Maps) and server layer (by ourselves) are available from any location in the world. Each adopted layer is described in detail next.

5.2.2.1 Map layer

This layer contains the source imagery data to be used by the system, i.e., the image repository. Google Maps is used in the current version by means of the Google Maps API V3 as a programming interface intended for accessing the provided maps. The current framework is limited to the types of maps provided by Google Maps. All types of maps provided by the API V3 can be used, including roadmaps (2D mosaics), satellite/airborne images, hybrid view (mixed satellite/airborne images and roadmap, superimposed), or terrain (physical relief). Also, all the potentials and functionalities provided by the Google Maps API V3 are included (this comprises management of zoom levels, image centering, location by geo-spatial coordinates, etc.).

Although Google Maps is now used by our system as a repository of images, the system is open and could support other possible alternative or complementary repositories such as Yahoo Maps or

[19] http://www.w3.org/Protocols/

OpenStreetMap (thus associating map data and meta-data to satellite/airborne imagery available in other repositories). These systems are all accessible free of charge and are easy to include in our proposed platform. In fact, the image repository can be used to capture any satellite/airborne images displayed by the Google Maps engine, and most importantly the images can be captured at different zoom levels. Even a single image can be extracted at different zoom levels, which is obtained by different image sizes and resolutions. This feature offers significant advantages in the accurate analysis of geo-registered satellite/airborne imagery at different resolutions [121].

5.2.2.2 Server layer

The server layer is one of the main components in the system. It is formed by two sub-modules: web server and compute server. The former is the part of the system hosting the source code of the application (developed using HTML5, Php, JavaScript and CSS) and deals with the incoming traffic and requests from client browsers. We have used the Apache web server due to its wide acceptance, performance, and free-of-charge license. Further, Php is used both in the server layer and also for managing the communications between the clients and the web server (mainly dominated by the transmission of satellite/airborne imagery to be processed), and the web server and the compute server (intended for the processing of satellite/airborne images).

The compute server is mainly in charge of the actual image processing tasks which comprise clustering using k-means [106] and ISODATA [107] algorithms, and spatial post-processing [105]. The compute server receives the processing requests from end-users, manages them effectively by resorting to a remote server (in the future, we will use the GPU cluster made up of 44 NVidia Tesla C1060 GPUs[20] connected to this server by means of efficient OpenCL implementations), and then provides the obtained result to the end-user. The web server and the compute server are currently hosted on the same machine, which in our case is motivated by the fact that the processing capacity of the server was experimentally observed to be high enough to support also the computational demands introduced by the map layer, but the system also allows using different machines for this purpose, allowing for the incorporation of additional processing modules other than ISODATA, k-means and spatial post-processing.

5.2.2.3 Client layer

The client layer defines the interactions between the user (through an Internet browser) and our system. Only one web page is needed as user interface thanks to the adopted AJAX and JavaScript technologies, which allow for the web interface update without the need for interactions with the web server. At this point, it is important to emphasize that AJAX is a programming method (not a piece of software) and that it is built on JavaScript (not a standalone programming language). The design of the web interface has been done using jQuery UI, which provides built-in JavaScript modules that are attractive, easy to use and freely available. The interaction between the user and the client web interface is captured by the event handlers of the jQuery libraries, and executed at the local browser as JavaScript, only run on the browser in our implementation.

The only user-based actions transmitted to the server layer are those related to the processing of satellite/airborne images. In this context, the images to be processed are transmitted to the server using AJAX-based requests, and the web server provides such requests to the compute server (in our case, implemented in the same machine) so that the compute server can process the images very efficiently and produce a result that is then transmitted back to the client layer. This process is illustrated in

[20] http://www.nvidia.com/object/personal-supercomputing.html

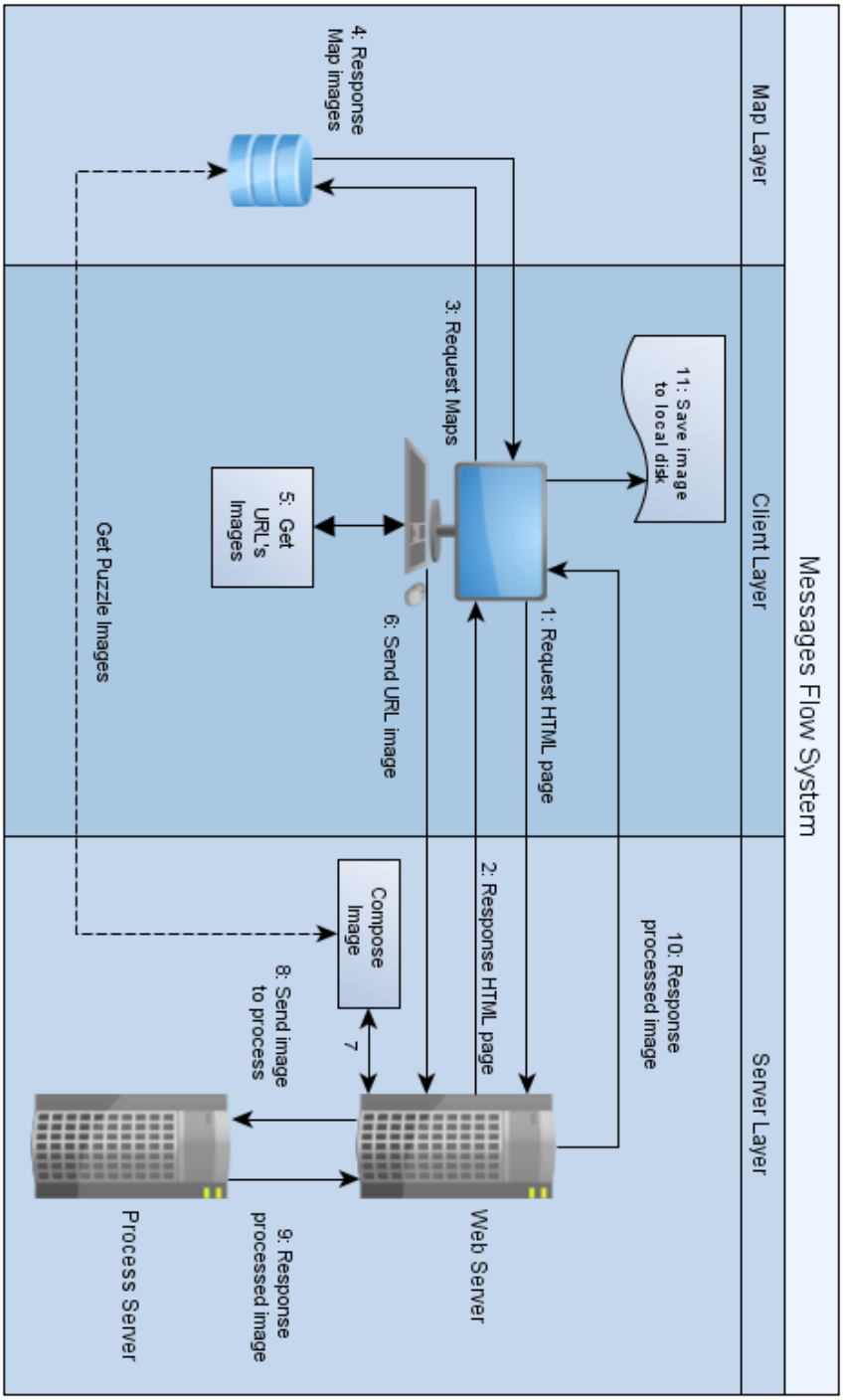


Figure 5.11: Interactions between the three main layers (map, client and server) of our system.

Fig. 5.11, which also shows how the client layer performs requests to the map layer in order to update the maps which are being handled by the end-user. This step comprises operations such as zooming, changing locations in the map, creation of maps, and selection of the specific view in which the processing will be accomplished (satellite, roadmap, hybrid, etc.).

In order to understand the interactions between the different layers of our system (see Fig. 5.11), an example is provided next about the flow of a processing request started by the client in the system and the different steps needed until a processing result is received by the end-user. The following steps can be identified in Fig. 5.11:

1. First, the client starts the use of the system by requesting a web page from the local Internet browser. This results in an HTTP request to the web server.

2. The web server receives this request and provides the client with an HTML web page and all necessary references (JavaScript libraries, CSS, etc.).

3. At this point, the client requests from the map server the information needed to perform the map modification locally (i.e., zooming). This operation is transparent to the system, and the requests are performed via messages from the client to the map server.

4. The map layer returns the information requested by the client in the form of updated maps that will be locally managed by the end-user.

5. A capture with all the URL addresses associated to each portion that compose the full map is performed to send this information to the web server. This process is locally managed at the client by means of JavaScript functions. We emphasize that the end-user can decide the zoom level and the image view (street, satellite, hybrid, etc.) of the map image to be processed.

6. Now, the Universal Resource Locator (URL) addresses associated to each portion of the full image are sent to the web server by means of AJAX functions and asynchronous requests. In this way, the interaction with the application at the client layer can continue while the packet is being transferred to the server.

7. The web server composes the full image by accessing to the Google Maps repository.

8. The web server provides the image to be processed to the compute server. Our system thus delegates the processing task to an independent remote server system that takes care of the processing task independently from other layers in the system.

9. Once the image has been processed, the compute server returns the obtained result to the web server. In our current implementation both the web server and the compute server are implemented in the same machine, hence in this case the communications are minimized.

10. Finally, the processing result is returned to the client so that it can be saved to disk as the final outcome of the adopted processing chain.

11. The client can save processed image to local disk.

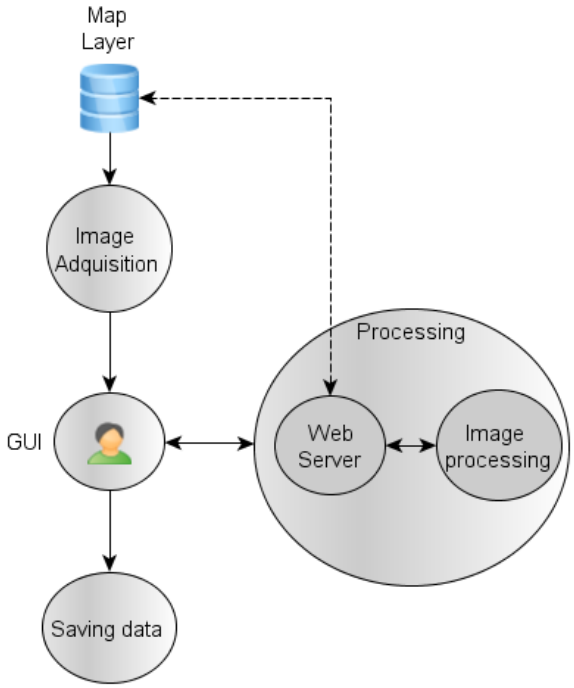


Figure 5.12: Flowchart describing the methodology adopted for the development of the system.

5.2.3 Methodology

This section describes the methodology adopted for the development of the proposed system. Several main tasks have been identified: image acquisition, graphical user interface (GUI), web server, image processing and image saving. These tasks, summarized in the flowchart given in Fig. 5.12, will be described in detail in this section.

5.2.3.1 Image acquisition

Image acquisition is the starting point of the system operation. The images to be processed are considered from two different viewpoints. On the one hand, the images are parts of a map and, on the other, the images can be considered as specific captures or snapshots of a larger map. The maps are dynamic entities that can be dragged, zoomed (i.e., displayed in more or less detail), but the captures can be seen as static parts of a map which are selected by the end-user via the interface. These captures or snapshots can then be sent to the server and processed in spite of the components of the map layer, in our case supported by the Google Maps engine.

The methodology implemented in our system for the image capture retrieval from the map layer has been developed using JavaScript libraries. These processes have access to the collection or "puzzle" of images that compose a certain map, thus taking advantage of the browser's cache memory to optimize such an operation. The query is directed to the map layer in case that the image is not already in the cache memory (a situation that seldom occurs). This option leads to some advantages, mostly to high speed achieved by the system in the task of image captures regardless of the latency of communications with the server. This feature reduces the communication traffic and increases the performance in the local management of image captures.

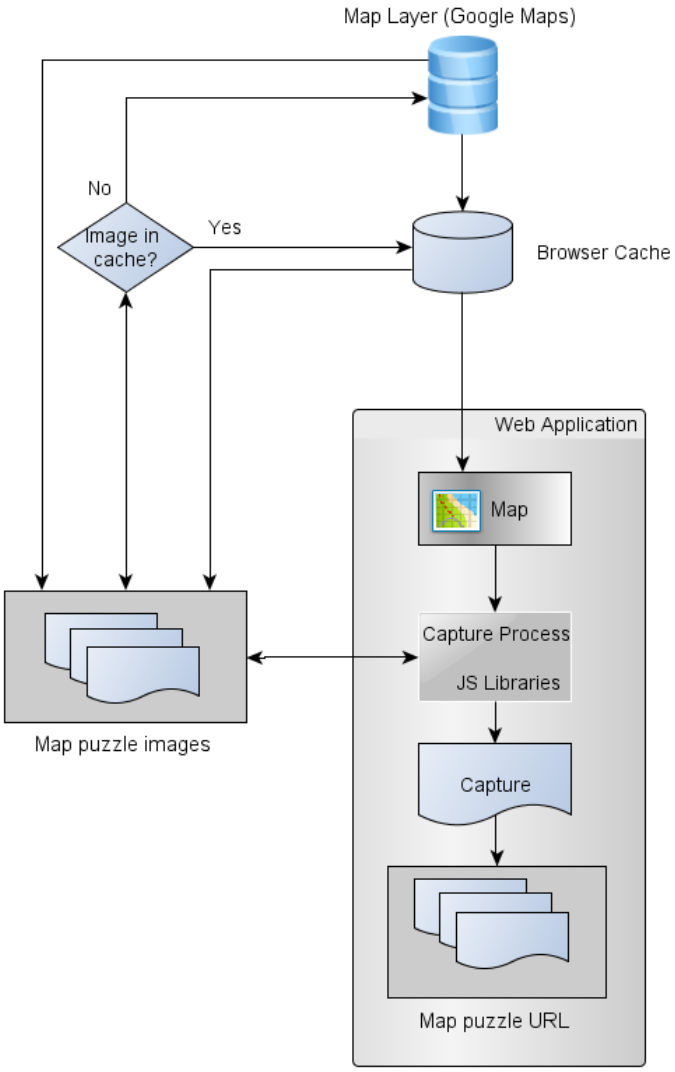


Figure 5.13: Flowchart describing the strategies adopted for managing image captures.

In order to achieve such a functionality, several layers of images from the server are considered once the image captures have been processed, thereby obtaining a stack of images in which each layer represents a class (as determined by the considered processing algorithms, i.e., k-means and ISODATA).The layers are completely independent, thus allowing visualization as individual entities or as a combination between layers, providing great flexibility in the analysis of the obtained results and a specific management of layers. Finally, the system also allows for the rapid acquisition of multiple captures from the same map, along with the simultaneous operation of multiple maps. Fig. 5.13 shows the strategy adopted by our system for image capture management.

5.2.3.2 Graphical user interface (GUI)

GUI is important because it is the visible part of our system and allows the user to perform all operations available in the application. An HTML page and JavaScript libraries have been used for

development. These libraries are the jQuery framework (version 1.6.2) and jQuery-UI (version 1.8.16). Other developments using JavaScript libraries have been accomplished in order to add new functionalities to the created widgets. As the whole GUI runs on the client layer, usability and speed of response are guaranteed, and the adopted design is very flexible. The GUI has been developed in the form of a single HTML page, avoiding repeated requests and responses back to the server.

Several features within the HTML5 standard have been used in order to design the GUI for our system application. The most important object used is the canvas, which allows for an efficient management of the images to be processed. Pixel-level access to the content of a canvas object is possible, thus largely simplifying the implementation of image processing operations. The container can carry out map captures (snapshots), to access the information of each pixel of the capture, to transfer all such information to the server layer, and to save the obtained information (processed images). As noted above, a stack of images is obtained as an outcome with as many layers as the classes identified by the processing algorithms k-means and ISODATA. The obtained layers can be merged in order to simplify the interpretation of the obtained results.

To achieve the aforementioned functionality, the only requirement at the client layer is the use of browsers that support HTML5. Some browsers currently support some (but not all) of the features in HTML5. One of the key features that client browsers must support in the context of our application is the so-called attribute crossOrigin of the image object in HTML[21]. Should this feature not be supported, the application will display a security error and will not work correctly.

Our application is fully accessible from mobile devices: although the application is developed to be accessed primarily from a PC browser, it is also operational on mobile applications, such as smartphones or tablets, as far as these devices use browsers that support HTML5. Fig. 5.14 shows an example of the designed GUI, designed in a simple but functional fashion. It consists of a single web page with a working panel, a container of maps, and a capture container. The work panel features the creation of maps, the update of a map's captures, the zoom level shift, and the selection of processing parameters for the k-means, ISODATA and spatial post-processing algorithms implemented for image analysis tasks in the current version. The map container can hold multiple maps of individual sizes, while the capture container allows performing several captures of the same map. The different captures of the same map always retain the same size as the original map size.

Finally, Fig. 5.15 provides a display of our application. Different layers are managed in this case. In this screenshot, we aim to exemplify how an image has been processed and several classes have been identified by one of the considered processing algorithms. The system allows users to display, hide and merge different classes identified by such processing algorithms. The colors associated to these classes can also be edited and fully customized. The layers can be superimposed on the original image (capture) to be processed, thus generating a final product which comprises an unsupervised classification of a certain area whose location, size, dimensionality, zoom level, etc. are defined by the end-user.

5.2.3.3 Image processing

Two different modules deal with image processing in our system. First, the web server receives the URLs corresponding to the portions of the image to be processed, then composes the full image and processes it with the methodologies implemented in the system, and then forwards the obtained result to the client layer. Secondly, the compute server processes the image effectively (e.g., applying the considered clustering and spatial post-processing algorithms). The image processing tasks are described next at

[21] http://www.w3.org/TR/cors

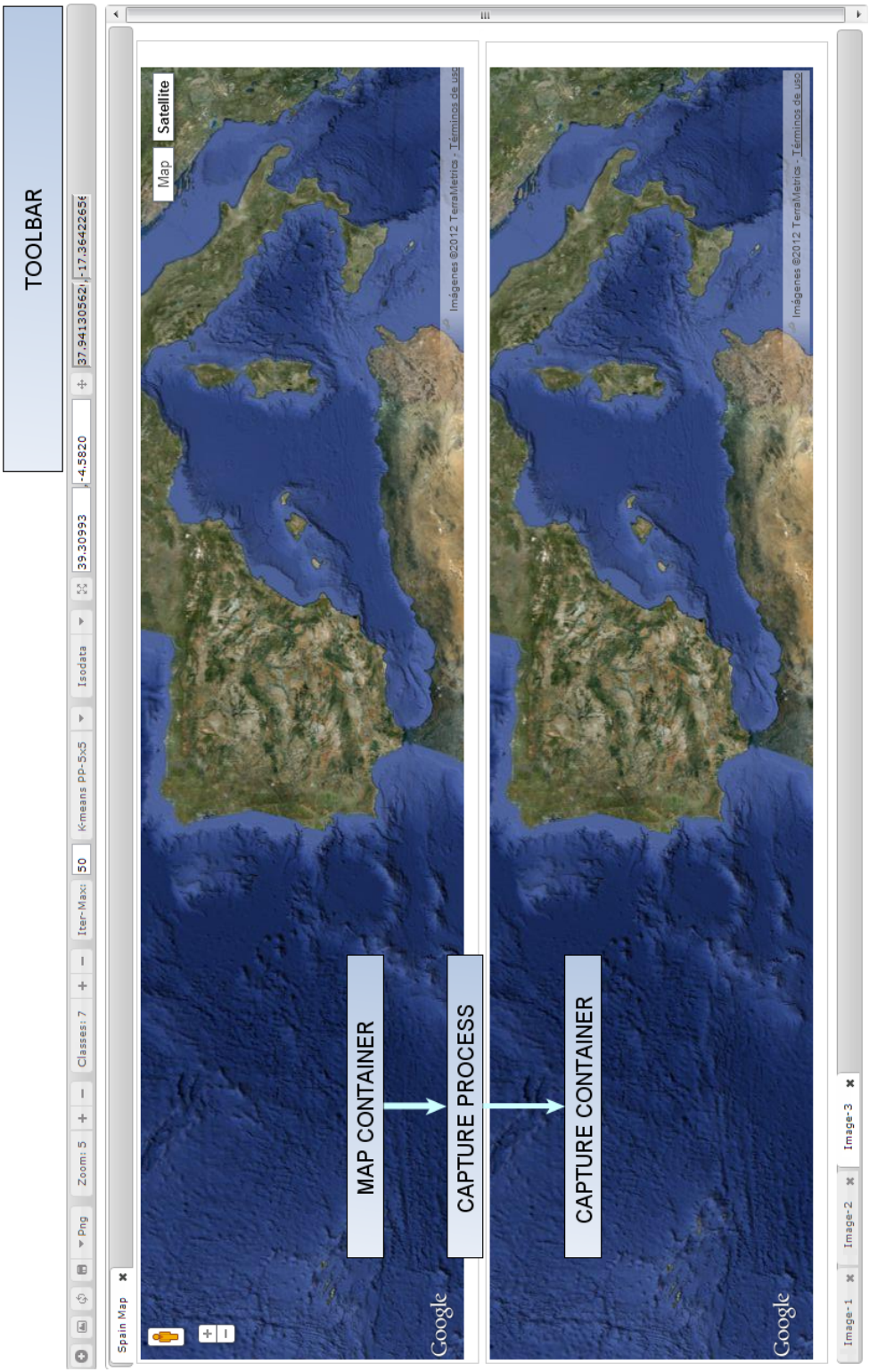


Figure 5.14: Graphical user interface in the application.

Figure 5.15: An example illustrating the management of different layers in the application.

both layers, and in particular, the method for image processing request management is explained.

1) Handling image processing requests: The main task of the web server is to receive requests from clients. Such requests are handled as follows. First, the client selects the image capture to be processed and a function gets the URL of each portion in order to compose the full image. In this way the information needed to compose the full image can be encapsulated into an AJAX request together with all processing options, and sent to the web server asynchronously using the HTTP protocol, and specifically a function called post.

Once the web server has received the full request, a Php function creates an image from the URL received and then obtains the parameters needed to accomplish the processing task (number of classes, number of iterations, etc.). Then the compute server assigns a timestamp to the image to identify it as a unique entity. The image is stored onto a temporary folder, and the web server calls the compute server indicating the location and unique identification of the image, so that the data can be efficiently processed by the compute server (both the web server and compute server can access the image structure and its particular location). Finally, the compute server generates a final product (in our case, the processed image) from the information received, and stores the output on another temporary location. After the processing task has been completed, the web server takes control again. It collects the final product generated by the compute server and sends it to the client in response to the original AJAX request originated at the client layer, thus closing the communication cycle with the client that originated the request.

2) Performing the actual image processing task: Next we describe in more detail how the actual image processing task is performed at the compute server. C language is used to perform the analysis of the input image, thus producing a final product (processed image) which is stored on a different location. As described above, the communication between the system layers (i.e., the different modules that compose the design of our system) is performed using system calls from the web server to the compute server. This is done by an asynchronous call using the method `POST` of AJAX. The call is received by the web server, which creates a new processing thread and performs a call to the compute server using sentences from the operating system. In our case, these sentences are simple invocations to the processing software using input (image to be processed) and output (processed image) parameters as follows: `$return=shell_exec(parameters).` The processing thread now simply awaits the finalization of the processing task and returns the obtained result to the web server. Since each thread is independent from each other, there are no possible conflicts between different operating system calls. This is because the *write* operations in disk are independent as a result of the fact that unique image identifiers (timestamps) are used. Currently our system only uses one CPU and, hence, the signaling can still be handled in a conventional way.

Fig. 5.16 summarizes the whole image processing strategy adopted by our proposed system, from the receipt of the URLs that allow for the composition of the full image, to the generation of an AJAX request in order to start the processing cycle at the web server, and ultimately to the execution of the request and the compute server, and the provision of the final product (processed image) back to the client that initiated the request. An important final step of the process is to overlay the final product with the original image to be processed. This is done at the client, once the processing cycle has been finalized. The method used to perform this task is putImageData, also a method of the canvas object. Fig. 5.16 reveals a modular design with clearly defined interactions between the different layers of the system.

Finally, Fig. 5.17 shows a processing example of an image captured from Google Maps corresponding to the Iberian Peninsula. As indicated in Fig. 5.15, many classes could be identified in the considered

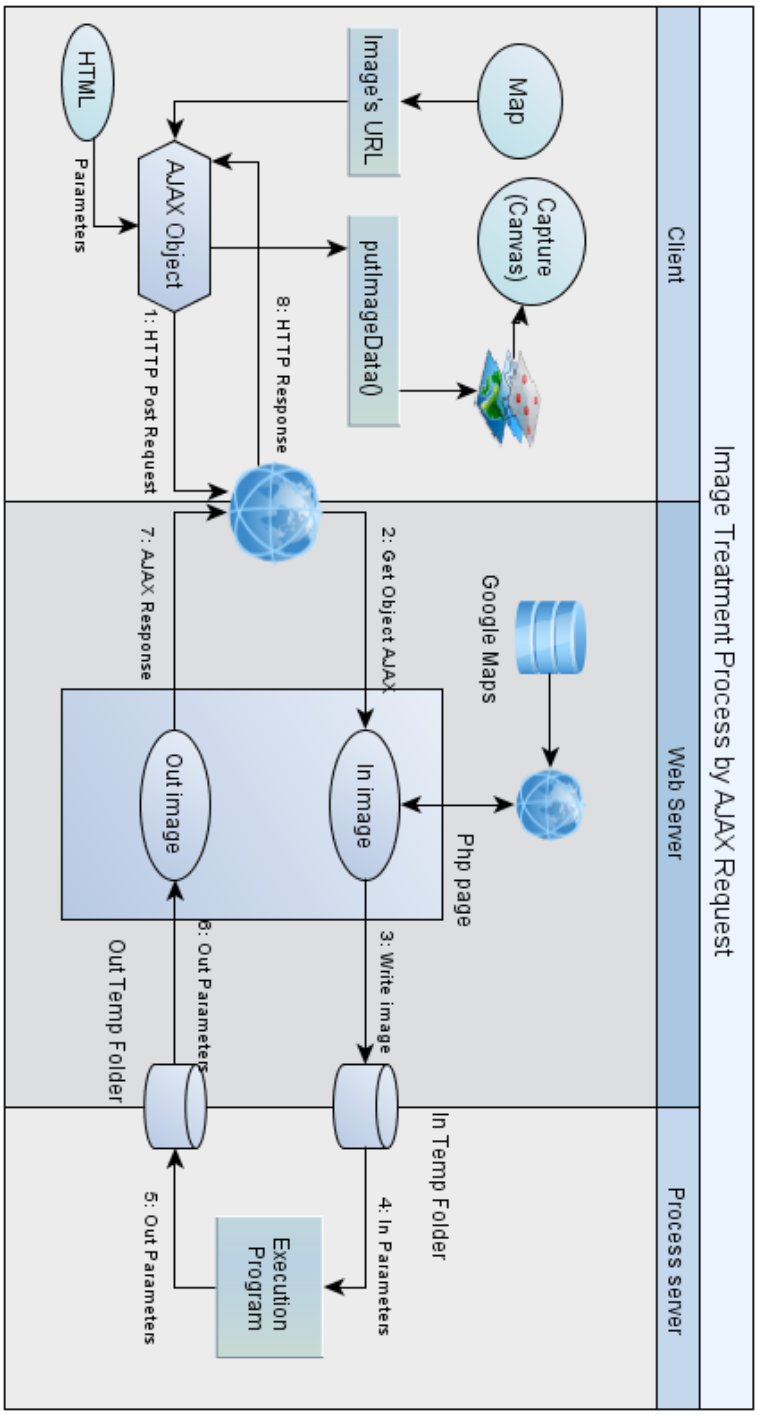


Figure 5.16: A summary of the whole image processing strategy adopted by our proposed system.

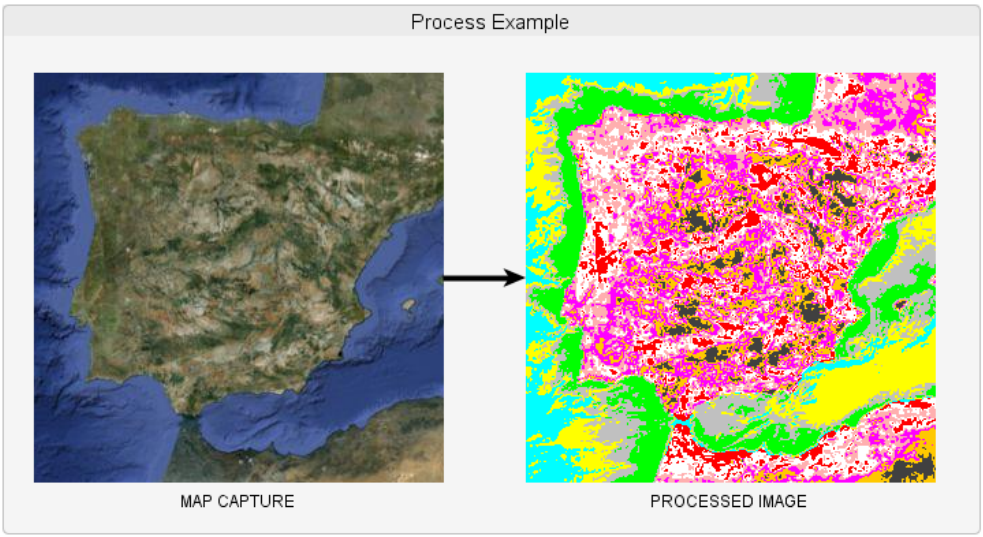


Figure 5.17: Processing example, using the proposed system, of a Google Maps image of the Iberian Peninsula. The example shows that the classification can be refined by merging classes.

test case, including different ones in the water areas. The combination of classes (functionality included in the system) can lead to improved results by joining different classes (e.g., belonging to water).

3) Saving the final results: This part specifies how final results are saved, as this requires a special treatment in the implementation. Specifically, the generated product is not stored on any server when the processing is completed, and it is only located in the local memory of the browser at the client. The results can be expressed in different forms, e.g., as a processed image, as a collection of layers that can be superimposed with the original data set, or as a combination of both. Two specific actions are taken:

1. A JavaScript library (called canvas2image) saves the contents of the canvas object on the local device using different image formats, such as JPEG, PNG or Bitmap.

2. The combined result is saved after putting together different layers of the results inside the canvas object by applying another canvas container which integrates all the data layers to be displayed. Once the image is saved, the initial container is not retained. This process is transparent to the user and is also optimized from the viewpoint of computational performance. Fig. 5.18 displays the content of a canvas object, which is the container of both the original image capture and processed image, decomposed in the form of different layers after the processing is completed.

5.2.4 Experimental results

This section describes the experimental validation of our developed system by using satellite/airborne images obtained from Google Maps across different locations. The validation is conducted by means of the following steps:

1. An experiment with the k-means unsupervised classification algorithm by selecting a satellite image over an urban area (Pavia city, Italy). This choice represents a challenging classification scenario due to the presence of complex urban features. The validation has been conducted by evaluating

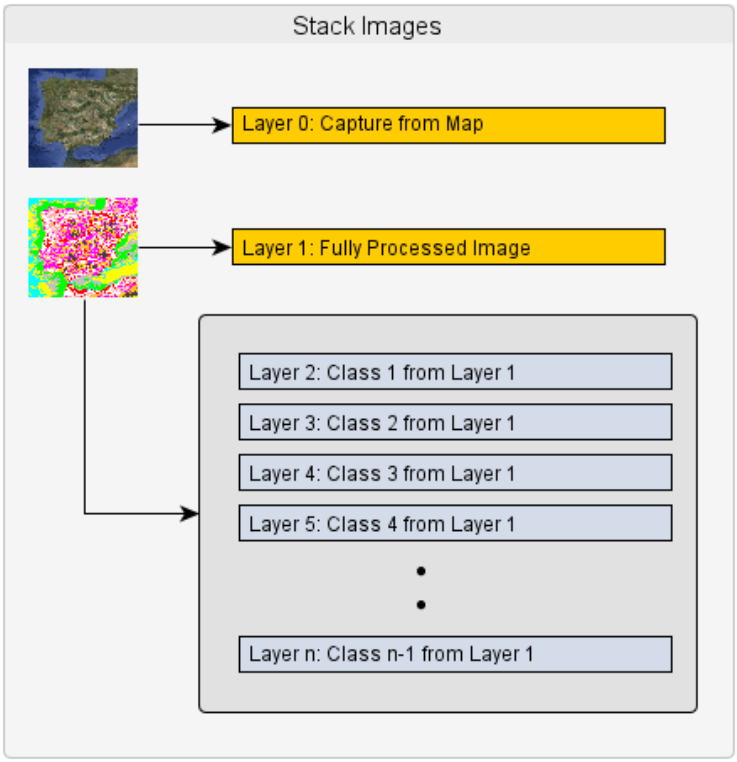


Figure 5.18: Structure of the canvas object that contains the initial image to be processed and the outcome of the processing.

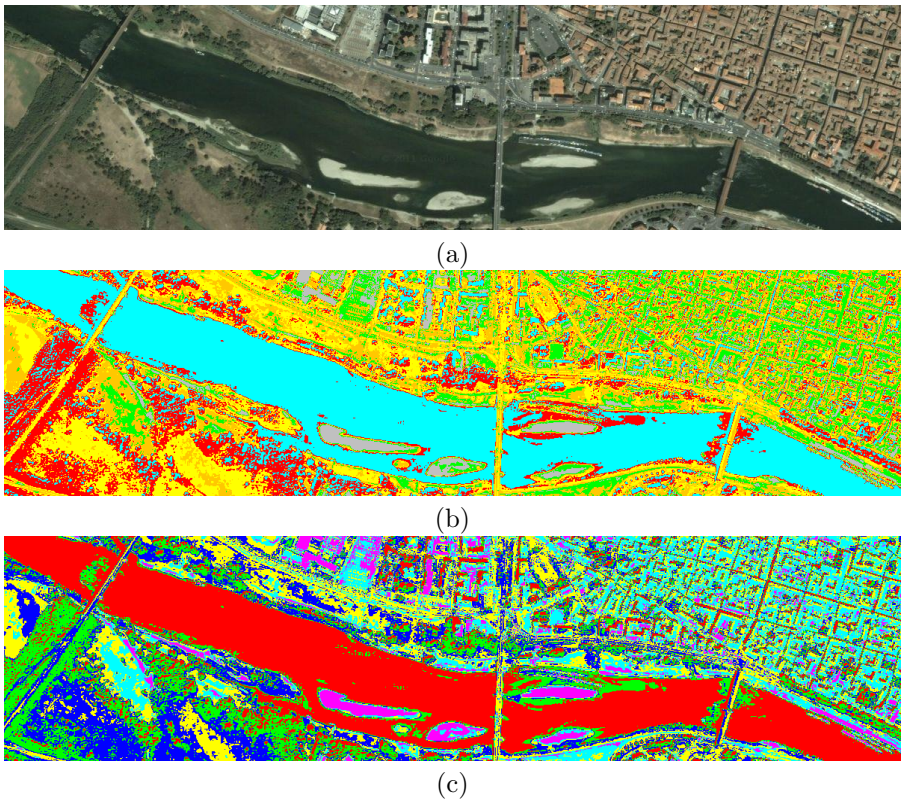
the agreement between the classification results provided by our implemented version of k-means in relation to those available in the well-known ENVI commercial software package. We adopt the same parameters when running our implementations as those available in the ENVI package.

2. The second experiment is a similar analysis, but this time based on a satellite image collected over the city of Mérida, Spain. This area contains archaeological remains from Roman times, and was used in our context to examine how the web-based tool can work in the context of archaeological-oriented remote sensing applications. In this experiment, we also evaluate the impact of using the ISODATA algorithm and spatial post-processing over the considered processing chain based on a visual assessment of the classification results obtained.

3. The first two tests used Google Map images, obtained at the highest level of zoom available, whereas in the third experiment, a satellite image is obtained over the Amazon river in South America by using a higher zoom level that seeks to evaluate our tool when processing much larger areas of the Earth's surface. In this experiment we also evaluate the computational performance of the server-client architecture developed for the fast processing of massive data sets.

5.2.4.1 Experiment 1: Validation of the k-means unsupervised classification algorithm

For this experiment, Pavia, Italy [see Fig. 5.19(a)] represents a challenging classification scenario due to the presence of complex urban features. The spatial resolution of the image is approximately 1.2 meters



(a) (b) (c)

Figure 5.19: (a) Satellite image collected over the city of Pavia, Italy. (b) Classification using our processing chain implemented with k-means and $c = 6$ classes. (c) Classification using ENVI's k-means implemented with the same parameters.

Table 5.12: Percentage of agreement (in terms of individual classes and from a global point of view) after comparing the classification map in Fig. 5.19(b), produced by our tool (with the k-means algorithm) with the classification map in Fig. 5.19(c), produced by ENVI.

Water (Cyan)	Arid soil (Grey)	Grove (Red)	Urban areas #1 (Orange)	Semiarid soil (Yellow)	Urban areas #2 (Green)	Overall agreement
100.00	78.64	90.40	75.65	81.79	75.01	83.58

per pixel. Fig. 5.19(b) shows the unsupervised classification result obtained by the proposed processing chain, using the well-known k-means algorithm, implemented to search for a total of $c = 6$ clusters [107]. No spatial post-processing is performed, thus the spectral clustering performance of the algorithm takes place without any spatial information.

Fig. 5.19(c) shows the classification result obtained by the k-means algorithm implemented by Research Systems ENVI software, using also $c = 6$ classes. As shown by comparing Fig. 5.19(b) and 5.19(c), the color labels obtained in the different classification results are different, but the classification maps provided by our processing chain applied to the original satellite image in Fig. 5.19(b) and by ENVI's implementation of k-means in Fig. 5.19(c) are similar. Table 5.12 reports the classification agreement (in percentage) [47] measured after comparing our processing chain result, based on k-means classification, with the one obtained by ENVI (assuming the latter as the reference). As shown by Table 5.12, the agreement between the obtained classification maps is always very high regardless of the

Table 5.13: Confusion matrix obtained after comparing the classification map in Fig. 5.19(b), produced by our system (with the k-means algorithm) with the classification map in Fig. 5.19(c) produced by ENVI.

Class	Water (Red)	Arid soil (Magenta)	Grove (Green)	Urban areas #1 (Yellow)	Semiarid soil (Blue)	Urban areas #2 (Cyan)
Water (Cyan)	99,927	0	5,562	0	0	0
Arid soil (Grey)	0	12,970	0	0	0	0
Grove (Red)	0	0	52,368	0	11,933	0
Urban areas #1 (Orange)	0	0	0	49,938	0	13,520
Semiarid soil (Yellow)	0	0	0	16,073	53,607	0
Urban areas #2 (Green)	0	3,523	0	0	0	40,579

labeling of the classes. This is also confirmed by the confusion matrix [122] displayed in Table 5.13. This experiment reveals that our k-means classifier is very similar to the one available in the commercial (ENVI) software.

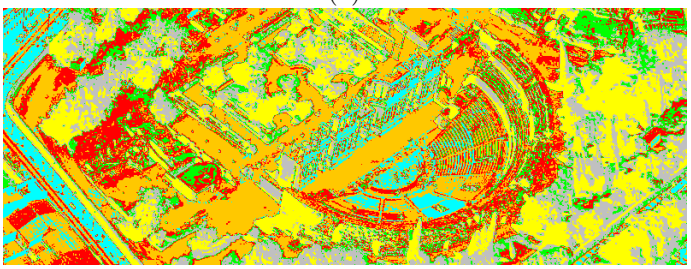
5.2.4.2 Experiment 2: Validation of the ISODATA algorithm with spatial post-processing

In this second experiment, the satellite-based image taken of Mérida, Spain [see Fig. 5.20(a)], offers a high spatial resolution of approximately 1.2 meters per pixel. The image was collected over the Roman Theater of Mérida (dating back to 16 - 15 BC, but renovated later on). The theatre is located in one of the most extensive archaeological sites in Spain. It was declared a World Heritage Site by UNESCO in 1993. The theatre was located at the edge of the Roman city near the walls. The grandstand consists of a semicircular seating area (cavea), with a capacity for 6,000 spectators eventually divided into three areas: the lowest tier called the *ima cavea* (22 rows), the medium tier called the *media* (5 rows), and a top tier called the *summa*, this one in less good condition. The Roman theater is the most visited monument in the city, and its festival classic theater is performed for the first time in 1933 and still continues today.

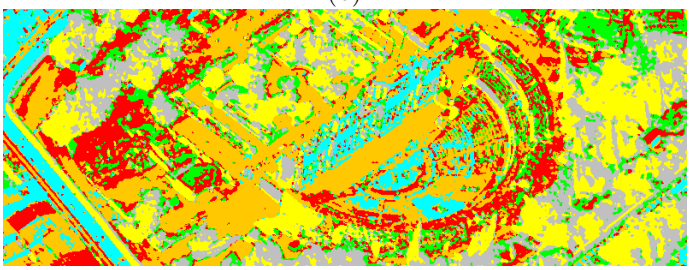
This monument has been chosen as an example of remotely sensed archeology, and we have decided to enhance a view offered by Google Maps to improve the visualization of the structure and the scale of this relevant monument for the region. Figs. 5.20(b) and 5.20(c) respectively show the unsupervised classification results obtained from our processing chain using the ISODATA algorithm and the classification obtained from applying spatial post-processing (using a processing window of 3×3 pixels) over the classification result obtained from Figs. 5.20(b). Fig. 5.20(d) and 5.20(e) respectively show the unsupervised classification of the ISODATA algorithm and the classification obtained from applying spatial post-processing using the same parameters and the ones obtained from our implementation.

In this experiment, a fixed number of $c = 6$ and a window size of 3×3 pixels have been considered. Also, although the color class labels for the implementations are different, the classification map provided by our implementation (without spatial post-processing) and the ones obtained using ENVI are very similar. In both cases, the algorithm parameters have been set to exactly the same values. Table 5.14

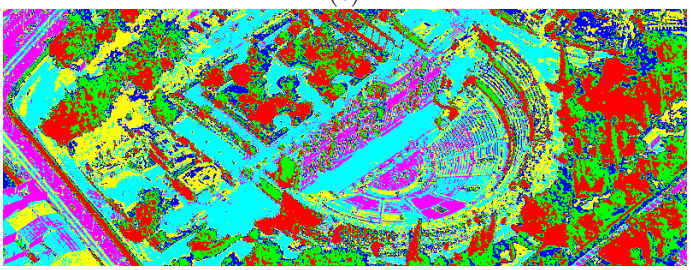
(a)



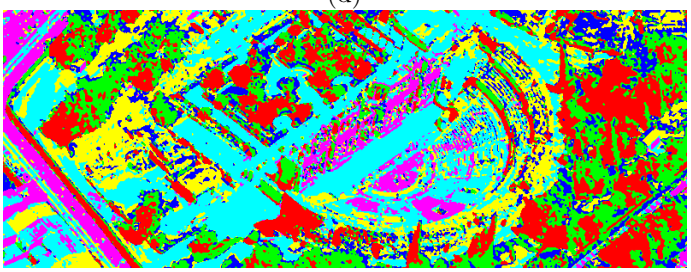
(b)



(c)



(d)



(e)

Figure 5.20: (a) Satellite image collected over the Roman city of Mérida, Spain. (b) Classification using our processing chain with ISODATA. (c) Classification using ENVI's ISODATA. (d) Classification using our processing chain with ISODATA with spatial post-processing. (e) Classification using ENVI's ISODATA with spatial post-processing.

Table 5.14: Percentage of agreement after comparing the classification map in Fig. 5.20(b), produced by our tool (with ISODATA), with the classification map in Fig. 5.20(d), produced by ENVI, and after comparing the map in Fig. 5.20(c) produced by our tool (with spatial post-processing) with the classification map in Fig. 5.20(e) produced by ENVI (also with spatial post-processing).

Clustering algorithm	Sand (Red)	Structure (Cyan)	Trees (Grey)	Shadows (Yellow)	Rocks (Green)	Pavement (Orange)	Overall agreement
ISODATA	87.90	89.23	93.64	100.00	84.56	91.33	91.11
ISODATA with spatial post-processing	79.48	91.09	87.02	98.42	79.32	89.36	87.45

Table 5.15: Confusion matrix obtained after comparing the classification map in Fig. 5.20(b), produced by our system (with ISODATA) with the map in Fig. 5.20(d), produced by ENVI.

Class	Sand (Yellow)	Structure (Magenta)	Trees (Grey)	Shadows (Red)	Rocks (Blue)	Pavement (Orange)
Sand (Red)	35,986	0	0	0	0	5,248
Structure (Cyan)	0	17,928	0	0	0	0
Trees (Grey)	0	0	40,335	0	5,187	0
Shadows (Yellow)	0	0	2,739	45,373	0	0
Rocks (Green)	4,951	0	0	0	25,947	0
Pavement (Orange)	0	2,088	0	0	0	55,298

reports the classification percentages of agreement measured after comparing our ISODATA classification maps with and without to apply a spatial post-processing. As shown by Table 5.14, the agreement between the maps is always very high (about a 90%). The confusion matrices for ISODATA and ISODATA with a spatial post-processing are respectively provided in Tables 5.15 and 5.16. With this example, the potential of our proposed tool for perform classification with and without applying a spatial post-processing over a satellite image is shown.

5.2.4.3 Experiment 3: Performance of the client-server architecture

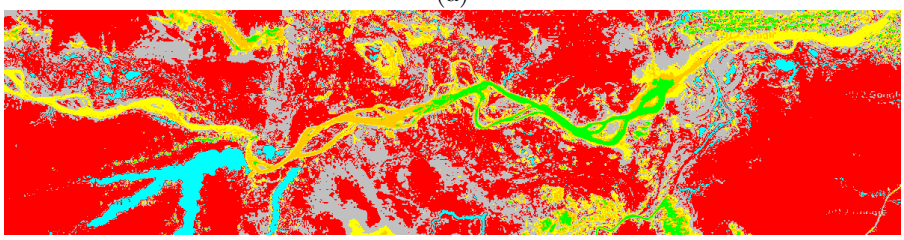
This third test assesses the performance of the system in terms of computational cost and processing time. The experiment consists of two main parts. First, we evaluate the performance of the proposed application using a web-based server and three different client configurations (medium, high, and very high quality of Internet access). Then, we discuss the impact of using a local compute server or a remote server in the experiments. A satellite image is collected over the Amazon river in South-America, using different zoom levels to illustrate the impact of the parameters from the two previous experiments when performing the image capture (snapshot) in the Google Maps engine. The comparison is simply intended to illustrate the computational advantages that can be gained by using a remote compute server with regards to the case in which local processing is performed in a desktop computer.

Fig. 5.21(a) shows the image to be processed at a given zoom level. Fig. 5.21(b) shows the obtained

Table 5.16: Confusion matrix obtained after comparing the classification map in Fig. 5.20(c), produced by our system (with ISODATA plus spatial post-processing), with the classification map in Fig. 5.20(e), produced by ENVI.

Class	Sand (Yellow)	Structure (Magenta)	Trees (Grey)	Shadows (Red)	Rocks (Blue)	Pavement (Orange)
Sand (Red)	32,503	77	336	137	494	4,935
Structure (Cyan)	515	17,547	122	30	246	924
Trees (Grey)	186	20	37,425	198	4,524	51
Shadows (Yellow)	60	12	3,703	46,062	199	26
Rocks (Green)	6,189	57	1,261	287	21,590	751
Pavement (Orange)	1,440	1,550	158	88	165	56,132

(a)



(b)

Figure 5.21: (a) Satellite image collected over the Amazon river in South-America. (b) Classification using our processing chain with k-means.

processing result, using the k-means algorithm with $c = 6$ classes and without spatial post-processing. Table 5.17 summarizes the different zoom levels that will be considered in this scene. The table indicates the spatial resolution of each image (depending on the considered zoom level) and the size in MB of the captured image at each zoom level. Before describing the obtained timing results, we summarize in Table 5.18 the client-server configurations adopted in our experimental evaluation. The table describes three different scenarios given by different processing speeds, where the "Home PC" scenario can be considered as medium quality, the "Work PC" scenario as high quality, and the "UEX LAN" scenario (here, the abbreviation "UEX" refers to "University of Extremadura"), as very high quality (in terms of bandwidth transmission). All the previous scenarios make use of a remote server called "CETA-Ciemat".

Fig. 5.22 compares the timing results obtained in the three considered scenarios (Home PC, Work PC and UEX LAN) with and without spatial post-processing. As Fig. 5.22 suggests, processing time

Table 5.17: Different zoom levels considered in the experiments with a satellite image over the Amazon river in South America and available in Google Maps.

Considered zoom level	Image size (MB)	Dimensions (pixels)
Zoom 1	0.20	600×150
Zoom 2	0.50	1200×300
Zoom 3	2.59	2400×600
Zoom 4	10.00	4800×1200

Table 5.18: Different scenarios used in the computational performance evaluation of our system.

Settings	External LAN (Home PC)	External LAN (Work PC)	UEX LAN	Server CETA-Ciemat LAN
CPU	Intel Core 2 Duo, 3 GHz	Intel Core 2 Duo, 3 GHz	Intel Core 2 2.53 GHz	2×QuadCore Intel Xeon 2.27 GHz
Main memory	6GB	4 GB	4 GB	12 GB
Client browser	Google Chrome	Google Chrome	Google Chrome	Google Chrome
Operating system	Windows Vista Business 32 Bits	Windows Vista Business 32 Bits	Windows 7 Professional 64 Bits	CentOS 6.2
Download speed	36,000 Kbps	36,000 Kbps	13,258 Kbps	100 Mbps
Upstream speed	1,100 Kbps	1,100 Kbps	10,593 Kbps	100 Mbps

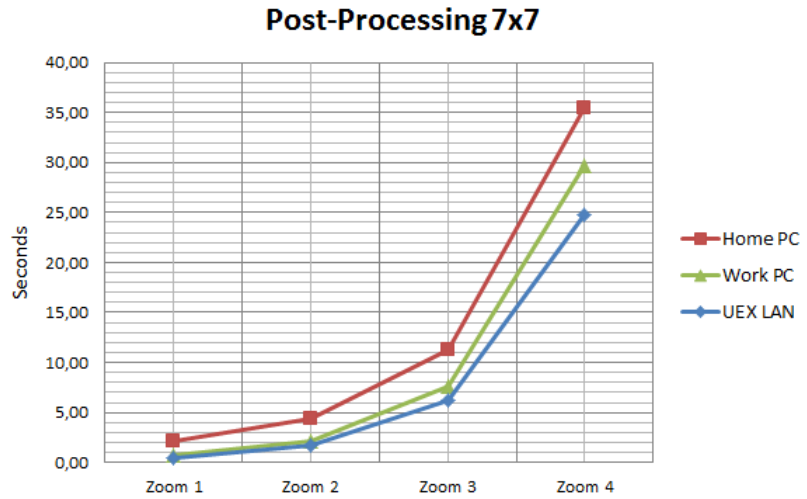
depends on the type of Internet connection. Secondly, processing time depends largely on image size. As indicated in Fig. 5.22, this time will grow significantly as the zoom level becomes more detailed, in particular for the medium-quality scenario. Finally, it is also clear from Fig. 5.22 that spatial post-processing increases execution time but not significantly, regardless of the quality of the connection available.

In all cases, we can observe how processing complexity increases with image size (i.e., with the considered zoom level). Spatial post-processing does not significantly increase computational complexity (despite the relatively large size of the spatial window adopted, with 7×7 pixels). Another important observation is that, as the quality of the used configuration increases, processing times significantly decreases, though the proposed system performs suitably in all cases. This fact illustrates the portability of the proposed system to different quality configurations.

Finally, a comparison is made between the processing time invested by the proposed system to perform the same image processing task described if the compute server is implemented locally versus the developments in which the compute server is run on a remote machine. Fig. 5.23 shows the processing times measured for the considered image processing scenario when the compute server is executed locally (in "Home PC") and with a remote server system (using the "Server CETA-Ciemat").

After comparing Fig. 5.23(a) and 5.23(b), the significant improvements achieved by implementing the compute server as a remote server can be observed, which reduces the processing times significantly. These improvements are detailed below: 1) if the compute server is implemented locally, the processing times are always much higher than if using a remote server, even if the local processing removes communication times over the network, as all the work is done on the same machine; and 2) the larger the

(a)



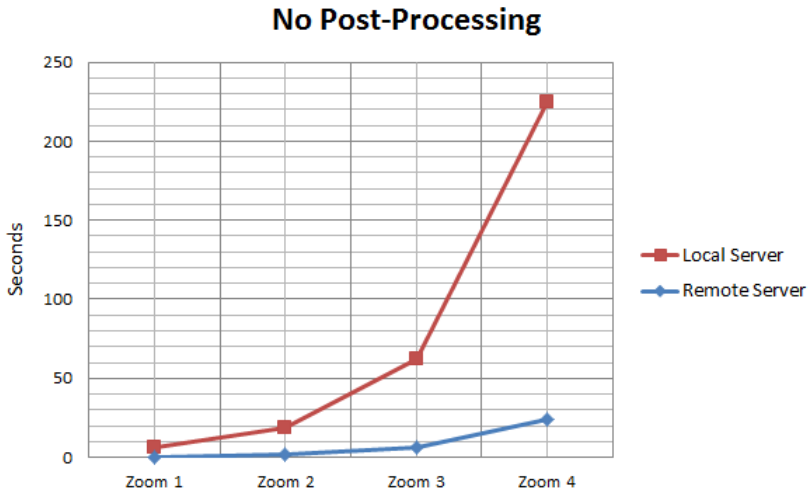
(b)

Figure 5.22: Comparison of the timing results obtained in the three considered scenarios (Home PC, Work PC and UEX LAN) with and without spatial post-processing.

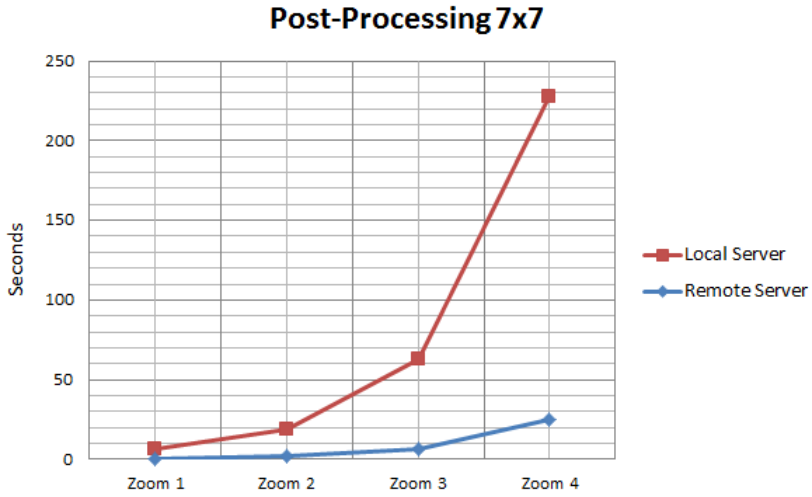
image size, the higher the times measured for the local processing while the remote processing times do not increase so significantly. This observation demonstrates that the use of a dedicated (remote) compute server, offers important advantages from the viewpoint of the computational efficiency of the considered application. In this case, a LAN connection is recommended in order to keep the communication times within reasonable levels.

5.2.5 Summary and future research lines

This subchapter has described a web-based system for computationally efficient processing of satellite/airborne images. The system, developed with the Google Maps applications programming interface (API), incorporates functionalities such as unsupervised classification of image portions selected by the user (at the desired zoom level) using the k-means and ISODATA clustering algorithms, followed by spatial post-processing. Most importantly, the processing of satellite/airborne images is conducted by means of a centralized server which receives the image to be processed, performs the analysis efficiently,



(a)



(b)

Figure 5.23: Comparison of the timing results obtained by the proposed system when the compute server is implemented as a local machine or as a (remote) server.

and returns the classification result to the end-user. This represents an improvement over a previous desktop application presented in [113] and in the previous subchapter of this thesis. The system has been implemented using a modular design which allows for the future incorporation of full multi-GPU functionality (using OpenCL implementations) in order to expand its processing capabilities. This would require a further evaluation of the possibility to perform the computation in distributed fashion, accompanied of a comparison in terms of energy consumption to determine the efficiency of the used server setups.

Our experimental results, conducted by comparing the obtained classification results with those provided by commercial products such as the popular ENVI software package, reveal that the proposed web-based tool provides classification maps with high similarity in relation to those provided by ENVI for the same satellite/airborne imagery, but with the possibility to perform the classification of any image portion available in the Google Maps engine. However, in future developments we are planing

on using the overlays available in Google Maps API to superimpose the outcome of the classification process (possibly with different transparency levels) on the original satellite/airborne imagery provided by Google Maps.

In the future, our optimal scenario would be to validate the classification accuracies that can be achieved by the proposed system in comparison to some reference data or other supervised classification methods. However, reference data are very difficult to obtain in practice and we could not obtain such information registered to the Google Maps data used by the proposed system. Hence, in this work we decided to use the agreement with the classification results provided by the same algorithms implemented in a widely used and highly consolidated software tool such as ENVI as an indicator of the performance of our system in classification tasks. The obtained agreements were very high, indicating that our system can provide classification results which are similar to those already provided by commercial software.

In addition, we also plan to incorporate other advanced classifiers to the proposed web-based system, such as Random Forests (RFs) and Support Vector Machines (SVMs). We are also considering the inclusion of alternative data sources without watermarks (e.g., Bing Maps) following a highly modular design. We are also planning on separating the web server and the processing server functionalities in order to make our design more modular and also to guarantee security (it would be highly desirable to have a design in which the compute server only has access to the web server and cannot be externally accessed). On the other hand, we would like to extend the developed tool with the incorporation of CBIR functionalities. For that purpose, the strategy will be based on a query system linked to feature extraction from an image repository (such as Google Maps or other maps server). The retrieved features (which will comprise shape descriptors, texture features, etc.) will be stored on a database of features and used to compare the feature vector of the input query with those recorded by means of a similarity function. This facility will provide a result to the end-user in the form of image portions (across different locations) that have enough similarity in relation to the features of the input query. An immediate research focus would thus be the integration of a CBIR architecture in our system.

Chapter 6

Conclusions and Future Work

This thesis presented novel techniques and methodologies for different problems related with the analysis of remotely sensed images, with the ultimate goal to incorporate our new developments in advanced information extraction and classification systems for efficient remote sensing data retrieval and interpretation in the context of remote Earth observation. On one hand, our proposed methodologies for classification of images with low spectral resolution allow us to expand the dimensionality of these scenes in order to process them using strategies normally reserved for scenes with higher spectral dimensionality. On the other hand, for the processing of scenes with high spectral resolution we have focused on the problem of spectral unmixing, studying in detail several approaches in the literature and achieving efficient implementations on GPU architectures, providing real-time processing for a complete unmixing chain. Also, we have achieved for the first time in the literature, real-time processing results for the same problem on multi-core platforms. The main achievements of this thesis can be more specifically summarized as follows:

- First and foremost, the problems related with the classification of images with only a few spectral channels have been addressed. In order to address this challenge, strategies based on kernel feature extraction have been adopted to increase the spectral information of the scenes and also, to include the spatial information by means of morphological characterization, which allowed us to improve the classification accuracies. In the considered data sets, these results were superior to those provided by EMAPs (built on the multispectral data), and even to those obtained using the full hyperspectral information with hundreds of spectral channels.

- Second, the unmixing problem has been addressed in the context of hyperspectral images, developing different algorithms to exploit specialized hardware systems such as FPGAs, GPUs and cloud computing platforms. In such a way, alternatives to clusters of computers have been proposed, which are very difficult to adapt to onboard processing requirements introduced by applications with real-time constraints such as wild land fire tracking, biological threat detection, monitoring of oil spills and other types of chemical contamination. These alternatives are oriented to professional environments such as:

  - *Cloud systems*, for which a complete unmixing chain made up of the following steps: 1) dimensionality reduction; 2) automatic endmember identification; and 3) fully constrained abundance estimation has been integrated with the Web Coverage Processing Service (WCPS), an image processing framework developed at NASA that can help quickly process large

amounts of data and deliver finished products to the end-user at an extremely low cost. The experimental results with a hyperspectral scene collected over the Okavango Basin in Botswana suggest that our newly developed approach can be integrated on global missions or missions with different resolutions and sensor capabilities, exploiting the potential of spectral unmixing for improved interpretation of these images on a computationally efficient way.

- *FPGA devices*, for which the impact of different optimizations (Gram-Schmidt orthogonalization, operations in fixed point, etc.) have been investigated on an algorithm for automatic target identification in hyperspectral images. For this purpose, we have designed a methodology to facilitate the implementation of algorithms using a high-level perspective from Matlab or C/C++ code, where the proposed approach allows us to preliminarily test the suitability of a certain algorithm for hardware implementation. This approach can report frequencies above 200 MHz without compromising target detection accuracy.

- *Multi-core and GPU platforms*, for which we have implemented computationally efficient implementations of a full hyperspectral unmixing using CUDA + cuBLAS library and API OpenMP + Intel Math Kernel Library (MKL), respectively. We have compared both architectures in terms of performance, cost and mission payload considerations based on the results obtained in two different scenarios using hyperspectral data collected by NASA's AVIRIS. With these efficient implementations, both architectures can resolve the spectral unmixing problem in real-time, being the first time that this kind of assessment is conducted for multi-core systems and achieving real-time performance for a complete unmixing chain previously reported in the literature. It is important to emphasize that GPUs compared with multi-core processors present a better performance on processing results but the power consumption of GPUs is quite high. In this regard, multi-core platforms can be a good option by means of the energy consumption of the chain implemented and it is expected that systems with hundreds of cores will be soon available, thus offering the possibility to replace many-core systems such as GPUs as the default platforms for high performance computing for onboard processing of remote sensing data in many applications.

- Finally, two systems able to provide fast, scalable and advanced data processing solutions have been developed for remotely sensed imagery. Both systems are able to perform classification of any image portion available in Google Maps engine using a processing chain based on three steps: 1) pre-processing; 2) data analysis and interpretation; and 3) post-processing. This is enforced by the exploitation of graphics coprocessors (desktop system) or a remote server (web-based system) for efficient data processing. The proposed systems have been demonstrated with remotely sensed data with different spatial and spectral resolutions, providing classification maps with high similarity in relation to those provided by another commercial tool (Research Systems ENVI) for the same satellite/airborne imagery. However, the web-based system has improved the processing of large images at different zoom levels without the necessity of creating a mosaic in the image acquisition, which was a main problem in the desktop version, using a remote server. Also, it is possible to combine classes with high similarity in the classification process. With these important improvements and the flexibility to extend to other map servers and software platforms, the web-based version can be a good candidate to incorporate more techniques such as CBIR functionalities and high performance computing for the processing of large images.

In addition to the systems and novel techniques proposed, we are currently working on the integration of all aforementioned above into the web system (see Fig. 6.1) to provide a unified tool for improved classification and interpretation of remotely sensed images. For this purpose, we plan to use higher level strategies to derive spatial features such as object-based image analysis and knowledge-based methods. In future work, we would also like to extend the developed tool, on the one hand, with the incorporation of CBIR functionalities based on queries of features extracted from an image repository (such as Google Maps, a hyperspectral repository currently under development in the HyperComp research group) or other map servers to be stored on a database of features and used to compare the feature vector of the input query with those recorded by means of a similarity function, which will provide a result to the end-user in the form of image portions that have enough similarity in relation to the features of the input query. On the other hand, we are also investigating the use of OpenCL as a computing standard to implement efficient algorithms on GPU devices and other specialized hardware accelerators (susceptible of being used onboard the sensor platform) such as digital signal processors (DSPs) or field programmable gate arrays (FPGAs).

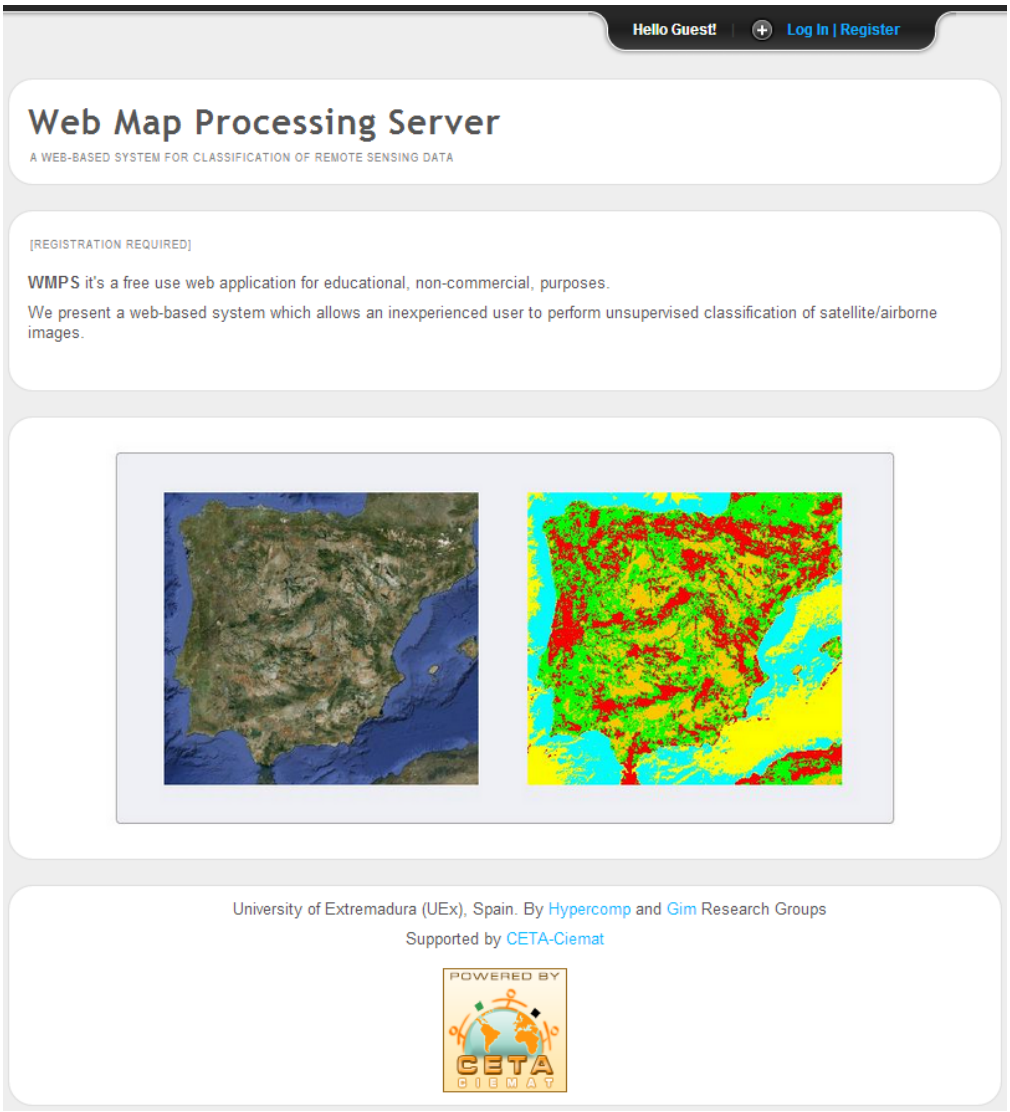


Figure 6.1: Web application interface to perform advanced classification, interpretation and content retrieval tasks using large-scale data repositories of remote sensing data. (available online: http://hypergim.ceta-ciemat.es)

Apendix A

Publications

The results of this thesis work have been published in several international journal papers, peer-reviewed international conference papers and peer-reviewed national conference papers. Specifically, the candidate has co-authored 8 journal citation reports (JCR) papers, 9 peer-review international conference papers and 1 peer-review national conference paper directly related with this thesis work. This thesis work have been partially supported by the following programs, funds and resources:

1. European Communitys Marie Curie Research Training Networks Programme under reference MRTN-CT-2006-035927, Hyperspectral Imaging Network (HYPER-I-NET).
2. HYPERCOMP/EODIX proyect by the Spanish Ministry of Science and Innovation, reference AYA2008-05965-C04-02.
3. CEOS-SPAIN project by the Spanish Ministry of Science and Innovation, reference AYA2011-29334-C02-02.
4. DREAMS project by the Spanish Ministry of Science and Innnovation, reference TEC2011-28666-C04-04.
5. Spanish Government Board (National Research Projects) and the European Union (FEDER founds) by means of the grant reference TIN2008-03063.
6. Basic support to research groups by the "Junta de Extremadura" (Spain), reference GR10035.
7. Icelandic Research Fund.
8. Computing resources of Extremadura Research for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). The CETA-CIEMAT belongs to the Spanish Ministry of Science and Innovation.

The candidate has been a pre-doctoral research associate the Department of Technology of Computers and Communications at the University of Extremadura, forming part of the research group "Hyperspectral Computing Laboratory" (HyperComp) of the University of Extremadura (Spain), jointly with the Faculty of Electrical and Computer engineering, at the University of Iceland. In the following, we outline the content of each of the publications achieved by the candidate, providing also a short description of the journal or workshop where it was presented and also a description of the specific contribution of the candidate in each publication.

A.1 International journal papers

1. **S. Bernabé**, P. R. Marpu, A. Plaza, M. Dalla Mura and J. A. Benediktsson, "Spectral-Spatial Classification of Multispectral Images Using Kernel Feature Space Representation", *IEEE Geoscience and Remote Sensing Letters*, accepted for publication, 2013 [JCR(2012)=1.823]. This paper will be published in the journal *IEEE Geoscience and Remote Sensing Letters* which is a very important journal in the second quartile of the field "Remote Sensing". The paper shows a novel methodology to incorporate spectral and spatial information into the classification of multispectral images, described partially in the subchapter 2 of this thesis. Specifically, we have adopted strategies based on kernel feature extraction to incorporate more spectral information and also, to include more spatial information by means of morphological characterization. The contribution of the candidate to this paper was the design of the proposed methodology and also the development of all experiments presented in the paper. This work was carried out in collaboration with experts from the Masdar Institute of Science and Technology, Abu Dhabi (UAE), University of Iceland and GIPSA-Lab, Grenoble (France).

2. A. Ferrán, **S. Bernabé**, P. G. Rodríguez and A. Plaza, "A Web-Based System for Classification of Remote Sensing Data", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 4, pp. 1934–1948, August 2013 [JCR(2012)=2.874]. 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 field "Remote Sensing". The paper shows a web application to the development of advanced information extraction and classification for efficient remote sensing data retrieval and interpretation, described partially in the subchapter 5.2 of this thesis. The developed system can be used with large data repositories such as Google Maps™. The contribution of the candidate to this paper was the design and implementation of the processing chain implemented in C language to perform information extraction and classification of remotely sensed images and also the development of all experiments presented in the paper.

3. **S. Bernabé**, S. Sánchez, A. Plaza, S. López, 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*, vol. 6, no. 3, pp. 1386–1398, June 2013 [JCR(2012)=2.874]. This paper was published in the journal *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* which is a journal in the first quartile of the field "Remote Sensing". The paper shows a performance comparison of different full unmixing chains on domestic and professional GPU and Multi-core platforms, described partially in the subchapter 4 of this thesis. The processing chain is composed for the following methods: VD, OSP-GS and UCLS. The contribution of the candidate to this paper were to provide the OSP-GS and UCLS methods in serial, multi-core and GPU versions and VD method in multi-core version for inclusion in the full unmixing chain, achieving the faster unmixing chain on GPU and real-time performance of a full unmixing chain on a domestic multi-core platform for the first time in the literature. This work was carried out in collaboration with experts from the University of Las Palmas de Gran Canaria (Spain) and University of Iceland.

4. P. Cappelaere, S. Sánchez, **S. Bernabé**, A. Scuri, D. Mandl and A. Plaza, "Cloud Implementation of a Full Hyperspectral Unmixing Chain within the NASA Web Coverage Processing Service for EO-1", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 1–11, April 2013 [JCR(2012)=2.874]. This paper was published in the journal *IEEE*

Journal of Selected Topics in Applied Earth Observations and Remote Sensing which is a journal in the first quartile of the field "Remote Sensing". The paper shows an implementation study on a full unmixing chain applied to images with high spectral dimension. The unmixing chain will be available online with the Web Coverage Processing Service (WCPS), an image processing framework that can run on the cloud, as part of the NASA SensorWeb suite of web services, described partially in the subchapter 3.3 of this thesis. The processing chain consists of the following steps: dimensionality reduction, endmember extraction and abundance estimation. The contribution of the candidate to this paper was to provide the FCLSU method in serial version to estimate the fractional abundance of each endmember in each pixel of an image. The full unmixing chain was adapted to the cloud environment. This work was carried out in collaboration with experts from the NASA/Goddard Space Flight Center and Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil.

5. A Remón, S. Sánchez, **S Bernabé**, E. S. Quintana and A. Plaza, "Performance versus Energy Consumption of Hyperspectral Unmixing Algorithms on Multi-Core Platforms", *EURASIP Journal on Advances in Signal Processing*, vol. 68, pp. 1–15, April 2013 [JCR(2012)=0.807]. This paper was published in the journal *EURASIP Journal on Advances in Signal Processing* which is an important journal in the third quartile of the field "Engineering, electrical & electronic". The paper shows a study relating the performance obtained by running different hyperspectral processing chains with energy consumption in multi-core processors. For this purpose, we have addressed hyperspectral imaging via spectral unmixing composed by three stages: (i) the estimation of the number of endmembers, (ii) the identification of a collection of these, and (iii) the estimation of the fractional abundances, using kernels from highly tuned linear algebra libraries and OpenMP directives on a platform equipped with 48 AMD cores. The contribution of the candidate to this paper was to provide the OSP-GS method in serial and GPU versions, in order to carry out the study. This work was carried out in collaboration with experts from the University of Jaume I, Castellón (Spain).

6. **S Bernabé**, S. López, A. Plaza and R. Sarmiento, "GPU Implementation of an Automatic Target Detection and Classification Algorithm for Hyperspectral Image Analysis", *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 2, pp. 221–225, March 2013 [JCR(2012)=1.823]. This paper was published in the journal *IEEE Geoscience and Remote Sensing Letters* which is a journal in the second quartile of the field "Remote Sensing". The paper shows several optimizations for accelerating the computational performance of ATDCA algorithm, described partially in the subchapter 3.1.3 of this thesis. The first one focuses on the use of the Gram-Schmidt orthogonalization method and the second one is focused on the development of the algorithm on GPUs, including shared memory, taking advantage of the performance of operations in single precision floating point and providing coalesced accesses to memory that lead to very significant speedup factors. With these approaches, we have achieved real-time performance of ATDCA for the first time in the literature using different hyperspectral scenes. The contribution of the candidate to this paper was the design and implementation of the method and also the development of all experiments presented in the paper. This work was carried out in collaboration with experts from the University of Las Palmas de Gran Canaria (Spain).

7. **S Bernabé**, A. Plaza, P. R. Marpu and J. A. Benediktsson, "A New Parallel Tool for Classification of Remotely Sensed Imagery", *Computers & Geosciences*, vol. 46, pp. 208–218, September 2012 [JCR(2012)=1.834]. This paper was published in the journal *Computers & Geosciences* which

is an important journal in the second quartile of the field "Computer science, interdisciplinary applications". The paper shows a desktop application to perform classification of remotely sensed images in a computationally efficient form through the exploitation of the fine granularity of parallelism offered by GPU architectures, described partially in the subchapter 5.1 of this thesis. The processing chain is based on three main parts: (1) pre-processing, performed using morphological profiles; (2) classification, which can be performed in supervised or unsupervised fashion; and (3) post-processing. This processing chain has been integrated into the tool to allow processing of satellite images available from Google Maps engine and developed using Java and the SwingX-WS library. A general framework for parallel implementation of the processing chain has also been developed and specifically tested on GPUs. The contribution of the candidate to this paper was the design and implementation of the tool with all the methods and also the development of all experiments presented in the paper.

A.2 International journal papers submitted (under review)

1. J. Sevilla, **S Bernabé** and A. Plaza, "Unmixing-Based Content Retrieval System for Remotely Sensed Hyperspectral Imagery on GPUs", *Journal of Supercomputing*, under review, 2013 [JCR(2012)=0.917]. This is a very important journal in the second quartile of the field "Computer science, hardware & architecture". The paper shows a new unmixing-based retrieval system for remotely sensed hyperspectral imagery. It particularly focuses on the design of the system and its efficient implementation on GPUs. The contribution of the candidate to this paper was to provide the ATDCA-GS method in serial and GPU versions and also support in the development of all experiments presented in the paper.

A.3 Peer-reviewed international conference papers

1. J. Sevilla, **S Bernabé** and A. Plaza, "Unmixing-Based Retrieval System for Remotely Sensed Hyperspectral Imagery on GPUs", *13th International Conference on Computational and Mathematical Methods in Science and Engineering*, Almería, Spain, 2013. This work was presented in an oral presentation in the workshop *CMMSE 2013 - Minisymposium: HPC*. This minisymposium 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 presented a new unmixing-based image retrieval system for remote sensed hyperspectral imagery. In order to deal with the computational cost of executing spectral unmixing algorithms, we have used GPUs. The contribution of the candidate to this paper was to provide the ATDCA-GS method in serial and GPU versions, in order to carry out the experiments.

2. **S Bernabé**, P. R. Marpu, A. Plaza and J. A. Benediktsson, "Spectral Unmixing of Multispectral Satellite Images with Dimensionality Expansion Using Morphological Profiles", *SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference*, San Diego, CA, 2012. This work was presented in an oral presentation in the workshop *SPIE Optics and Photonics*. This workshop is often attended by more than thousands of worldwide researches in the remote sensing field, and it is one of the most international important workshop in remote sensing. In this work we presented a new framework for spectral unmixing of multispectral remote sensing images with limited spectral resolution. First, the image is expanded using KPCA + EMAP

and then, unmixing is performed on the expanded data and the corresponding stack of abundance maps are used for classification to validate the effectiveness of the unmixing strategy.

3. A. Ferrán, **S Bernabé**, P. G. Rodríguez and A. Plaza, "A New Web-Based System for Unsupervised Classification of Satellite Images from the Google Maps Engine", *SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference*, San Diego, CA, 2012. This work was presented in an oral presentation in the workshop *SPIE Optics and Photonics*. In this work we presented a new web-based system (which represents a follow-up of our previous work in [113]) that allows an inexperienced user to perform an unsupervised classification of satellite images obtained via Google Maps^{TM} using the API. The contribution of the candidate was the design and implementation of the unsupervised methods and also the development of all experiments presented in the paper. The results showed during this conference are also presented partially in this thesis work.

4. J. Sevilla, **S Bernabé**, A. Plaza and P. G. Rodríguez, "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, CA, 2012. This work was presented in an oral presentation in the workshop *SPIE Optics and Photonics*. In this work we presented the first step towards the development of a digital repository for remotely sensed hyperspectral data. The proposed system allows searching for images on the spectral unmixing information using both synthetic and real hyperspectral images stored in the database. The contribution of the candidate was support in the design of the repository and with the integration of the different unmixing algorithms.

5. **S Bernabé**, A. Plaza, S. López and R. Sarmiento, "Parallel Implementation of a Hyperspectral Unmixing Chain: Graphics Processing Units versus Multi-Core Processors", *IEEE Geoscience and Remote Sensing Symposium (IGARSS'12)*, Munich, Germany, 2012. This work was presented in an oral presentation in the workshop *IEEE IGARSS* in 2012. This workshop is often attended by thousands of worldwide researches in the remote sensing field, and it is the most international important workshop in the remote sensing field. In this work we presented two efficient implementations of a full hyperspectral unmixing chain on two different kinds of high performance computing architectures inter-compared in the context of hyperspectral images: graphics processing units (GPUs) and multi-core processors. The results showed during this conference are also presented partially in this thesis work.

6. **S Bernabé** and A. Plaza, "Commodity Cluster-Based Parallel Implementation of an Automatic Target Generation Process for Hyperspectral Image Analysis", *IEEE International Conference on Parallel and Distributed Systems*, Tainan, Taiwan, 2011. This work was presented in an oral presentation in the workshop *IEEE ICPADS* in 2011. This workshop is often attended by thousands of researchers with results on all aspects of parallel and distributed systems. In this work we presented a new parallel version of ATDCA algorithm by incorporating a new method for calculating the orthogonal projection process based using the Gram-Schmidt method. This algorithm is implemented on a multi-core cluster system made up of sixteen nodes, and quantitatively evaluated using hyperspectral data.

7. **S Bernabé**, S. López, A. Plaza, R. Sarmiento and P. G. Rodríguez, "FPGA Design of an Automatic Target Generation Process for Hyperspectral Image Analysis", *IEEE International Conference on*

Parallel and Distributed Systems, Tainan, Taiwan, 2011. This work was presented in an oral presentation in the workshop *IEEE ICPADS* in 2011. In this work we presented a methodology for algorithms which are codified in Matlab (or alternative C/C++) to obtain a register transfer level (RTL) description that can be ported to FPGAs. Also, we develop a quantitative and comparative study using two different FPGA architectures. Experimental results have been obtained in the context of hyperspectral data. The results showed during this conference are also presented partially in this thesis work.

8. **S Bernabé** and A. Plaza, "A New System to Perform Unsupervised and Supervised Classification of Satellite Images from Google Maps", *SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference*, San Diego, CA, 2010. This work was presented in an oral presentation in the workshop *SPIE Optics and Photonics*. In this work we presented a system which makes use of the SwingX-WS library to access to Google Maps engine, and incorporates functionalities such as unsupervised or supervised classification of image portions selected by the user, followed by spatial post-processing. The experimental results have been conducted by comparing the obtained classification results with those provided by commercial software. The results showed during this conference are also presented partially in this thesis work.

9. **S Bernabé** and A. Plaza, "A New Tool for Information Extraction and Mining from Satellite Imagery Available from Google Maps Engine", *3rd International Symposium on Recent Advances in Quantitative Remote Sensing (RAQRS)*, Valencia, Spain, 2010. This work was presented in a poster presentation in the workshop *RAQRS'10*. This workshop is often attended by hundreds of researchers in the remote sensing field. In this work we presented a new tool for information extraction and mining from satellite images available in Google Maps™. The tool comprise the possibility to perform unsupervised classification of image portions selected by the user, followed by spatial post-processing. The results showed during this conference are also presented partially in this thesis work.

A.4 Peer-reviewed national conference papers

1. **S Bernabé** and A. Plaza, "A New Tool for Classification on Satellite Images Available from Google Maps: Efficient Implementation in Graphics Processing Units", *XXII Jornadas de Paralelismo*, La Laguna, Spain, 2011. This work was presented in an oral presentation in the workshop *JP2011*. This workshop is often attended by hundreds of Spanish researchers with interest on all aspects of parallel and distributed systems. In this work is presented a new parallel implementation of the k-means unsupervised clustering algorithm for GPUs using satellite images obtained from Google Maps engine. Also, we have analyzed the consensus or agreement in the classification achieved by our implementation and an alternative implementation of the algorithm available in commercial software. The results showed during this conference are also presented partially in this thesis work.

Bibliography

[1] P. E. Ardanuy, D. Han, and V. V. Salomonson. The moderate resolution imaging spectrometer (MODIS) science and data system requirements. *IEEE Transactions on Geoscience and Remote Sensing*, 29(1):75–88, 1991. [Cited in pags. xx, 39 and 45]

[2] R. A. Schowengerdt. *Remote sensing: models and methods for image processing, 2nd ed.* Academic Press: San Diego, USA, 1997. [Cited in pags. 1 and 89]

[3] D. A. Landgrebe. *Signal theory methods in multispectral remote sensing.* John Wiley & Sons: New Jersey, USA, 2003. [Cited in pags. 1, 67 and 89]

[4] X. Jia, J. A. Richards, and D. E. Ricken. *Remote sensing digital image analysis: an introduction.* Springer-Verlag Berlin, 1999. [Cited in pags. 1 and 67]

[5] J. Serra. *Image analysis and mathematical morphology.* Academic Press: London, 1982. [Cited in pags. 3, 68 and 70]

[6] P. Soille. *Morphological image analysis: principles and applications.* Springer-Verlag Berlin, 2003. [Cited in pags. 3, 68, 70 and 71]

[7] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J.C. Tilton, and G. Trianni. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113:110–122, 2009. [Cited in pags. 3, 52, 68 and 90]

[8] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3804–3814, 2008. [Cited in pags. 3, 9 and 90]

[9] E. Quiros, A. M. Felicisimo, and A. Cuartero. Testing multivariate adaptive regression splines (MARS) as a method of land cover classification of TERRA-ASTER satellite images. *Sensors*, 9(11):9011–9028, 2009. [Cited in pags. 3 and 90]

[10] A. Cuartero, A. M. Felicisimo, M. E. Polo, A. Caro, and P. G. Rodriguez. Positional accuracy analysis of satellite imagery by circular statistics. *Photogrammetric Engineering and Remote Sensing*, 76(11):1275–1286, 2010. [Cited in pags. 3 and 90]

[11] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery. Active learning methods for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 47(7):2218–2232, 2009. [Cited in pags. 3 and 90]

[12] S. Bernabe, P. R. Marpu, A. Plaza, and J. A. Benediktsson. Spectral unmixing of multispectral satellite images with dimensionality expansion using morphological profiles. *In Proceedings of SPIE, Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference*, 8514:1–8, 2012. [Cited in pag. 3]

[13] R. M. Stair and G. W. Reynolds. *Principles of information systems.* MIS Series, 2009. [Cited in pag. 3]

[14] J. A. O'Brien and G. M. Marakas. *Management information systems.* McGraw-Hill, 2006. [Cited in pag. 3]

[15] 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. 3 and 90]

[16] D. Landgrebe. Hyperspectral image data analysis. *IEEE Signal Processing Magazine*, 19(1):17–28, 2002. [Cited in pag. 3]

[17] N. Keshava and J. F. Mustard. Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57, 2002. [Cited in pags. 3, 38, 45 and 52]

[18] A. Plaza, J. Plaza, A. Paz, and S. Sanchez. Parallel hyperspectral image and signal processing. *IEEE Signal Processing Magazine*, 28(3):119–126, 2011. [Cited in pags. 3, 29, 38, 54 and 91]

[19] A. Plaza, Q. Du, J. M. Bioucas-Dias, X. Jia, and F. A. 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, 38 and 52]

[20] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza. Sparse unmixing of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 49(6):2014–2039, 2011. [Cited in pag. 3]

[21] S. Sanchez, 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 pags. 3, 20, 27, 50 and 58]

[22] P. Cappelaere, S. Sanchez, S. Bernabe, A. Scuri, D. Mandl, and A. Plaza. Cloud implementation of a full hyperspectral unmixing chain within the NASA web coverage processing service for E0-1. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2, Part. 1):408–418, 2013. [Cited in pag. 3]

[23] S. Bernabe, S. Sanchez, 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 pag. 3]

[24] A. Plaza, D. Valencia, J. Plaza, and P. Martinez. Commodity cluster-based parallel processing of hyperspectral Imagery. *Journal of Parallel and Distributed Computing*, 66(3):345–358, 2006. [Cited in pags. 3, 54, 68, 73, 74 and 91]

[25] S. Bernabe and A. Plaza. Commodity cluster-based parallel implementation of an automatic target generation process for hyperspectral image analysis. *In Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, pages 1038–1043, 2011. [Cited in pags. 3 and 54]

[26] J. B. Adams, M. O. Smith, and P. E. Johnson. Spectral mixture modeling: a new analysis of rock and soil types at the viking lander 1 site. *Journal of Geophysical Research*, 91(B8):8098–8112, 1986. [Cited in pags. 3 and 52]

[27] A. Plaza and C.-I Chang. Clusters versus FPGA for parallel processing of hyperspectral imagery. *International Journal of High Performance Computing Applications*, 22(4):366–385, 2008. [Cited in pags. 3, 29, 54 and 68]

[28] C. Gonzalez, D. Mozos, J. Resano, and A. Plaza. FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):374–388, 2012. [Cited in pags. 3, 50 and 54]

[29] S. Bernabe, S. Lopez, A. Plaza, R. Sarmiento, and P. G. Rodriguez. FPGA design of an automatic target generation process for hyperspectral image analysis. *In Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, pages 1010–1015, 2011. [Cited in pags. 3 and 54]

[30] A. Remon, S. Sanchez, A. Paz, E. S. Quintana-Orti, and A. Plaza. Real-time endmember extraction on multi-core processors. *IEEE Geoscience and Remote Sensing Letters*, 8(5):924–928, 2011. [Cited in pags. 3 and 54]

[31] S. Bernabe, A. Plaza, S. Lopez, and R. Sarmiento. Parallel implementation of a hyperspectral unmixing chain: graphics processing units versus multi-core processors. *In Proceedings of the IEEE Geoscience and Remote Sensing Symposium (IGARSS'12)*, pages 3463–3466, 2012. [Cited in pags. 3 and 54]

[32] A. Remon, S. Sanchez, S. Bernabe, E. S. Quintana-Orti, and A. Plaza. Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms. *EURASIP Journal on Advances in Signal Processing*, 2013(id:68):1–15, 2013. [Cited in pags. 3 and 54]

[33] J. Setoain, M. Prieto, C. Tenllado, and F. Tirado. GPU for parallel on-board hyperspectral image processing. *International Journal of High Performance Computing Applications*, 22(4):424–437, 2008. [Cited in pags. 3, 54 and 69]

[34] S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento. GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 10(2):221–225, 2013. [Cited in pags. 3 and 54]

[35] M. Pesaresi and J. A. Benediktsson. A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):309 –320, 2001. [Cited in pag. 9]

[36] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):480 – 491, 2005. [Cited in pags. 9, 11 and 68]

[37] J. Li, J. M. Bioucas-Dias, and A. Plaza. Spectral-spatial hyperspectral image segmentation using subspace multinomial logistic regression and markov random fields. *IEEE Transactions on Geoscience and Remote Sensing*, 50(3):809 –823, 2012. [Cited in pag. 9]

[38] B. Zhang, S. Li, X. Jia, L. Gao, and M. Peng. Adaptive markov random field approach for classification of hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 8(5):973 –977, 2011. [Cited in pag. 9]

[39] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson. SVM- and MRF-based method for accurate classification of hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 7(4):736 –740, 2010. [Cited in pag. 9]

[40] D. Tuia, F. Pacifici, M. Kanevski, and W. J. Emery. Classification of very high spatial resolution imagery using mathematical morphology and support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 47(11):3866 –3879, 2009. [Cited in pag. 9]

[41] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10):3747 –3762, 2010. [Cited in pags. 10 and 11]

[42] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. Extended profiles with morphological attribute filters for the analysis of hyperspectral data. *International Journal of Remote Sensing*, 31(22):5975–5991, 2010. [Cited in pags. 10 and 11]

[43] P. R. Marpu, M. Pedergnana, M. Dalla Mura, S. Peeters, J. A. Benediktsson, and L. Bruzzone. Classification of hyperspectral data using extended attribute profiles based on supervised and unsupervised feature extraction techniques. *International Journal of Image and Data Fusion*, 3(3):269–298, 2012. [Cited in pags. 10, 11, 12 and 17]

[44] B. Scholkopf, A. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. [Cited in pag. 10]

[45] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press Series, Cambridge, MA, 2002. [Cited in pag. 10]

[46] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. [Cited in pag. 10]

[47] J. A. Richards and X. Jia. *Remote sensing digital image analysis: an introduction*. Springer-Verlag Berlin, 2006. [Cited in pags. 10, 12, 89 and 105]

[48] P. R. Marpu, M. Pedergnana, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone. Automatic generation of standard deviation attribute profiles for spectral-spatial classification of remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 10(2):293–297, 2013. [Cited in pags. 12 and 15]

[49] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias, and J. A. Benediktsson. Generalized composite kernel framework for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 51(9):4816–4829, 2013. [Cited in pag. 17]

[50] C.-I Chang. *Hyperspectral imaging: techniques for spectral detection and classification*. Kluwer Academic/Plenum Publishers: New York, USA, 2003. [Cited in pags. 20, 24, 25, 58 and 68]

[51] H. Ren and C.-I Chang. Automatic spectral target recognition in hyperspectral imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1232–1249, 2003. [Cited in pags. 20 and 30]

[52] D. C. Heinz and C.-I Chang. Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(3):529–545, 2001. [Cited in pags. 20, 39, 44, 45 and 52]

[53] R. A. Neville, K. Staenz, T. Szeredi, J. Lefebvre, and P. Hauff. Automatic endmember extraction from hyperspectral data for mineral exploration. *In Proceedings of the 21st Canadian Symposium on Remote Sensing*, pages 1–8, 1999. [Cited in pags. 20 and 52]

[54] I. S. Reed and X. Yu. Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distrihution. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(10):1760–1770, 1990. [Cited in pag. 20]

[55] J. C. Harsanyi and C.-I Chang. Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):779–785, 1994. [Cited in pags. 20 and 52]

[56] G. Shaw and D. Manolakis. Signal processing for hyperspectral image exploitation. *IEEE Signal Processing Magazine*, 19(1):12–16, 2002. [Cited in pag. 20]

[57] A. Plaza and C.-I Chang. *High performance computing in remote sensing*. Taylor & Francis: Boca Raton, FL, 2007. [Cited in pags. 20, 25, 54, 68, 73 and 91]

[58] A. Paz and A. Plaza. Clusters versus GPUs for parallel target and anomaly detection in hyperspectral images. *EURASIP Journal on Advances in Signal Processing*, 2010(id:915639):1–18, 2010. [Cited in pags. 20, 22, 26, 27, 30 and 69]

[59] S. Lopez, P. Horstrand, G. M. Callico, J. F. Lopez, and R. Sarmiento. A low-computational-complexity algorithm for hyperspectral endmember extraction: modified vertex component analysis. *IEEE Geoscience and Remote Sensing Letters*, 9(3):502–506, 2012. [Cited in pag. 20]

[60] H. Yang, Q. Du, and G. Chen. Unsupervised hyperspectral band selection using graphics processing units. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):660–668, 2011. [Cited in pags. 20 and 54]

[61] E. Christophe, J. Michel, and J. Inglada. Remote sensing processing: from multicore to GPU. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):643–652, 2011. [Cited in pag. 20]

[62] 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*, 4(3):441–445, 2007. [Cited in pags. 22 and 73]

[63] 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. 26, 47 and 55]

[64] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford. Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of xilinx FPGAs. *In Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 1–6, 2006. [Cited in pag. 29]

[65] K. Compton and S Hauck. Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys*, 34(2):171–210, 2002. [Cited in pag. 29]

[66] R. Tessier and W. Burleson. Reconfigurable computing for digital signal processing: a survey. *Journal of VLSI Signal Procesing Systems*, 28:7–27, 2001. [Cited in pag. 29]

[67] M. Hsueh and C.-I Chang. Field programmable gate arrays (FPGAs) for pixel purity index using blocks of skewers for endmember extraction in hyperspectral imagery. *International Journal of High Performance Computing Applications*, 22(4):408–423, 2008. [Cited in pag. 29]

[68] C. Gonzalez, J. Resano, D. Mozos, A. Plaza, and D. Valencia. FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis. *EURASIP Journal on Advances in Signal Processing*, 2010(id:969806):1–13, 2010. [Cited in pags. 29 and 50]

[69] P. G. Cappelaere, D. Mandl, J. Stanley, S. Frye, and P. Baumann. WCPS: an open geospatial consortium standard applied to flight hardware/software. *In Proceedings of the American Geophysical Union*, 2009. [Cited in pag. 37]

[70] S. G. Ungar, J. S. Pearlman, J. A. Mendenhall, and D. Reuter. Overview of the Earth observing one (EO-1) mission. *IEEE Transactions on Geoscience and Remote Sensing*, 41(6):1149–1159, 2003. [Cited in pags. 37, 38 and 44]

[71] 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. 37, 52 and 62]

[72] D. C. Guerin, J. Fisher, and E. R. Graham. The enhanced MODIS airborne simulator hyperspectral imager. *In Proceedings of SPIE, Algorithms and Technologies for Multispectral, hyperspectral, and Ultraspectral Imagery XVII*, 8048(id:80480L):1–10, 2011. [Cited in pag. 37]

[73] Q. Zhang, E. M. Middleton, B.-C. Gao, and Y.-B. Cheng. Using EO-1 hyperion to simulate HyspIRI products for a coniferous forest: the fraction of PAR absorbed by chlorophyll and leaf water content (LWC). *IEEE Transactions on Geoscience and Remote Sensing*, 50(5):1844–1852, 2012. [Cited in pags. 38 and 50]

[74] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: geometrical, statistical and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379, 2012. [Cited in pag. 38]

[75] M. Parente and A. Plaza. Survey of geometric and statistical unmixing algorithms for hyperspectral images. *In Proceedings of the 2nd IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, Reykjavik, Iceland, Jun. 14–16, 2010. [Cited in pag. 38]

[76] J. M. Bioucas-Dias and A. Plaza. Hyperspectral unmixing: geometrical, statistical, and sparse regression-based approaches. *In Proceedings of SPIE, Image and Signal Processing for Remote Sensing XVI*, 7830:1–15, Toulouse, France, Sept. 20–23, 2010. [Cited in pags. 38 and 39]

[77] A. Plaza, Q. Du, Y.-L. Chang, and R. L. 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. 38]

[78] M. E. Winter. N-FINDR: an algorithm for fast autonomous spectral endmember determination in hyperspectral data. *In Proceedings of SPIE, Imaging Spectrometry V*, 3753:266–270, 1999. [Cited in pags. 39, 43, 45 and 52]

[79] M. Partridge and R. Calvo. Fast dimensionality reduction and simple PCA. *Intelligent Data Analysis*, 2(1–4):203–214, 1998. [Cited in pag. 43]

[80] D. W. Deering. Rangeland reflectance characteristics measured by aircraft and spacecraft sensors. *Ph.D. Diss. Texas A&M Univ., College Station, 632p*, 1978. [Cited in pag. 44]

[81] FLAASH Module User's Guide. ENVI FLAASH Version 4.2, Research Systems, Inc. August, 2005 Edition. Available online: http://geol.hu/data/online_help/flaash.pdf. [Cited in pag. 44]

[82] Center for the Study of Earth from Space (CSES). Atmosphere Removal Program (ATREM), Version 3.1, Users Guide, University of Colorado, Boulder, 12 p.,. 2008. [Cited in pag. 44]

[83] L. Ma, M. Crawford, and J. Tian. Local manifold learning-based k-nearest-neighbor for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4099–4109, 2010. [Cited in pags. 47 and 48]

[84] S. Sanchez and A. Plaza. Real-time implementation of a full hyperspectral unmixing chain on graphics processing units. *In Proceedings of SPIE Optics and Photonics, Satellite Data Compression, Communication, and Processing Conference*, 8157:1–9, 2011. [Cited in pag. 50]

[85] 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. 50]

[86] S. Kaewpijit, J. Le Moigne, and T. El-Ghazawi. Automatic reduction of hyperspectral imagery using wavelet spectral analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4):863–871, 2003. [Cited in pag. 50]

[87] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock. Imaging spectrometry for Earth remote sensing. *Science*, 228(4704):1147–1153, 1985. [Cited in pag. 52]

[88] C.-I Chang and D. C. Heinz. Constrained subpixel target detection for remotely sensed imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 38(3):1144–1159, 2000. [Cited in pag. 52]

[89] C. C. Borel and S. A. W. Gerstl. Nonlinear spectral mixing models for vegetative and soil surfaces. *Remote Sensing of Environment*, 47(3):403–416, 1994. [Cited in pag. 52]

[90] W. Liu and E. Y. Wu. Comparison of non-linear mixture models: sub-pixel classification. *Remote Sensing of Environment*, 94:145–154, 2005. [Cited in pag. 52]

[91] K. J. Guilfoyle, M. L. Althouse, and C.-I Chang. A quantitative and comparative analysis of linear and nonlinear spectral mixture models using radial basis function neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 39(10):2314–2318, 2001. [Cited in pag. 52]

[92] J. W. Boardman, F. A. Kruse, and R. O. Green. Mapping target signatures via partial unmixing of AVIRIS data. *Summaries of the JPL Airborne Earth Science Workshop*, pages 23–26, 1995. [Cited in pag. 52]

[93] J. H. Bowles, P. J. Palmadesso, J. A. Antoniades, M. M. Baumback, and L. J. Rickard. Use of filter vectors in hyperspectral data analysis. *In Proceedings of SPIE, Infrared Spaceborne Remote Sensing III*, 2553:148–157, 1995. [Cited in pag. 52]

[94] A. Ifarraguerri and C.-I Chang. Multispectral and hyperspectral image analysis with convex cones. *IEEE Transactions on Geoscience and Remote Sensing*, 37(2):756–770, 1999. [Cited in pag. 52]

[95] C. A. Lee, S. D. 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 pags. 54 and 91]

[96] Y. Tarabalka, T. V. Haavardsholm, I. Kasen, and T. Skauli. Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing. *Journal of Real-Time Image Processing*, 4(3):287–300, 2009. [Cited in pags. 54 and 69]

[97] C.-C. Chang, Y.-L. Chang, M.-Y. Huang, and B. Huang. Accelerating regular LDPC code decoders on GPUs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):653–659, 2011. [Cited in pags. 54 and 91]

[98] J. Camara, J. Cuenca, D. Gimenez, and A. M. Vidal. Empirical autotuning of two-level parallel linear algebra routines on large cc-NUMA systems. *In Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications*, pages 843–844, Leganes, Spain, July 10–13, 2012. [Cited in pag. 58]

[99] J. A. Benediktsson, M. Pesaresi, and K. Arnason. Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):1940–1949, 2003. [Cited in pags. 68 and 71]

[100] J. A. Richards. Analysis of remotely sensed data: the formative decades and the future. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):422–432, 2005. [Cited in pags. 68 and 72]

[101] S. Bernabe and A. Plaza. A new system to perform unsupervised and supervised classification of satellite images from google maps. *In Proceedings of SPIE, Conference on Satellite Data Compression, Communications, and Processing*, 7810:1–10, San Diego, CA, 2010. [Cited in pag. 68]

[102] R. L. King. Putting information into the service of decision making: the role of remote sensing analysis. *In Proceedings of the IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data*, pages 25–29, 2003. [Cited in pag. 68]

[103] J. Chanussot, J. A. Benediktsson, and M. Fauvel. Classification of remote sensing images from urban areas using a fuzzy possibilistic model. *IEEE Geoscience and Remote Sensing Letters*, 3(1):40–44, 2006. [Cited in pag. 68]

[104] L. Bruzzone, M. Chi, and M. Marconcini. A novel transductive SVM for semisupervised classification of remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(11):3363–3373, 2006. [Cited in pag. 68]

[105] F. Dell'Acqua, P. Gamba, A. Ferrari, J. A. Palmason, J. A. Benediktsson, and K. Arnason. Exploiting spectral and spatial information in hyperspectral urban data with high resolution. *IEEE Geoscience and Remote Sensing Letters*, 1(4):322–326, 2004. [Cited in pags. 68, 73, 90 and 93]

[106] G. H. Ball and D. J. Hall. *ISODATA. A novel method of data analysis and pattern classification.* Technical Report AD-699616, Stanford University, 1965. [Cited in pags. 68, 72, 90 and 93]

[107] J. A. Hartigan and M. A. Wong. Algorithm as 136: a k-means clustering algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28:100–108, 1979. [Cited in pags. 68, 72, 90, 93 and 105]

[108] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. [Cited in pag. 68]

[109] R. Chellappa and A. Jain. *Markov random fields: theory and applications.* Academic Press: New York, 1993. [Cited in pag. 68]

[110] J.-M. Beaulieu and M. Goldberg. Hierarchy in picture segmentation: a stepwise optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):150–163, 1989. [Cited in pag. 72]

[111] G. M. Foody. Status of land cover classification accuracy assessment. *Remote Sensing of Environment*, 80:185–201, 2002. [Cited in pag. 79]

[112] G. Camps-Valls and L. Bruzzone. Kernel-based methods for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1351–1362, 2005. [Cited in pag. 88]

[113] S. Bernabe, A. Plaza, P. R. Marpu, and J. A. Benediktsson. A new parallel tool for classification of remotely sensed imagery. *Computers and Geosciences*, 46:208–218, 2012. [Cited in pags. 90, 112 and 123]

[114] S. N. V. Kalluri, Z. Zhang, J. Jaja, S. Liang, and J. R. G. Townshend. Characterizing land surface anisotropy from AVHRR data at a global scale using high performance computing. *International Journal of Remote Sensing*, 22(11):2171–2191, 2001. [Cited in pag. 91]

[115] A. Plaza. Special issue on architectures and techniques for real-time processing of remotely sensed images. *Journal of Real-Time Image Processing*, 4:191–193, 2009. [Cited in pag. 91]

[116] T. Balz and U. Stilla. Hybrid GPU-based single- and double-bounce SAR simulation. *IEEE Transactions on Geoscience and Remote Sensing*, 47(10):3519–3529, 2009. [Cited in pag. 91]

[117] J. Mielikainen, B. Huang, and H. A. Huang. GPU-accelerated multi-profile radiative transfer model for the infrared atmospheric sounding interferometer. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):691–700, 2011. [Cited in pag. 91]

[118] C. Song, Y. Li, and B. Huang. A GPU-accelerated wavelet decompression system with SPIHT and Reed-Solomon decoding for satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):683–690, 2011. [Cited in pag. 91]

[119] S.-C. Wei and B. Huang. GPU acceleration of predictive partitioned vector quantization for ultraspectral sounder data compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):677–682, 2011. [Cited in pag. 91]

[120] 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 pag. 91]

[121] J. Le Moigne, N. S. Netanyahu, and R. D. Eastman. *Image registration for remote sensing.* Cambridge University Press: Wiley, New York, 2011. [Cited in pag. 93]

[122] H. G. Lewis and M. Brown. A generalized confusion matrix for assessing area estimates from remotely sensed data. *International Journal of Remote Sensing*, 22(16):3223–3235, 2001. [Cited in pag. 106]