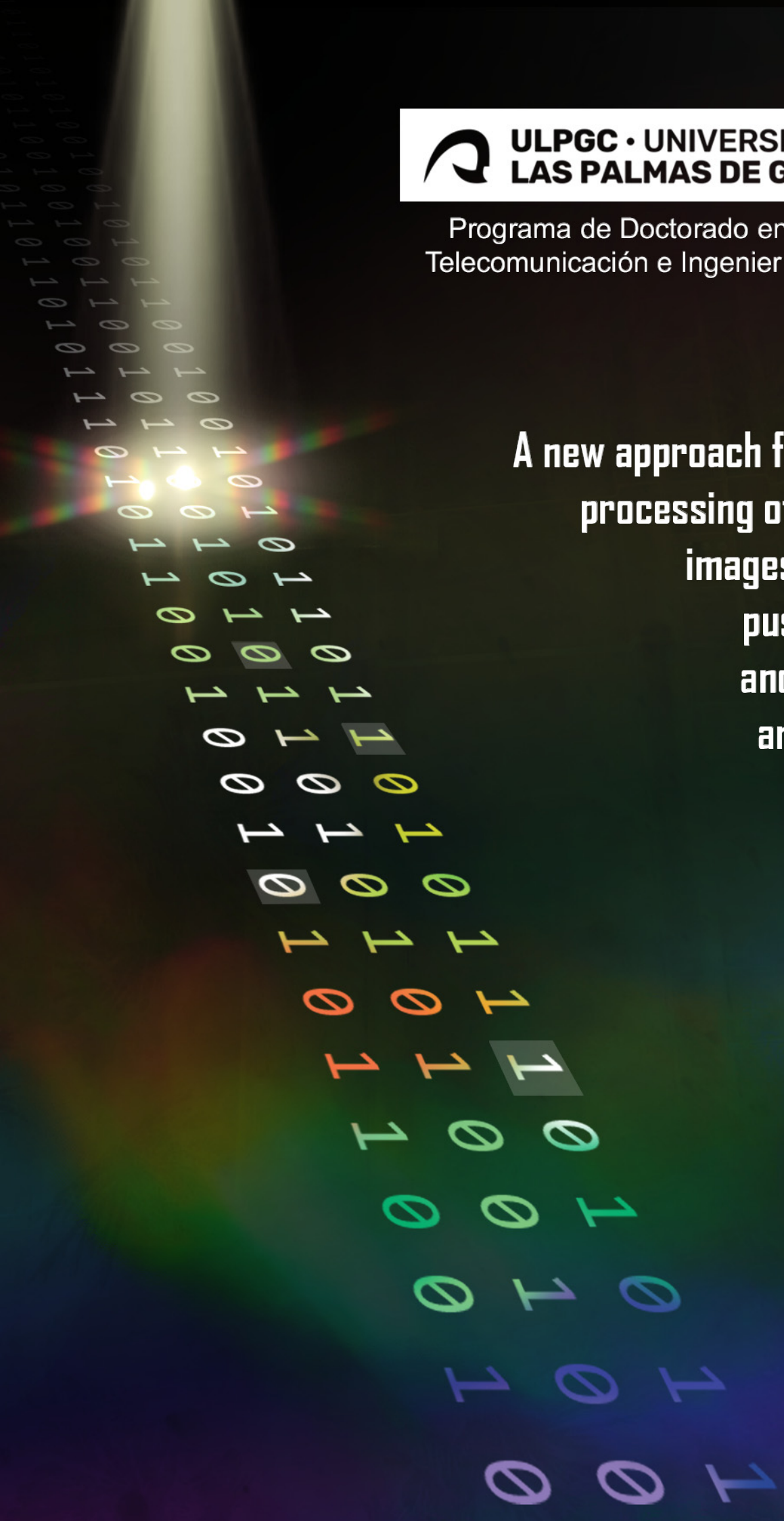


ULPGC • UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Programa de Doctorado en Tecnologías de Telecomunicación e Ingeniería Computacional

A new approach for the effective processing of hyperspectral images: application to pushbroom-based anomaly detection and compression systems

María Díaz Martín

PhD Thesis

Las Palmas de G.C.

Mayo 2021

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

D. GUSTAVO MARRERO CALLICÓ, COORDINADOR DEL PROGRAMA DE DOCTORADO EN TECNOLOGÍAS DE TELECOMUNICACIÓN E INGENIERÍA COMPUTACIONAL DE LA UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA,

INFORMA,

Que la Comisión Académica del Programa de Doctorado, en su sesión de fecha 16 de junio de 2021 tomó el acuerdo de dar el consentimiento para su tramitación, a la tesis doctoral titulada "*A new approach for the effective processing of hyperspectral images: application to pushbroom-based anomaly detection and compression systems*" presentada por la doctoranda Dña. María Díaz Martín y dirigida por el Doctor D. Sebastián López Suárez. Esta Tesis cuenta además con Mención Internacional.

Y para que así conste, y a efectos de lo previsto en el Artº 11 del Reglamento de Estudios de Doctorado (BOULPGC 7/10/2016) de la Universidad de Las Palmas de Gran Canaria, firmo la presente en Las Palmas de Gran Canaria, a 16 de junio de dos mil veintiuno.

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Instituto Universitario de Microelectrónica Aplicada

Instituto: INSTITUTO UNIVERSITARIO DE MICROELECTRÓNICA APLICADA

Programa de doctorado: DOCTORADO EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN E INGENIERÍA COMPUTACIONAL

Título de la Tesis

A NEW APPROACH FOR THE EFFECTIVE PROCESSING OF HYPERSPECTRAL IMAGES: APPLICATION TO PUSHBROOM-BASED ANOMALY DETECTION AND COMPRESSION SYSTEMS.

Tesis Doctoral presentada por Dña. MARÍA DÍAZ MARTÍN

Dirigida por el Dr. D. SEBASTIÁN LÓPEZ SUÁREZ

El Director/a,
(firma)

La Doctoranda
(firma)

Las Palmas de Gran Canaria, a 28 de Mayo de 2021

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada

DIVISIÓN DE DISEÑO DE SISTEMAS INTEGRADOS

TESIS DOCTORAL

A new approach for the effective processing of hyperspectral images: application to pushbroom-based anomaly detection and compression systems.

María Díaz Martín

Gobierno de Canarias
Consejería de Economía, Industria, Comercio y Conocimiento
Agencia Canaria de Investigación, Innovación y Sociedad de la Información

Canarias avanza con Europa
Fondo Social Europeo

Unión Europea

IUMA

Abstract

Hyperspectral imagery has gained an increasing interest by the scientific community in the last years in such a manner that it has been consolidated as one of the mainstream terrestrial Earth observing systems. Its growing popularity lies in the large amount of information along the electromagnetic spectrum that this technology brings for each single image pixel. In this regard, hyperspectral images (HSIs) are able to capture the reflection distribution from the observed objects along a broader range of the electromagnetic spectrum dividing it in many contiguous spectral bands. This characteristic enables detailed examinations of the land surfaces and the identification of visually similar materials.

Currently, there is a wide variety of remote sensing data acquisition systems, such as satellites, aircraft and more recently, drones. Nonetheless, the onboard real-time hyperspectral image processing still poses several challenges before it becomes a reality. Indeed, this situation jeopardizes the real-time response of time-sensitive applications that demand quick response. In this regard, images acquired by remote Earth observation platforms are traditionally downloaded to the ground segment for being off-line processed on supercomputing systems. Unfortunately, the main problem lies in the data transmission since important delays are introduced related to the communication of large amount of data and the bottleneck represented by the limited communication bandwidth of the downlink system. Consequently, this operating mode clearly compromises the efficient and safe execution of stringent applications that require immediate results.

Regrettably, the algorithms traditionally proposed for the hyperspectral analysis normally give rise to complex algorithms characterized by computationally costly operations, intensive memory requirements, high implementation costs and a non-scalable nature. This is because the algorithm developers are normally more concerned about the mathematical methods that optimize the quality of the results than the importance of their real-time performance in power-constrained scenarios, such as the onboard processing.

The aforementioned context becomes even more challenging when different time-sensitive applications coexist in the same computing device. The simplest and the most commonly adopted solution is to select a different mathematical algorithm from the wide assortment of proposals encountered in the literature for each hyperspectral image processing type to be performed and then, to accelerate them using parallel computing devices. The issue arises when they have to be sequentially processed onto the same computing device due to

restrictions in terms of power, weight and size. Therefore, there is a need in the literature for new algorithmic solutions that take into consideration the above mentioned currently existing constrains imposed by nowadays remote sensing applications. Additionally, the causality inherent to real-time frameworks based on pushbroom/whiskbroom scanners must be also met through the definition of non-global algorithms capable of independently processing blocks of image pixels. In turn, this prevents the storing and management of large data volumes, thereby reducing the computing resources and speeding up the execution process.

Against this backdrop, we have dealt in this Thesis work with the issue around the on-board execution of multiple hyperspectral image analysis techniques onto the same piece of hardware, paving the way for the real-time performance of the hyperspectral image processing. The main objective of this Thesis is to provide the research community with a set of common core operations that extract useful information from the HSIs for many applications; such as anomaly detection, target detection, lossy compression, classification, and unmixing. On the one hand, it results in many benefits in view of hardware acceleration in terms of a reduction in the execution times, hardware resources and above all, in human endeavours. Concerning this latter, it implies the studio and analysis of only a single mathematical approach, which consequently permits to focus the efforts from a methodological and productivity points of view. On the other hand, it also permits the simultaneous execution of many different tasks at the same time with the advantage of sharing the most computationally intensive operations. As a consequence, it promotes the decrease in the amount of computational resources compared with those scenarios in which different state-of-the-art algorithms are independently executed for each targeted processing analysis.

Based on the aforementioned set of core operations, a new algorithm for the detection of anomalous spectra has been also developed in this Thesis, named *A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery* (LbL-FAD). The LbL-FAD algorithm is a subspace-based anomaly detector designed to fulfil the constraints imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners. In this regard, the LbL-FAD algorithm is able to independently process blocks of hyperspectral pixels with not taking into consideration any spatial alignment requirement.

Additionally, a performance-enhancing version of the state-of-the-art *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) has been proposed for the spectral decorrelation and compression of HSIs. In this sense, the original algorithmic proposal

has been widened in this Thesis work in order to be adapted and, thereby, fallen within the proposed set of core operations. Moreover, an efficient and comprehensive compression system has been also introduced. As a further advantage, the HyperLCA algorithm permits compressing blocks of image pixels independently. This feature promotes, on the one hand, the reduction of the data to be managed at once, besides, the hardware resources to be allocated and, on the other hand, it becomes a very competitive solution for most applications based on pushbroom/whiskbroom scanners.

The feasibility of the concurrent execution of multiple hyperspectral analysis techniques based on the same mathematical method has been also demonstrated. In particular, it was verified the suitability of the proposed methodology for the concurrent execution of both the lossy compression of HSIs and the detection of anomalous signatures. To meet this issue, two optimized versions were eventually proposed. The former, referred to as *Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (ADeLoC), searches for the highest accuracy in the detection and compression results. The latter, named *Hardware-friendly proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (HADeLoC) prioritizes the optimization of the hardware resources and the minimization of the execution times at the expense of a loss of accuracy in the compression results.

In the interest of confirming the benefits of developing algorithmic solutions based on the same mathematical method and, also to verify the suitability of the developed algorithms for real-time applications, the main algorithmic proposals of this Thesis have been also implemented on different parallel devices, namely GPUs and FPGAs. Concretely, the LbL-FAD, the HyperLCA and the HADeLoC methods have been accelerated on a Xilinx system on chip (SoC) FPGA device, while the HyperLCA has been adapted to be launched in embedded computing boards from NVIDIA.

We have also briefly discussed the possibility of extending the use of the set of core operations proposed in this Thesis work, in the fields of band selection, target detection, unmixing and classification. Although the analysis made is far from being as exhaustive as those carried out by LbL-FAD detector and the HyperLCA compressor, it indeed represents a turning point in the way of future research works.

Finally, the algorithms and implementations carried out ratifies the suitability of the proposed algorithmic solution in the field of the onboard real-time processing of HSIs, certainly in the line of the research goals to be accomplished in this Thesis.

Resumen

En los últimos años, la tecnología hiperespectral se ha convertido en una herramienta de gran interés para la comunidad científica en el campo de la teledetección y la observación de la superficie terrestre. Su creciente popularidad se fundamenta en su capacidad para recoger información sobre la escena observada en muchas y continuas longitudes de onda a lo largo del espectro electromagnético. Esto deriva en la posibilidad de detectar e identificar distintos tipos de materiales presentes en la naturaleza que a simple vista pudiesen parecer el mismo ente.

En la actualidad, existen una gran variedad de plataformas de adquisición de datos hiperespectrales, como son los satélites, las aeronaves y recientemente, los drones. Sin embargo, el procesamiento a bordo y en tiempo real de las imágenes hiperespectrales aún presenta importantes retos a abordar para convertirse en una realidad. Esto a su vez presenta ciertos inconvenientes prioritarios, sobre todo para aquellas aplicaciones que requieren de resultados inmediatos. En este sentido, el procedimiento tradicional se ha centrado en el procesamiento en la superficie terrestre de los datos capturados remotamente tras su transmisión y descarga. Sin embargo, el principal problema radica en la transmisión de los datos debido a los importantes retrasos derivados del limitado ancho de banda de los sistemas de comunicación. Por lo tanto, este sistema de operación claramente compromete la ejecución eficiente, segura y en tiempo real de aplicaciones que requieren de respuestas inmediatas o en un corto periodo de tiempo.

Lamentablemente, los algoritmos tradicionalmente propuestos para el análisis de los datos hiperespectrales dan lugar a propuestas algorítmicas muy complejas, difícilmente implementables debido a la ejecución de operaciones computacionalmente costosas, intensivas en consumo de memoria y recursos *hardware* y caracterizadas por altas dependencias de datos. Esto se debe a que los desarrolladores de soluciones algorítmicas normalmente centran sus esfuerzos en los métodos matemáticos, buscando la optimización en los resultados obtenidos pero dejando en segundo plano la importancia de su funcionalidad en tiempo real y en escenarios limitados en términos de recursos, como es el procesamiento a bordo.

El escenario anteriormente planteado se vuelve aún más complicado cuando distintos procesos de análisis hiperespectral deben ser ejecutados de manera simultánea y coexistir en un único dispositivo de cómputo asegurando respuestas en tiempo real. En este sentido, la solución más simple y comúnmente utilizada se basa en la selección de distintos algoritmos para cada aplicación a llevar a cabo y acelerarlos usando dispositivos de cómputo

paralelo. El problema radica en el momento de su ejecución concurrente en un mismo dispositivo *hardware* en escenarios caracterizados por restricciones en términos de consumo, recursos de cómputo, peso y espacio disponibles, como son los drones. Por lo tanto, hay una gran necesidad en la literatura de soluciones algorítmicas que tengan en consideración las actuales restricciones y limitaciones existentes en las aplicaciones demandadas hoy en día. Además, también se requiere de la definición de nuevas alternativas algorítmicas que tengan en cuenta la causalidad inherente a los procesos de captura realizados por sistemas de tipo *pushbroom* y *whiskbroom*.

Partiendo de este contexto, la realización de esta Tesis Doctoral ha contribuido al campo del procesamiento abordo y en tiempo real de las imágenes hiperespectrales en aplicaciones donde múltiples técnicas de análisis deban coexistir en un único dispositivo de cómputo. Su principal objetivo se centra en proveer a la comunidad científica con un conjunto de operaciones capaces de extraer información espectral de utilidad para la realización de múltiples técnicas de análisis hiperespectral. El hecho de centrarse en la utilización de un único método matemático es especialmente beneficioso para la aceleración *hardware* de estos procesos. Por una parte, esto se traduce en un ahorro de tiempo, costes y esfuerzo humano durante la etapa de implementación *hardware* de estas soluciones algorítmicas pues sólo un único método matemático debe ser estudiado, comprendido y desarrollado. Por otra parte, esta metodología permite la ejecución conjunta de diversas tareas de procesamiento con la ventaja de compartir las operaciones más computacionalmente costosas y complejas, con los beneficios derivado de ello.

Basado en el conjunto de operaciones anteriormente citado, se ha desarrollado un nuevo algoritmo para la detección de agentes anómalos llamado *A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery* (LbL-FAD). Dicho algoritmo ha sido especialmente diseñado para contribuir a la literatura con algoritmos capaces de procesar las imágenes línea a línea y por tanto, está especialmente destinado a ser empleado en aplicaciones basadas en sistemas de adquisición del tipo *pushbroom/whiskbroom*.

También se ha propuesto una versión mejorada del algoritmo del estado del arte titulado *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) para la decorrelación espectral y compresión de las imágenes hiperespectrales. En este sentido, la metodología descrita en sus inicios se ha ampliado con el fin de ser adaptada y poder ejecutarse con el conjunto de operaciones propuesto en esta Tesis Doctoral. Además, también se ha definido un sistema de compresión de imágenes hiperespectrales completo que incluye la etapa de codificación entrópica de los vectores que conforman los datos

comprimidos. Este a su vez permite realizar una compresión linea a línea, lo que convierte esta propuesta en una candidata idónea para sistemas a bordo basados en sensores de tipo *pushbroom/whiskbroom*.

La viabilidad de la ejecución conjunta de múltiples técnicas de análisis hiperespectral basadas en el mismo método matemático también ha sido verificada en esta Tesis Doctoral. En particular, se ha verificado la idoneidad de la metodología propuesta para la ejecución simultánea de la compresión con pérdidas de imágenes hiperespectrales y la detección de firmas espectrales anómalas. Para atender esta problemática se han propuesto dos soluciones algorítmicas optimizadas. La primera, denominada *Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (ADeLoC), busca la mayor precisión en los resultados en términos de detección y compresión. La segunda propuesta, denominada *Hardware-friendly proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (HADeLoC), prioriza la optimización de los recursos *hardware* y la minimización de los tiempos de ejecución a expensas de una pérdida en la precisión y la calidad de los resultados en la etapa de compresión.

En aras de confirmar los beneficios derivados de desarrollar soluciones algorítmicas basadas en el mismo método matemático y también verificar la idoneidad de las propuestas algorítmicas desarrolladas para aplicaciones en tiempo real, estas han sido implementadas en diversos dispositivos de cómputo paralelo, tales como *FPGAs* y *GPUs*. En concreto, los algoritmos LbL-FAD, HyperLCA y HADeLoC han sido acelerados en plataformas SoC, de sus siglas en inglés *System on Chip*, basados en FPGAs de Xilinx. Además, el compresor HyperLCA también ha sido adaptado para su ejecución en plataformas embebidas basadas en GPU de NVIDIA.

También se ha analizado brevemente la posibilidad de extender la metología propuesta, basada en el conjunto de operaciones propuesta en esta Tesis Doctoral, a otras técnicas de procesamiento de imágenes hiperespectrales, como son la selección de bandas, el desmezclado, la detección de objetivos de interés y la clasificación.

Finalmente, los algoritmos e implementaciones desarrollados ratifican la idoneidad de la solución algorítmica propuesta en el campo del procesamiento a bordo y en tiempo real de las imágenes hiperespectrales, lo que claramente va en la línea de los objetivos planteados para la consecución de esta Tesis Doctoral.

Acknowledgements

This has been a long roller coaster ride that finally came to an end. There have been moments of euphoria and happiness but, also times of uncertainty and frustration. Nonetheless, the experiences lived in the last years have really made me grow both professionally and personally. They have shown me that it is possible to do whatever you want in life with confidence and perseverance. Thus, I would like to thank those people who have been part of this long journey.

For starters, my Thesis director Sebastián López, not only due to his role of mentoring but also, for his good tips and the confidence in me and my work. I highly appreciate the warm encouragement and constructive comments given by José López and Roberto Sarmiento. I am also grateful to Jesús Barba and Julián Caba for the great welcome I had in my stay in their facilities and their great commitment and implication in the works carried out. I certainly cannot forget all my team mates with whom I have actually a beautiful friendship. Thank you so much for your support, the good conversations, the coffees and the patience to bear "my craziness".

Over the years, I have received funding from several entities, which have supported me while I completed my PhD. I would like to thank the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI) of the Conserjería de Economía, Industria, Comercio y Conocimiento of the Gobierno de Canarias, jointly with the European Social Fund (FSE) (POC2014-2020, Eje 3 Tema Prioritario 74 (85%)) and, the Institute for Applied Microelectronics (IUMA) for their financial support.

My special acknowledgement goes to my family who has offered me unconditional aid and support during this adventure. In particular to my parents and my little sister, it does not matter how hard the battle is, I will win with their support. I would also like to have a special attention to my comrade of thousand discussions and experiences, Raúl, for being my source of inspiration and believing in me. Thanks also to who I call "my little girls" for installing me the confidence that I sometimes lose.

Contents

List of Figures

List of Tables

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
HSI	Hyperspectral Image
AAT	Arbitrary Affine Transform
ACE	Adaptive Coherence/Cosine Estimator
AD	Anomaly Detection
ADeLoc	Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs
AISA	Airborne Imaging Spectrometer for Applications
ASIC	Application Specific Integrated Circuits
AutoGAD	Autonomous Global Anomaly detector
AVIRIS	Airborne Visible/InfraRed Imaging Spectrometer
bpppb	Bits Per Pixel Per Band
BS	Block size
CASI	Compact Airborne Spectrographic Imager
CCSDS	Consultative Committee for Space Data Systems

CHRIS	Compact High Resolution Imaging Spectrometer
CEM	Constraint Energy Minimization
CNN	Convolutional Neural Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CR	Compression ratio
CRD	Collaborative-representation-based Detector
CUDA	Computer Unified Device Architecture
DAIS	Digital Airborne Imaging Spectrometer
DBN	Deep Belief Network
DWT	Discrete Cosine Transform
DDR	Double Data Rate
DMR	Dual Modular Redundancy
DDR	Dynamic range
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
eMAS	Enhanced MODIS Airborne Simulator
ESA	European Space Agency
EnMAP	Environmental Mapping and Analysis Program
EO	Earth observation
FCLSU	Fully Constrained Least Squares linear spectral Unmixing
FF	FlipFlops

FIFO	First in, First out
FL	Fast Lossless
FLEX	Fast Lossless EXtended
FLOP	Floating-Point Operation
FPGA	Field-Programmable Gate Array
FUN	Fast algorithm for linearly UNmixing hyperspectral images
FOV	Field of view
FPR	False positive rate
FP	False positive
FPS	Frame per second
FSM	Finite State Machine
FWHM	Full width at half maximum
GAN	Generative Adversarial Network
GBs	Giga-bytes
GIPREBAD	Global Iterative Principal Component Analysis Reconstruction-Error-based anomaly detector
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HLS	High-Level Synthesis
HYDICE	Hyperspectral Digital Imagery Collection Experiment
HyMap	Hyperspectral Imaging Sensor
HyperLCA	Lossy Compression Algorithm for Hyperspectral image systems

HySIME Hyperspectral Subspace Identification by Minimum Error

HyT Hyperspectral Thermal Imaging

HW-LbL-FAD Hardware-friendly Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery

HADeLoC Hardware-friendly proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs

IASI Infrared Atmospheric Sounding Interferometer

ICA Independent Component Analysis

IQ Interquartile range

ICR Infrared

IP Intellectual property

IWT Integer Wavelet Transform

JPEG Joint Photographic Experts Group

KLT Kahrunen-Loève Transform

LbL-FAD A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery

LMM Linear Mixing Model

LMS Least Mean Square

LPGPU Low-power Graphics Processing Unit

LRASR Low Rank and Sparse Representation anomaly detector

LRR Low-Rank Representation

LRX Local Reed-Xiaoli

LSMAD Low-rank and Sparse matrix decomposition-based Mahalanobis distance method

LUT	Look-up-table
LWIR	Long Wavelength InfraRed
MAC	Maximum abundance classification
MAD	Maximum absolute difference
MaxSE	Maximum single error
MF	Matched Filter
MSE	Mean Square Error
MWIR	Medium Wavelength InfraRed
NASA	National Aeronautics and Space Administration
NIR	Near InfraRed
OP	Operation
OSP	Orthogonal Subspace Projection
OSPRX	Orthogonal Subspace Projection after Reed-Xiaoli
PCA	Principal Component Analysis
PCI	Peripheral Component Interconnect
PE	Processing Element
pFUN	parallel Fast algorithm for linearly UNmixing hyperspectral images
PL	Programmable Logic
PLP-KRXD	Progressive Line Processing of a Kernel version of the RX algorithm
POT	Pairwise Orthogonal Transform
PPI	Pixel Purity Index
PRISMA	Precursore Iperspettrale della Missione Applicativa

PS	Processor System
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Access Memory
RGB	Red, Green, Blue
RHBD	Radiation Hardened By Design
RIT	Rochester Institute of Technology
RMSE	Root Mean Square Error
ROIS	Reflective Optics System Imaging Spectrometer
RTL	Register Transfer Language
RX	Reed-Xiaoli
RXD	Reed-Xiaoli Detector
SAM	Spectral Angle Mapper
SAFL	Spectral Adversarial Feature Learning
SNR	Signal-to-Noise Ratio
SM	Streaming multiprocessor
SoC	System on chip
SRAM	Static Random Access Memory
SSH	(Secure Shell Protocol
SSRX	Subspace Reed-Xiaoli
SSRX	Structural Similarity Index
SVD	Singular Value Decomposition
SUnSAL	Spectral Unmixing by variable Sitting and Augmented Lagrangian

SVM	Support Vector Machine
SWIR	Short Wavelength Infrared
TMR	Triple Modular Redundancy
TPR	True Positive Rate
TP	True Positive
UAV	Unmanned aerial vehicles
UNRS	Unsupervised Nearest Regularized Subspace
USGS	U.S. Geological Survey
UTD	Uniform Target Detector
VCA	Vertex Component Analysis
VD	Virtual Dimensionality
Verilog HDL	Verilog Hardware Description Language
VHDL	VHSIC Hardware Description Language
VNIR	Visible and Near InfraRed
VSWIR	Visible Shortwave InfraRed
WASP	Wildfire Airborne Sensor Program
WLAN	Wireless local area network
WTD	World Trade Center

Chapter 1

Introduction

The main objective of this Chapter is to present the motivations for this Thesis, whose major contributions are in the field of hyperspectral image processing oriented to pushbroom-based systems working on real or near real-time constraints. Firstly, the basics about hyperspectral imaging are briefly outlined alongside with the most typical capturing processes and their applications. Secondly, the issue around the onboard processing of hyperspectral images is discussed as well as the necessity of new hardware-friendly algorithmic solutions that permits the coexistence of multiple hyperspectral analysis techniques onto the same computing device. Finally, the research goals and contributions of this Thesis as well as the organization of this document are presented.

1.1 Rationale

In the last decades, hyperspectral imagery systems have become one of the most powerful tools for the observation and data acquisition of the Earth surface. Nowadays, hyperspectral technology is widely used in multiple remote sensing applications such as precision agriculture, geoscience, environmental monitoring, surveillance and homeland security, among others [6–11]. In addition, hyperspectral imaging is also increasingly being used in other fields such as biomedical, commercial and industrial applications for food quality, medical diagnosis, quality control in recycling plants, in-line material sorting, etc. [12–16].

This rising popularity is driven by the great richness of spectral information that this three-dimensional (3D) imaging technique collects along the electromagnetic spectrum. Whereas the human eyes are sensitive to three broadband spectral bands (red, green and blue), hyperspectral sensors gather spectral information of hundreds of continuous and narrow wavelengths along the electromagnetic spectrum, which allows the analysis of spectrum areas beyond the visible light such as the near-infrared (NIR) or the infrared (IR). Therefore, any single pixel within a hyperspectral image (HSI) is associated with a full continuous spectrum in a given range, commonly called spectral signature of the pixel, that is helpful for the detection and identification of specific materials or targeted entities on the basis of their unique spectral signature.

Hence, the potential of the hyperspectral imaging systems for the Earth observation is clearly evident. For this reason, remote sensing is the field in which hyperspectral imaging techniques have traditionally acquired more relevance. In this sense, hyperspectral data analysis permits performing a detailed examination of the land surfaces and the identification of visually similar materials, as well as the estimation of physical parameters of many complex surfaces. Within this domain, three different hyperspectral acquisition platforms have been extensively utilized, such as aeroplanes, satellites and drones.

Airborne hyperspectral sensors; such as Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS) [17], Compact Airborne Spectrographic Imager (CASI) [18], Hyperspectral Digital Imagery Collection Experiment (HYDICE) [19], Digital Airborne Imaging Spectrometer (DAIS) [20], Reflective Optics System Imaging Spectrometer (ROSIS) [21], Airborne Imaging Spectrometer for Applications (AISA) [22], Hyperspectral Imaging Sensor (HyMap) [23], Enhanced MODIS Airborne Simulator (eMAS) [24], among others [25];

have played a decisive role in the inclusion of the hyperspectral research in the development of the current remote sensing applications and data science due to the well trade-off between spatial and spectral resolution.

Nonetheless, airborne systems do not allow regular and synoptic coverages over large areas as spaceborne sensors [6]. For this reason, different spaceborne missions carrying hyperspectral satellites have been launched since 1992, such as Hyperion (EO-1 Mission) [26], Compact High Resolution Imaging Spectrometer (CHRIS) [27], Precursore Iperspettrale della Missione Applicativa (PRISMA) [28], Environmental Mapping and Analysis Program (EnMAP) [29]. This reduced number of hyperspectral spaceborne sensors is rooted by the high volume of data to be stored and processed against the limited hardware resources, apart from other issues, such as involved sensor costs and external factors as atmospheric distortions.

Additionally to the previous solutions, the unmanned aerial vehicles (UAVs) are gaining momentum in the last years in applications oriented to collect data for inspection, surveillance and monitoring in the areas of defence, security, environmental protection, precision agriculture and civil domains, among others [11, 30, 31]. The potential of these aerial platforms in relation to the previous solutions, such as satellites or manned airborne platforms, is that they represent a lower-cost approach with a more flexible revisit time and a better spatial and spectral resolution, which permits a deeper and more accurate data analysis. UAVs have also attractive features in terms of flexibility to carry different types of sensors and to plan and modify a trajectory if necessary. The main drawbacks of these aerial observation platforms are their limited autonomy due to the battery life, which is around 30 minutes for a commercial product [32].

Nevertheless, despite the variety of remote sensing platforms, images are traditionally downloaded to the ground segment in order to be off-line pre-processed and then distributed to the final users that perform the final processing (classification, unmixing, target detection, anomaly detection, etc.) on supercomputing systems typically based on Graphics Processing Units (GPUs), Central Processing Units (CPUs), heterogeneous CPU/GPU architectures, or even Field-Programmable Gate Array (FPGAs) [33]. The main reasons behind this are the limited available onboard computational resources as well as the restrictions in power budget and storage space [34, 35].

Unfortunately, the limited bandwidth of the downlink connection from the remote sensing platforms to the Earth surface introduces delays related to the transmission of a large amount of data between the source and the final target. This introduces a bottleneck

that can seriously reduce the effectiveness of real-time applications or applications that demand prompt replies [36, 37]. Consequently, real-time onboard processing has become a very interesting topic within the remote sensing field to provide a solution to this type of applications [37–40].

Nevertheless, it is still necessary to overcome many other obstacles imposed by the available aboard hardware devices in order to efficiently carry out this alternative and to execute onboard the hyperspectral data processing. In this context, many different algorithms have been developed in the last decades for providing solutions to all the aforementioned applications using the hyperspectral technology. However, hyperspectral imaging applications under real or near real-time constraints require prompt response and hence, to rapidly execute the algorithmic proposals. On this basis, the most commonly used solution consists in implementing the hyperspectral processing algorithms into high performance architectures that are suitable for parallel computing, such as GPUs and FPGAs. In fact, many works have been uncovered during the last years for implementing different algorithms for processing HSIs into parallel hardware with the purpose of speeding up the hyperspectral analysis process [41–47].

Regrettably, the algorithms traditionally proposed for the hyperspectral analysis have been addressed as independent entities, using those mathematical methods that better maximize the results for each particular case. In addition, these approaches normally give rise to complex algorithms characterized by computationally costly operations, intensive memory requirements, high implementation costs and a non-scalable nature. For instance, some of the most widely used operations are eigenvalue decomposition, covariance matrix computation, matrix inverse computation and factorization, resulting in very hard problems. For these reasons, a great deal of effort has been made to reduce the heavy computational burden required by these operations in order to accelerate the process through alternative mathematical methods or through parallel computing via high-performance architectures. Nevertheless, this last alternative results in very inefficient hardware implementations where there is little room for speeding them up due to the mathematical complexity of the involved operations as well as the so many data dependencies.

All of this makes the onboard processing of the acquired hyperspectral data not fully viable, especially under real-time constraints [38, 48]. In view of the computationally intensive nature of the hyperspectral imaging algorithms found in the literature, many efforts have been done in the last years in order to develop lower computational complexity algorithmic solutions. Unfortunately, many of these state-of-the-art methods need to work

with the entire hyperspectral image cube to perform the hyperspectral analysis. This represents an important limitation for real-time applications, specially considering that the most widely used sensors in remote sensing applications are based on pushbroom scanners [49], [31], which sense the data in a line-by-line fashion, perpendicular to the flight direction of the capturing platform [50], [51].

This situation becomes even more stringent when multiple time-sensitive image processing techniques have to be executed. In this case, the simplest and most commonly used solution is to select a different mathematical algorithm from the wide assortment of proposals encountered in the literature for each hyperspectral data analysis technique to be addressed and accelerated using parallel hardware devices. The problem arises in that they have to be sequentially processed onto the same computing device due to restrictions in terms of power, weight and size. In this sense, FPGAs offer the appealing possibility of adaptively selecting a hyperspectral processing algorithm to be applied due to their inherent reconfigurability nature while GPUs deliver up to hundreds and thousands of Gigaflops of double-precision peak performance. However, all of this is not without challenges in terms of energy, flexible parallelization strategies, execution times, resource optimization and human endeavours invested during the implementation stage [52, 53].

Unfortunately, the above mentioned issues make the onboard execution of hyperspectral imaging processing not fully viable, specially under real-time constraints when different time-sensitive applications coexist onto the same computing device. In this context, this Thesis has contributed to address the existing needs of nowadays remote sensing applications as follows:

- Developing hardware-friendly algorithmic solutions that are thought from their conception for being implemented into parallel computing devices.
- Adapting the proposed methods for being executed in a line-by-line fashion in order to provide a compromise performance for applications based on pushbroom/whiskbroom sensors.
- Defining a set of reusable core operations that permits the simultaneous execution of many different tasks at the same time with the advantage of sharing the most computationally intensive operations. As a consequence, it promotes the decrease in the amount of computational resources compared with those scenarios in which different state-of-the-art algorithms are independently executed for each targeted processing analysis.

- Demonstrating the real-time performance of the proposed algorithmic solutions through their acceleration in low-power parallel computing devices, such as FPGAs and GPUs.

1.2 Preliminary concepts

The following subsections introduce the reader to the main topics that will be repeatedly mentioned throughout this document, including a definition about spectral images and how they are characterized according to their spectral and spatial resolutions, an explanation about the different types of existing data collection systems and, a brief explanation of the motivations of this Thesis. On this later point, the issues around the onboard processing of HSIs are listed as well as the majority of hyperspectral imaging analysis techniques and the methods traditionally employed to address them. Additionally, a short clarification of how we have tackled this matter with the set of core operations proposed in this Thesis is also given.

1.2.1 Characterization and resolution of the spectral images

Spectroscopy studies the interaction between the light that is emitted by or reflected from materials and its variation in energy with wavelength. As applied to the field of optical remote sensing, spectroscopy exploits the fact that materials reflect, scatter, absorb and emit electromagnetic radiation in ways characteristic of their molecular composition and their macroscopic scale and shape [54]. In the field of reflected-light spectroscopy, spectral imaging sensors are designed to measure the ratio of reflected energy to incident light as a function of wavelength. In this context, reflectance variation with the wavelength is a property of the materials that selectively absorb the incident energy in certain regions of the electromagnetic spectrum. On this basis, by measuring the reflectance arriving at a sensor at many broad spectral bands, the resulting pattern or spectrum can be used to identify the materials in a scene and discriminate among different classes of materials, as shown in Figure 1.1 where the spectrum of different materials are plotted.

All this leads to 3D spectral images composed of a stack of two-dimensional (2D) images that represent the reflectance in one respective band or wavelength interval [55]. Due to this interpretation, HSIs are also termed *data cubes* where the spatial information is

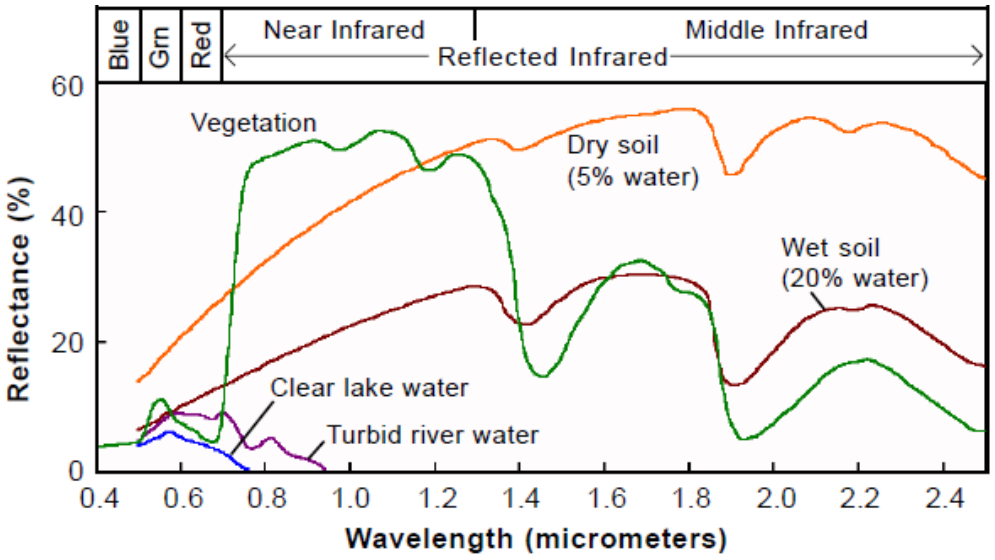


FIGURE 1.1: Spectral signatures of different materials (image extracted from [1]).

represented in the X-Y plane and the spectral information in the Z-direction, as shown in Figure 1.2 where Nr, Nc and Nb are the number of rows, columns and bands, respectively.

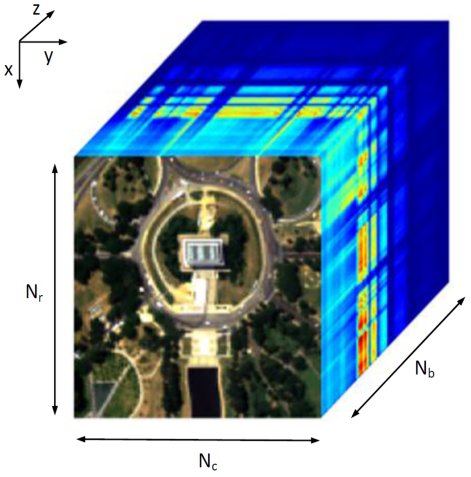


FIGURE 1.2: Spectral data cube (image extracted from [2]).

In terms of the geometrical properties of a spectral image, the spatial resolution refers to the size of each pixel in the X-Y plane, which is a function of the field of view (FOV) of the remote sensing imaging system, while the spectral resolution is determined by the bandwidth of the spectral bands. Since images are spatially and spectrally sampled by digital sensors, it is also needed to define an extra resolution measurement named radiometric resolution. This corresponds with the sensitivity of the imaging system to discriminate very slight differences in the magnitude of the captured electromagnetic energy. In other words, the radiometric resolution is defined by the number of bits required

for storing the energy sensed in each spectral band of each spatial pixel. The higher radiometric resolution of the sensor, the more sensitive it is to detect small differences in the reflected energy and the more bits are required.

While the radiometric resolution of the spectral imaging sensors is typically set to 8 to 16 bits per pixel and per band, the spatial and spectral resolutions widely vary depending on the sensor and the targeted application. For this reason, spectral images are typically classified according to their spectral resolution and number of spectral bands as multispectral, hyperspectral and ultraspectral images. Although there is is not one mind of issues of how to stablish the differences between them, HSIs are defined in this work as those that collect at least 100 spectral bands of less than 20 nm width. On the contrary, multispectral sensors are referred to as those collecting less than 20 non-contiguous spectral bands [56]. For this reason, hyperspectral sensors are more sensitive to subtle variations in reflected energy since it is measured in narrower and more numerous bands than multispectral sensors. This feature provides an almost continuous spectral measurement across the entire electromagnetic spectrum, allowing the detection of even slight differences among spectra.

Although the spatial resolution is not considered for making a distinction between hyperspectral and multispectral images, there is a trade-off between the increment in the number of bands of the spectral images and the spatial resolution. Normally, a higher spectral resolution is linked to a reduction in the spatial resolution. Hence, multispectral systems typically provide higher spatial resolutions than hyperspectral ones. Nevertheless, for certain applications, there is a need of a really high spectral resolution and hence, a larger number of spectral bands but, on the downside, with a lower spatial resolution. These systems, which typically collect thousands of bands, are classified as ultraspectral imaging systems.

Finally, most spectral imaging systems work in a wavelength range from the visible to the infrared, the latter commonly divided in bands called *Near InfraRed* (NIR), λ: 0.7 - 1.1 μm; *Short Wavelength InfraRed* (SWIR), λ: 1.1 - 3.0 μm; *Medium Wavelength InfraRed* (MWIR), λ: 3.0 - 5.5 μm; and *Long Wavelength InfraRed* (LWIR), λ: 7.7 - 14 μm.

1.2.2 Data collection systems

Hyperspectral data collection systems typically involve some forms of time-sequenced imaging. The capturing process can be accomplished by either a time sequence of 2D

spatial images at each spectral band of interest, or a time sequence of cross-track one-dimensional line of spatial pixels with all their spectral bands. Pushbroom and whiskbroom scanners are based on the second approach where the entire HSI is obtained joining the one-dimensional spatial images collected over time. They are the most widely used sensors in remote sensing applications, since they take advantage of the movement of the aircraft, the satellite or the UAV that carries them for capturing the data.

Figure 1.3 graphically describes the sensing process with these kinds of sensors. As it can be noticed, whiskbroom scanners (Figure 1.3 a) use a rotating mirror to scan across-track and to record the data one pixel at a time, that is, pixel-by-pixel and line-by-line. On the contrary, pushbroom scanners (Figure 1.3 b) collect the data along-track using a row of sensors arranged perpendicular to the flight direction, that is, the data are collected line-by-line.

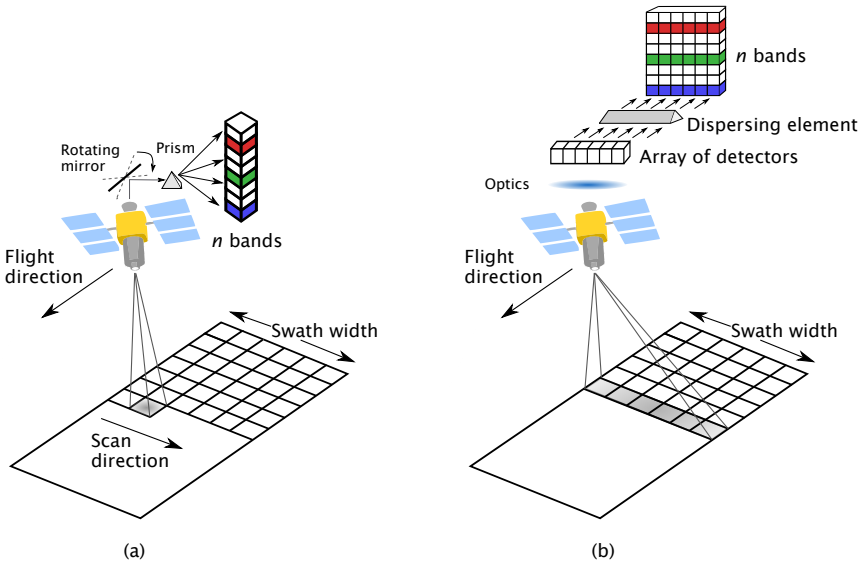


FIGURE 1.3: Remote sensing scanners. a) WhiskBroom b) Pushbroom (images extracted from [3]).

1.2.3 Hyperspectral data analysis methods and applications

As it was already explained, hyperspectral imaging provides a high spectral resolution by decomposing the reflected incident light on an observed scene into a large number of very close spectral bands, so that the spectra of different objects present a nearly continuous shape. This characteristic turns the spectral analysis into an effective technique for distinguishing different objects, especially those land cover types with visually indiscriminate

appearances, which enables a myriad of applications requiring fine identification of materials or estimation of physical parameters. To do this, the majority of hyperspectral data analysis methods and algorithms for image processing in hyperspectral remote sensing can be categorized as follows:

1. Anomaly and target detection:

   Hyperspectral imagery has been used in reconnaissance and surveillance applications where targets of interest are detected and identified. In some applications, there is a prior knowledge about the spectral characteristics of the desired targets. Target detection consists in searching known spectral signatures of interest, for example from a database. In these situations, the target spectral characteristics can be defined by a single target spectrum or by a signal subspace obtained from a spectral library or from a set of training data. The most employed algorithms within this field are the Constraint Energy Minimization (CEM) algorithm, the Adaptive Coherence/Cosine Estimator (ACE) detector and the Matched Filter (MF) [57].

   On the contrary, anomaly detection is a generalization of target detection where there is no prior information about the target signature [58], [59]. Anomalies are regarded as a group of rare pixels whose spectral signature differs significantly from their surroundings, i.e., those pixels that differ from the background pattern and whose existence may be indicative of abnormal or suspicious behaviour. Hence, anomaly detection could be seen as a binary classification problem where targets are scarce while the background class is predominant. In particular, anomaly detection methods extract characteristic features from the background, considering as anomalies those groups of pixels that differ more from the pattern. To do that, anomaly detectors mark every image pixel with a certain score, considering as rare pixels those whose values are higher than a threshold. In the recent decades, several anomaly detection algorithms have been proposed. The well-known Reed-Xiaoli (RX) [60] algorithm is one of the first developments in this field, being viewed as a benchmark to which other methods are compared. Several variations of the RX detector have been proposed in the literature in order to improve its performance [44]. [61]. Alternatively, different non-RX-based algorithms have been also developed in the recent decades, which can automatically estimate the background without assuming a normal probability function. Concretely, the sparse representation [62, 63] and the collaborative representation [64] have received considerable attention in the field of hyperspectral anomaly detection.

2. Compression:

   Limitations in the transmission bandwidth of the data link between the remote sensing platforms to the Earth surface jointly with the large volume of data sensed by the latest-generation sensors have positioned the data compression as one of the hottest topics in the hyperspectral imaging research community in the last decades. The compression process consists in modifying, encoding or converting the bits structure of the original hyperspectral data in such a way that it reduces the data volume to be stored, transmitted and processed. In general terms, lossless or near-lossless image coding are usually preferred in onboard compression scenarios in order to ensure a high fidelity of the reconstructed images. Nevertheless, the former techniques provide compression ratios that could be considered as moderate [65, 66]. In this scenario, limited bandwidth and ever-growing data rates make compulsory to reach higher compression ratios, which forces moving from lossless to lossy compression solutions.

   Generally, hyperspectral compression methods consist of a spatial and/or a spectral decorrelator, a quantization stage and an entropy coder. The decorrelator can be transform-based or prediction-based. In the former approach, a transform like the Discrete Wavelet Transform (DWT), Kahrunen-Loève Transform (KLT), or Principal Component Analysis (PCA) is utilized to decorrelate the data, whereas in the latter, the samples are predicted from neighbouring samples (in the spectral or spatial domains), and the prediction errors are encoded [3]. Transform-based methods are mostly preferred for lossy compression and normally are based on 2D compression techniques coupled by an additional transform in the spectral domain. A popular approach is to apply a one-dimensional spectral decorrelator, such as KLT or the DWT, followed by the Joint Photographic Experts Group (JPEG) 2000 standards[67–69]. However, this methodology is considered a too complex solution that requires significant computational resources, which are not available onboard. In this scenario, predictive coding paradigm represents an alternative that presents a good trade-off between performance and computational complexity. Although they are preferred for lossless compression, near-lossless or lossy compression can be achieved by means of selecting an appropriate quantization method. In this context, the Consultative Committee for Space Data Systems (CCSDS) has been working to develop a family of recommended compression schemes following a general paradigm of low complexity requirements, including lossy extensions of the standards [70–72].

3. Dimensionality reduction:

The extremely high dimensionality and size of the hyperspectral data, resulting from the improved spatial, spectral and temporal resolutions provided by the new-generation hyperspectral sensors, makes the processing of HSIs a complex and computationally expensive task. Additionally, the high number of bands causes the cursed phenomena or Hughes effects, which is intimately bound up with high dimensional spaces where the available data become sparse [73]. This sparsity is problematic for any method that requires statistical significance. In order to address this issue, the amount of data needed to support the results must grow exponentially with the dimensionality.

In this context, a dimensionality reduction of the hyperspectral vectors can highly facilitate the analysis afterwards. As it has been already mentioned, HSIs are spectrally and spatially smooth and as a consequence, features of HSIs normally reside in a subspace that normally has a much smaller dimension than the original number of spectral bands. On this basis, the goal of dimensionality reduction is to obtain an image with a reduced number of bands while trying to preserve the most useful spectral information as possible. In this context, dimensionality reduction consists in computing an spectral subspace in order to facilitate subsequent processing tasks, such as classification. Traditionally, methods based on Principal Component Analysis (PCA) are the most employed ones to decorrelate the spectral information and project the data in a lower dimensional manifold [74].

4. Unmixing:

   There is always a trade-off between the spatial and spectral resolution of an image sensor. HSIs offer a high spectral resolution in return for a limitation in the spatial resolution, which is normally less than the size of most land object types, and so, spectrally mixed pixels exist. Hence, the signal recorded by a hyperspectral sensor at a given band and from a given pixel is a mixture of the "light" scattered by the constituent substances located in the respective pixel coverage [55]. Therefore, it is common that different materials coexist at a subpixel level within the image. This results in pixels whose spectral signature does not directly correspond with any of the spectral signatures of the materials that conform a particular pixel, but with a mix of them. As a consequence, in this scenario is not anymore possible to determine what materials are present in pixels directly from the measured spectra. Hence, the discrimination of materials based on their unique spectral response is compromised.

In this context, the hyperspectral unmixing paradigm provides a solution to the spectral mixing modeling issue, which consists in determining the spectrally pure components, also called endmembers, present in mixed pixels as well as the amount of spectral information collected by each image pixel that can be represented by each endmember, also called abundances. In this way, the image interpretation can be deepened into a subpixel level, which is very useful for accurate land object mapping and the physically spectra inversion. In the last decades, hyperspectral spectral unmixing has become one of the hottest topics in the information extraction field by remote sensed images and thus, many different contributions have been uncovered. Foremost among them are the Pixel Purity Index (PPI) [75], Vertex Component Analysis (VCA) [76] and the N-FINDR [77] as algorithms for the extraction of the endmembers and, the Fully Constrained Least Squares linear spectral Unmixing (FCLSU) [78] and the Spectral Unmixing by variable Sitting and Augmented Lagrangian (SUnSAL)[79] as methods for the calculation of the abundance fractions.

5. Classification:

   HSIs are spectrally and spatially smooth. For this reason, neighbouring spectra generally correspond to similar materials and thus, they can be grouped to characterize materials. This process can be done by classification techniques. In the hyperspectral imagery domain, classification tries to discriminate different materials assigning an unique label to each pixel vector. Therefore, the spatial resolution is a quite important feature for classification techniques, as their main assumption is that the spatial resolution of the data is high enough to assume that pixels are mostly represented by a single predominant spectral signature. In the opposite scenario, in which data mostly contain mixed pixels, it is preferable to use spectral unmixing techniques to perform the analysis [55].

   Supervised classification has been more widely used in the literature, and in particular, the Support Vector Machine (SVM) algorithm [80] has been the most popular adopted solution among those offered by the state-of-the-art. However, the supervised techniques faces several challenges related with the high dimensionality of the data and the limited availability of training samples. A great deal of research has aimed to find more efficient ways to overcome this problem. Recent research methodologies to solve these issues are feature mining [74], subspace-based approaches [81] and semi-supervised learning techniques [82]. Additionally, it also exists the unsupervised learning, also named clustering, that generates clusters based on similar

spectral characteristics inherent in the image. Then, the data is classified in each cluster without providing training samples of its own.

6. Change detection:

   With the numerous hyperspectral missions to be launched in the near future, the number of multitemporal HSIs taken over the same geographic scene will significantly increase. The use of remote sensing data acquired at different times permits to detect changes in land cover, which has proven to be a valuable tool for monitoring, inspection and security. Change detection techniques employ the intrinsic spectral signatures of materials to identify and discriminate changes among them associated with significant spectral variations in the spectrum of the data captured at different periods of time. To do this, multitemporal analysis employs each single time image and the correlation between HSI pairs to detect potential temporal changes. Multitemporal strategies can be classified in three main types: image-comparison operators, image-stacking approaches and independent image analysis. Nevertheless, change detection applications nowadays still pose many open issues, such as handling of the high dimensionality and redundancy of the hyperspectral data, the implementation of accurate preprocessing techniques, the development of highly representative change features from the high-dimensionality spectral channels, and the effective design of unsupervised and automatic change detection algorithms [83].

1.2.4 Acceleration through parallel computing platforms.

FPGAs and GPUs are extensively used in a wide range of hyperspectral imaging applications as parallel computing devices in charge of the execution and acceleration of the targeted analysis techniques to be onboard carried out. In the space domain, FPGAs have consolidated as the standard choice for onboard remote sensing processing due to their smaller size, weight and power consumption, as well as for the existence of radiation-hardened and radiation-tolerant FPGAs [38, 84]. However, these devices are more expensive, physically larger and are often technology generations behind in both performance and functionality than their commercial counterparts [38, 39]. Due to this reason, the nowadays trend for small satellites is to use Commercial Off-The-Shelf (COTS) onboard electronic devices. Moreover, commercial FPGAs based on Static Random Access Memory (SRAM) are attracting attention because of its reconfigurable capabilities and low cost compared to Application Specific Integrated Circuits (ASICs) [85]. Nonetheless, the

use of COTS devices implies the necessity of applying mitigation techniques in order to increase the robustness of the application performance in environments exposed to radiation. In this sense, different Radiation Hardened By Design (RHBD) strategies have been developed over the years to protect the FPGA-based designs against radiation [86–88], such as Dual Modular Redundancy (DMR) schemes thought to detect errors and Triple Modular Redundancy (TMR) designs for error masking.

Regarding GPUs, they have evolved into a highly parallel, multithreaded, many-core processors with tremendous computational speed and very high memory bandwidth [55]. However, they exhibit very high power dissipation figures and are not radiance-tolerant, which has prevented their full incorporation to spaceborne Earth observation missions [55]. Fortunately, the emergence of computing boards that embed low energy consumption GPUs has made more attractive their use in power-constrained environments, especially in onboard applications carried out by UAVs [89–91]. In addition, the adoption of GPUs in space is also emerging rapidly due to the existing necessity of handling massive data in-orbit or in deep space [92]. Indeed, the National Aeronautics and Space Administration (NASA) and the European Space Agency (ESA) have conducted several studies for the investigation of heterogeneous systems based on embedded GPUs from a radiation perspective [39, 93–95]. As an example of a CubeSat with heterogeneous architecture is the NASA Hyperspectral Thermal Imaging (HYT) mission being integrated by University of Hawaii [93]. Nonetheless, the main drawback of these new emergent low-power GPUs (LPGPUs) is that they are not as high performing as their desktop counterpart, although many advances have been done in the last years in order to bring to market low-power embedded modules that deliver the capability of a desktop GPU workstation.

When implementing numerical algorithms on either of these computing platforms, we can choose to represent operands with different levels of accuracy. A trade-off exists between the numerical accuracy of arithmetic operators and the resources needed to implement them [96]. In modern computing, there are two major approaches to store and manipulate numeric representations of data, these are fixed-point notation and floating-point notation. In fixed-point representation, there are a fixed number of bits for representing the integer and fractional parts of the digital numbers. On the contrary, floating-point representation does not reserve a specific number of bits for any of the parts. Instead, the placement of the decimal point can float relative to the significant digits of the number through the definition of a certain number of bits for the number itself and a certain number of bits to stablish where the decimal place is within that number. As a consequence, floating-point assures a much larger dynamic range than fixed point, and hence, it normally yields much

greater precision [97]. Nevertheless, with fixed-point notation, the gaps between adjacent number are always equal and known before-hand whereas in floating-point notation, gaps are not uniformly spaced in such a way that smaller numbers have smaller gaps between them and vice-versa.

In this context, FPGAs are thought to perform concurrent fixed-point operations with a close-to-hardware programming approach, while GPUs are optimised for parallel processing of floating-point operations using thousands of small cores [98]. Indeed, it is widely accepted that floating-point designs lead to higher power usage compared to lower precisions [99, 100]. This remains true for FPGAs where floating-point implementations require larger amounts of FPGA resources than an equivalent fixed-point solution. Moreover, higher resource usage inherently leads to higher power consumption and ultimately increases overall cost of a design implementation [101]. On this basis, developed algorithms for imaging processing must take into consideration this hardware-design characteristics in order to be efficiently implemented in the platforms that better fulfils the requirements imposed by the targeted applications.

1.3 Motivations, research goals and contributions of this Thesis

1.3.1 Motivations of this Thesis

As it was already mentioned, HSIs are composed of a stack of 2D images where each one collects the reflectance of all image pixels for a narrow bandwidth of wavelengths. On this basis, the resulting data volume typically comprises several Gigabytes of data that have to be stored and processed, which consequently limits the performance of the hyperspectral imaging applications under real-time or near real-time constraints. In this context, several efforts have been directed in the last decades towards the acceleration of hyperspectral imaging calculations using high-performance computing infrastructures and parallel techniques. The latest advances in parallel programmable devices, such as FPGAs and GPUs, have bridged the gap towards the onboard analysis of the hyperspectral data [36]. However, although the computational capabilities of the next-generation commercial computing devices is expected to continue growing, hyperspectral sensors will also continue

increasing in spatial, spectral and temporal resolutions, which presents new emerging challenges in several applications with real-time processing requirements.

Additionally, the majority of algorithms for image processing in hyperspectral remote sensing has been traditionally addressed as independent entities being developed with the purpose of producing the best results, with independence of their mathematical complexity and how they could be later implemented into parallel computing platforms for providing real-time results. In this context, no matter how parallel computing devices increase their computational resources if there is a little room for speeding up the proposed algorithms employing parallel techniques due to the intensive computing of the involved operations and the so many data dependencies. For these reasons, several efforts have been directed in the last decades towards the hyperspectral imaging calculations using high-performance computing infrastructures. Nevertheless, this situation becomes even more stringent when multiple time-sensitive image processing techniques have to be executed. In this case, the simplest and most commonly used solution is to select a different mathematical algorithm for each particular case from the wide assortment of algorithmic solutions encountered in the literature and to be accelerated using parallel hardware devices. The problem arises in that they have to be sequentially processed onto the same computing device due to restrictions of power, weight and size.

All of the aforementioned issues, together with the huge amount of data contained in these images, limits the use of the hyperspectral technology in systems with limited available computational resources where multiple applications must be performed under tight latency/power/memory constraints. Under this scenario, the main motivation of this Thesis is to contribute to the field of onboard hyperspectral imaging processing when different time-sensitive applications coexist onto the same computing hardware device. In particular, it has been proposed a set of common core operations that extracts information from the HSIs useful for many applications. Thereby, this provides several benefits in the issue of the onboard processing. On the one hand, it means less cost, time and effort during the implementation phase of the independent hardware solutions since algorithms are based on the same mathematical method. Hence, designed blocks of logic or data (intellectual property (IP) core) in making FPGA-based solutions or routines compiled for GPUs (kernels) can be shared among all proposed algorithms. On the other hand, the proposed methodology permits the execution of many different tasks at the same time with the advantage of sharing the most computationally costly core operations, thus reducing the overall computational cost and the required hardware resources. All of this makes this

proposal an excellent solution for being implemented in low-power-consumption devices such as FPGAs and low-power graphic processing units (LPGPUs).

Additionally, the proposed set of core operations can be efficiently and independently applied to blocks of image pixels without requiring any specific spatial alignment. The vast majority of the state-of-the-art solutions need to work with the entire hyperspectral image cube. Unfortunately, this represents an important limitation for real-time applications, specially when using hyperspectral sensors based on pushbroom/whiskbroom scanners since lines of image pixels, also named hyperspectral frames, can be processed as soon as they are sensed. Therefore, it is mandatory to use causal algorithms where only the data samples up to the pixel being processed is used for the analysis while future data are not involved. Apart from this, the possibility of independently processing blocks of pixels as soon as they are sensed avoids storing and processing large amount of data, thereby reducing the amount of required hardware resources and also speeding-up the entire process.

1.3.2 Research goals of this Thesis

The main objective of this Thesis is to provide the research community with a set of common core operations that extract useful information from the HSIs for many applications. This objective includes developing new parallel hardware-friendly algorithmic solutions for hyperspectral imaging analysis under real-time constraints. The fact of being based on the same mathematical method is especially favourable and beneficial for their acceleration in different parallel devices. On the one hand, it saves cost, time and effort during the implementation phase of the different hardware solutions. On the other hand, it permits the execution of many different tasks at the same time with the advantage of sharing the most costly core operations, thus reducing the overall computational cost and the required hardware resources. Additionally, the proposed methodology shows low computational costs and it has been thought for being efficiently implemented in different highly parallel computing devices. In term, this objective is divided in six main goals which are detailed next.

1. Developing a new hardware-friendly set of core operations. These developed operations must be able to extract features from HSIs useful for many hyperspectral analysis techniques. In addition, they have to be efficiently implemented into parallel computing devices for applications under real or near real-time constraints using

both integer and floating-point arithmetic in order to be adapted better and faster to the end-user application. Furthermore, the proposed set of core operations must be able to process blocks of image pixels as they are sensed without requiring any specific spatial alignment with the purpose of obtaining a real-time performance in pushbroom/whiskbroom-based applications.

2. Demonstrating the feasibility of the concurrent execution of multiple hyperspectral analysis techniques based on the same mathematical method. It must be proved that multiple hyperspectral analysis can be simultaneously executed in the same hardware device reusing the outputs of the aforementioned set of core operations. Additionally, it must be also shown that the overall computational cost, the required hardware resources, as well as the invested human endeavours for the hardware implementations of the proposed algorithmic solutions are significantly reduced employing this methodology.

3. Developing a new hardware-friendly algorithm for anomaly detection based on the aforementioned set of core operations. The development algorithm must be able to find pixels that significantly differ from the background distribution fulfilling the requirements imposed by nowadays remote sensing applications and the causality required by applications based on pushbroom scanners.

4. Developing a new hardware-friendly transform-based algorithm for the lossy compression of HSIs based on the aforementioned set of core operations. The developed algorithm must be able to perform the spectral decorrelation and reduction stage of the compression process. Furthermore, a low complexity and a high degree of parallelism and scalability is desired in order to be suitable for the onboard applications based on pushbroom/ whiskbroom scanners.

5. Verifying the suitability of the developed algorithms for real-time applications by implementing them into different parallel devices, namely GPUs and FPGAs. Furthermore, it is desired to evaluate the efficiency of the different parallel devices according to the characteristics of the targeted applications and images. It is also desired to evaluate the accuracy of the obtained results using fixed-point notation in FPGAs and floating-point notation in GPUs.

6. Analysing the suitability of the aforementioned set of core operations for extending this methodology to other fields, such as target detection, band selection, classification and unmixing.

1.3.3 Contributions of this Thesis

The main contributions of this Thesis in the field of remote sensing for accomplishing the aforementioned objectives are detailed next:

1. A set of core operations, based on the well-known Gram-Schmidt orthogonalization method, has been proposed. This set of core operations is able to extract hyperspectral features useful for many hyperspectral analysis techniques, such as, anomaly detection, target detection, lossy compression, classification and unmixing. Hence, it permits the simultaneous execution of many different tasks at the same time with the advantage of sharing the most computationally intensive operations. Additionally, these operations feature low computational complexity since non-complex matrix calculations are involved and previously computed information is reused. As a novelty, the proposed set of core operations can be efficiently and independently applied on blocks of image pixels without requiring any specific spatial alignment, which makes our proposal a promising solution for real or near real-time applications specially when using hyperspectral sensors based on push-broom/whiskbroom scanners. In addition, this characteristic avoids storing and processing large amount of data, thereby reducing the amount of required hardware resources and also speeding-up the entire process. Finally, this set of core operations can be efficiently implemented in parallel hardware devices using both fixed-point and floating-point notation.

2. One algorithm, named A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD), has been developed. This algorithm is a subspace-based anomaly detector able to estimate the background distribution and the orthogonal subspace spanned by it where anomalous pixels are better detectable. This algorithm is based on the aforementioned set of core operations and can be applied in a line-by-line fashion.

3. The state-of-the-art Lossy Compression Algorithm for Hyperspectral image systems (HyperLCA), firstly defined in [2], has been optimized employing the set of core operations as it has been described in this Thesis, so that it can be more efficiently executed in parallel computing devices and using integer arithmetic. In general terms, the HyperLCA provides an efficient solution for performing the spectral decorrelation and reduction of hyperspectral and ultraspectral images for their compression. It is specially designed for independently processing small blocks of image pixels,

being a very suitable option for compressing HSIs when the execution times and hardware resources are limited.

4. The suitability of the proposed set of core operations for the concurrent execution of multiple hyperspectral imaging applications has been evaluated. In particular, we have focused on the lossy compression of HSIs and the detection of anomalous spectra. On the basis that the LbL-FAD and the HyperLCA are based on the same set of core operations, the outputs of the aforementioned operations can be computed just once and being reused for carrying out both processes. It results in many benefits in view of hardware acceleration for real-time or near real-time performance in terms of a reduction in the execution times, hardware resources and above all, in human endeavours. Concerning this latter, it implies the studio and analysis of only a single mathematical approach, which consequently permits to focus the efforts from a methodological and productivity points of view, resulting in a reduction in the time to market. To this end, we analyse different methodological strategies ranging from the independent execution of the targeted applications to the definition of two highly optimized versions for the joint implementation of both processes using a single configuration of the hardware utilisation, referred to as ADeLoC and HADeLoC respectively. Additionally, this methodology could be potentially extended to include other processes without a relevant increment of the required computational resources.

5. The LbL-FAD, the HyperLCA and the HADeLoC proposals have been implemented in different parallel hardware devices, namely GPUs and FPGAs. The accomplished implementations prove that real-time or near real-time results can be obtained by using algorithms thought from their conception to be implemented into parallel hardware. As a further research goal, the benefits of developing algorithmic solutions based on the same mathematical method in terms of reducing the execution-time, the hardware resources and the human effort have been also confirmed.

Further contributions achieved as part of the progress done during this work have been done. These contributions are detailed next.

1. The proposed set of core operations has been also evaluated in order to extend this methodology to other fields, such as target detection, unmixing, band selection and classification.

2. The developed algorithms have been assessed and compared against other state-of-the-art algorithms for the same tasks, with the purpose of evaluating not only their efficiency, in terms of speed and computational burden, but also the accuracy of the results. Several experiments have been carried out using many HSIs collected by different sensors as well as different evaluation metrics and state-of-the art algorithms.

3. The LbL-FAD, the HyperLCA and the HADeLoC algorithms have been implemented into different kinds of parallel hardware devices. This allows not only to verify the suitability of the algorithms for applications under real-time or near real-time constrains, but also to make interesting comparisons between the devices, their benefits and weaknesses according to the characteristics of the targeted applications.

The results that will be shown in the following chapters of this Thesis demonstrate the goodness of all these contributions for actual and future hyperspectral imaging applications under real-time or near real-time constrains.

1.4 Organization of this document

The present document is structured in seven chapters, including this introductory one dedicated to present the main motivations and goals of this Thesis work, and one appendix. A short outline about the subsequent chapters that make up this document is described next.

1.4.1 Chapter 2: Set of Core Operations

The onboard processing of remotely sensed images for on-the-fly making-decision applications has gained relevance in the last years. Nevertheless, the adoption of this onboard processing strategy brings further challenges for the remote sensing research community mainly related to the high data rate of the new-generation sensors and the onboard limitations in term of power budget and computational capacity. Consequently, there is an emerging trend towards the development of more hardware-friendly algorithms to cope with these exiting constraints. In this regard, this Chapter presents the set of core operations developed in this Thesis for the extraction of spectral features that are

useful for many hyperspectral analysis techniques, such as unmixing, compression and target/anomaly detection. Accordingly, it permits the concurrent execution of such techniques reusing operations and thereby, requiring much less computational resources than if they were separately executed. The basics about the mathematical methods behind these operations are analysed in detail in this Chapter. In addition, the computational complexity of this proposal is evaluated in terms of the number of basic operations involved in each algorithm step. Finally, the suitability of the proposed set of core operations for being executed using both integer and floating-point arithmetic is also verified through the definition of four different versions of them according to the data type and precision of the variables involved in the process.

1.4.2 Chapter 3: Hyperspectral Anomaly Detection

In this Chapter, the Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD) proposed in this Thesis work is fully described and documented. The LbL-FAD algorithm is a subspace-based anomaly detector that employs an orthogonal projection strategy, in particular the set of core operations extensively analysed in previous Chapter 2, for estimating the orthogonal subspace spanned by the background distribution where anomalous entities are better detectable. The goodness of the LbL-FAD algorithm for the detection of anomalous spectra has been evaluated in this Chapter using real hyperspectral images collected by different sensors. Additionally, the LbL-FAD method has been compared with other algorithms that conform the state-of-the-art for the anomaly detection problem. The obtained results clearly support the benefits of the proposed methodology, in terms of both the accuracy of the detection performance and the inherent computational complexity.

1.4.3 Chapter 4: Hyperspectral Lossy Compression

In this Chapter, a performance-enhancing version of the state-of-the-art Lossy Compression Algorithm for Hyperspectral Image Systems (HyperLCA) is proposed for the spectral decorrelation and compression of hyperspectral images. In this regard, the original version of the HyperLCA algorithm is modified in order to employ the set of core operations proposed in this Thesis work for the data transformation. Additionally, it also widens the definition of the operations performed by the *HyperLCA Transform* with the purpose of

analysing their suitability for being executed using integer arithmetic. The goodness of the HyperLCA algorithm for the lossy compression of hyperspectral images is evaluated using different quality metrics, such as the SNR, the MAD and the RMSE, among others, for different settings of the algorithm input parameters. Finally, it is demonstrated that the HyperLCA algorithm is able to preserve the most different pixels after the compression-decompression process, which is crucial for many hyperspectral imaging applications such as anomaly detection.

1.4.4 Chapter 5: Concurrent Execution of Multiple Hyperspectral Imaging Applications

The onboard processing of remotely sensed hyperspectral images for on-the-fly making-decision applications has gained momentum in recent years. Nonetheless, the adoption of this operation mode brings with it many huge challenges to be faced in the near future, mainly related with the increase of the sensor data rates and the limited amount of available onboard computational resources. Indeed, this situation becomes even more complex and cumbersome when different time-sensitive applications coexist, since different tasks must be sequentially processed onto the same computing device. As a contribution to this field of interest, we come up with a strategy based on the reuse of the set of core operations proposed in this Thesis work for the execution of several processing techniques. In particular, we analyse the potential of the suggested methodology towards the concurrent execution of multiple hyperspectral analysis processes, whilst optimizing the computational resources and the human endeavours invested during the implementation stage. In concrete, we focus on the issue behind the proposed HW-LbL-FAD algorithm and the HyperLCA method for the simultaneous detection of anomalous spectra and the lossy compression of hyperspectral images.

1.4.5 Chapter 6: Hyperspectral imaging acceleration through the utilization of embedded systems

Different algorithms for the processing of hyperspectral imagery have been proposed along the previous Chapters of this Thesis. All these algorithmic solutions share a set of common core operations based on projection techniques and more specifically, in the well-known Gram-Schmidt orthogonalization method. This Chapter verifies the benefits of developing

algorithmic approaches based on the same mathematical method in terms of reducing the execution-times, the hardware resources and the human endeavours. For doing this, the HW-LbL-FAD, the HyperLCA and the HADeLoC algorithms have been implemented into different kinds of parallel hardware devices, namely graphical processing units (GPUs) and field-programmable gate array (FPGAs). In this sense, the HW-LbL-FAD, the HyperLCA and the HADeLoC methods have been implemented on a Xilinx System on Chip (SoC) FPGA device, while the HyperLCA has been also accelerated in embedded computing boards from NVIDIA.

The proposed set of core operations takes into consideration the hardware-design characteristics of the most commonly used computing platforms. Accordingly, it is evaluated in this Chapter their adaptability to the requirements imposed by the targeted devices. Therefore, the aforementioned algorithms have been implemented in FPGA devices using integer arithmetic and the concept of fixed-point notation, while floating-point notation is exploited for GPUs-based systems.

1.4.6 Chapter 7: Conclusions and further research lines

This Chapter summarizes the main contributions of this Thesis and proposes further research topics, which are expected to complement and enhance the future development of this work.

1.4.7 Appendix A: Application of the proposed methodology to other hyperspectral image processing research fields

Future research lines are focused on extending the use of the set of core operations proposed in this Thesis work to other fields such as, band selection, target detection, unmixing and classification. In this Appendix, we briefly illustrate the first approximations towards the performance of these hyperspectral image analysis techniques.

Chapter 2

Set of Core Operations

The onboard processing of remotely sensed images for on-the-fly making-decision applications has gained relevance in the last years. Nevertheless, the adoption of this onboard processing strategy brings further challenges for the remote sensing research community mainly related to the high data rate of the new-generation sensors and the onboard limitations in term of power budget and computational capacity. Consequently, there is an emerging trend towards the development of more hardware-friendly algorithms to cope with these exiting constraints. In this regard, this Chapter presents the set of core operations developed in this Thesis for the extraction of spectral features that are useful for many hyperspectral analysis techniques, such as unmixing, compression and target/anomaly detection. Accordingly, it permits the concurrent execution of such techniques reusing operations and thereby, requiring much less computational resources than if they were separately executed. The basics about the mathematical methods behind these operations are analysed in detail in this Chapter. In addition, the computational complexity of this proposal is evaluated in terms of the number of basic operations involved in each algorithm step. Finally, the suitability of the proposed set of core operations for being executed using both integer and floating-point arithmetic is also verified through the definition of four different versions of them according to the data type and precision of the variables involved in the process.

2.1 Rationale

Although the hyperspectral technology has been around for quite some time, it has received increasing attention in the last years, becoming one of the most powerful tools for the Earth observation. Its expansion and growing recognition have been largely propelled by the richness of spectral information collected by this kind of sensors along the electromagnetic spectrum. This feature has positioned the hyperspectral analysis as the mainstream solution for the study of land areas and the identification and discrimination of visually similar surface materials.

Nonetheless, hyperspectral image processing still poses several challenges due mainly to the management of large amounts of data. This affects, on the one hand, the real-time performance of this kind of applications and, on the other hand, the requirements in terms of onboard storage resources. The issue is further complicated by the incorporation to the market of the latest-generation hyperspectral sensors that are characterized by, on one side, higher spectral and spatial resolutions and, on the other side, greater frame rates that can produce continual or nearly continual streams of higher dimensional data [102]. All of this makes the efficient data handling, from an onboard processing, communication, and storage points of view, even more challenging [102, 103].

Consequently, on-Earth processing has been the mainstream solution for remote sensing applications that sense hyperspectral images (HSIs). In this regard, images sensed by Earth observation (EO) platforms aboard satellites or manned/unmanned aerial vehicles are traditionally downloaded to the ground segment for being off-line processed on supercomputing systems. However, this operating mode could jeopardize the real-time response of time-sensitive applications due to the ever-growing acquisition data rates of the latest-generation sensors and the bottleneck represented by the limited communication bandwidth of the downlink system. Accordingly, the onboard processing has established as a potential solution for such restrictive scenarios.

Against this backdrop, there is a need in the literature for new algorithmic solutions that take into consideration the above mentioned currently existing constrains imposed by nowadays remote sensing applications from the earliest stage of development. Additionally, the causality inherent to real-time frameworks based on pushbroom/whiskbroom scanners must be also met through the definition of non-global algorithms capable of independently processing blocks of image pixels. In turn, this prevents the storing and management of large data volumes, thereby reducing the computing resources and speeding

up the execution process. Regrettably, the algorithms traditionally proposed for hyperspectral analysis have been normally addressed as independent global entities that pay more attention to the mathematical method that better maximizes the results than to the viability of being executed in power-constrained environments. Moreover, the onboard execution of imaging processing is not fully viable nowadays, specially under real-time constraints when different time-sensitive applications coexist onto the same computing device. This is because the simplest and most commonly used solution is to select a different mathematical algorithm from the wide assortment of proposals encountered in the literature for each hyperspectral data analysis technique to be performed and then, to accelerate them using parallel computing devices. Hence, the problem arises when they have to be sequentially processed onto the same computing device due to restrictions in terms of power, weight and size.

In this context, a new algorithmic solution is proposed in this Thesis to meet the aforementioned requirements, paving the way for the real-time performance of the hyperspectral image processing. With this in mind, we put forward a set of core operations that extract features from the HSIs useful for many applications. Consequently, it permits the concurrent execution of many different tasks at the same time with the advantage of sharing the most computationally intensive operations. To do this, the proposed set of core operations is based on orthogonal projection techniques, specially on the well-known Gram-Schmidt orthogonalization method [104, 105]. This methodology also features low computational complexity since non-complex matrix calculations are involved and previously computed information is reused. The basics about the mathematical method behind this set of core operations are analysed in detail in this Chapter. Its extension to other fields, such as target/anomaly detection, hyperspectral lossy compression, classification and spectral unmixing, are addressed in the following Chapters of this document.

2.2 Background Notions

Hyperspectral imaging gathers a huge amount of spectral information for hundreds of continuous and narrow wavelengths along the electromagnetic spectrum. However, HSIs are spectrally and spatially smooth that means that nearby spectra and wavelengths are highly correlated [55]. For this reason, pixels within an HSI may be grouped according to their spectral similarities and represented as a combination of relatively few spectral signatures that are representative of each cluster. Consequently, features of HSIs normally

lie in a lower-dimensional subspace than the original number of spectral bands. The accurate identification of this subspace enables the representation of hyperspectral pixels in a much smaller dimension, which offers many benefits in terms of execution times and complexity as well as in data storage [81].

In this context, hyperspectral unmixing is nowadays an essential tool for analysing remotely sensed HSIs. It refers to any process that separates the pixel spectra from an HSI into a collection of constituent spectra, also called endmembers, and a set of fractional abundances, one set per pixel. The endmembers are generally assumed to represent the pure materials present in the image and the abundances at each pixel to represent the percentage of each endmember that is present in the pixel [2]. On this basis, the linear mixing model (LMM) is based on the idea that each captured pixel in a HSI, $\mathbf{HI} = \{\mathbf{r}_j, j = 1, ...np\}$, which is composed by np pixels, $\mathbf{r}_j$, and nb spectral bands, can be represented as a linear combination of a set of p reference spectral signatures, $\mathbf{e}_n$. This linear mixture model assumes that secondary reflections and scattering effects can be neglected from the data collection procedure, and hence, the measured spectra can be expressed as a linear combination of the spectral signatures of materials present in the mixed pixels. The LMM is described as shown in Equation 2.1, where $a_{\mathrm{j,n}}$ is the fractional area covered by each $\mathbf{e}_n$ in $\mathbf{r}_j$, commonly named abundance, and $\mathbf{n}_{\mathrm{j}}$ represents the noise contained in each image pixel, $\mathbf{r}_j$.

$$\mathbf{r}_{\mathrm{j}} = \sum_{n=1}^{p} \mathbf{e}_{\mathrm{n}} \cdot a_{\mathrm{j,n}} + \mathbf{n}_{\mathrm{j}} \tag{2.1}$$

Therefore, it can be concluded that a HSI can be represented as a function of some image pixels, $\mathbf{E} = \{\mathbf{e}_n, n = 1, ...p\}$ and their corresponding abundances, $\mathbf{A} = \{\mathbf{a}_{j,n}, j = 1, ...np, n = 1, ...p\}$, which can be derived from the projection of each image pixel onto each endmember [105]. For this reason, the linear hyperspectral unmixing process could be a preliminary step for many other hyperspectral processes, such as compression, classification, anomaly detection, target detection, etc., since it allows a better understanding of the scene under analysis [2]. In the following lines, a few broad features of the aforementioned hyperspectral processing techniques and their relation with the LMM are introduced:

- In the field of lossy compression for hyperspectral imagery, many of the recent proposed techniques are based on decorrelating transforms in order to exploit spatial

and interband redundancies present in HSIs [106]. A popular approach involves the combination of a two-dimensional (2D) transform in the spectral domain, such as the Karhunen–Loève transform (KLT), the discrete wavelet transform (DWT), or the discrete cosine transform (DCT), followed by a spatial correlator as the Joint Photographic Experts Group (JPEG) 2000 standard. In this context, the extraction of the most representative pixels of materials present in a scene also permits performing dimensionality reduction and thus, to compress the HSIs. Just to set an example, considering a HSI composed of np pixels with nb spectral bands, if these np pixels are projected onto a subset of the p most different pixels within the scene, the original np pixels can be represented as a linear combination of their projections onto those p pixels. Therefore, the original hyperspectral cube can be represented on a new subspace of dimension p, being $p << nb$, and hence, to be compressed. Additionally, the number of p selected pixels directly determines the compression ratio achieved in the compression process [107].

- For the detection of desired targets of interest, the LMM can be used to characterize the targets and the interfering background. In this context, a subpixel target is mixed with the background spectra resulting in an image pixel with a combined spectral signature. Therefore, subpixel target detection issue could involve some kind of linear separation of pixel constituent elements [108]. Unlike the spectral signature of interest that is known in advance, the background subspace is estimated from the HSI using statistical or geometrical techniques. A good statistical approximation of the background can be done using the eigenvectors of the hyperspectral cube correlation matrix. On the contrary, the extraction of the p most representative pixels of the background, also commonly referred to as undesired signatures, could arise as an accurate geometrical description of the background distribution that will be used later to annihilate the spectral information that does not belong to the desired target. In addition, orthogonal projection techniques may be also used to select the spectral bands that best differentiate the desired target and the background signatures in order to maximize the spectral differences between both classes.

- Classification of a HSI entails the identification of which pixels contain various spectrally distinct materials that have been specified by the user. As it can be noticed, the classification problem can be treated as the target detection issue where undesired target signature matrix is formed by the other target signatures that are

known but not wanted in the image analysis. Indeed, orthogonal-subspace projection techniques were the first approaches proposed to separate the desired target signatures in a signal detection model where the undesired targets were eliminated prior to the detection of the desired ones so as to improve the signal detectability [109, 110].

- Similarly, anomaly detection represents an unsupervised application of the target detection issue where the desired target signature to be found is a-priori unknown. In general terms, anomalies are groups of rare and scarce pixels whose spectral signatures significantly differ from their surroundings, that is, those pixels that do not match the background pattern and whose existence may be indicative of abnormal or suspicious behaviour. On this basis, anomalous pixels cannot be well represented by the background distribution and hence, their projections in the orthogonal subspace spanned by the background distribution is notoriously higher. In this scenario, the solution for the anomaly detection issue lies in modeling the whole background and subtracting it from every image pixel by means of orthogonal subspace projections. In this sense, the p most characteristic pixels within a HSI can be used to represent the background distribution. In addition, the projection separation statistic for an image pixel, $\mathbf{r}_j$, can be calculated using the orthogonal projection matrix, $\mathbf{P} = \mathbf{I} - \mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$, where $\mathbf{W} = [\mathbf{w}_n, n = 1, ..., p]$ is a matrix whose columns are the p projection basis obtained from the background samples [58]. Therefore, orthogonal projection strategies may be used to extract the p pixels most representative of the background and thereby, to also compute the orthogonal subspace to that spanned by the selected background samples where anomalous pixels are better detectable.

- Finally, change detection consists of detecting spectral variations in materials present in bi-temporal hyperspectral remote sensing images. Due to the reduced spatial resolution inherent to the hyperspectral sensors, pixels in HSIs are generally mixed by a set of materials present in the scene, which turns the detection of potential spectral variations into a challenging task. To solve this problem, multitemporal spectral unmixing has the potential to add subpixel information to the detection procedure. In this context, hyperspectral unmixing decomposes each pixel spectrum into a collection of abundances of the purest materials or endmembers present in a scene. As a consequence, variations in the abundance maps for each underlying material is an indication of changes in the material spectra with time. Additionally, change detection by unmixing also may provide the nature of the change since it can

be caused by an alteration in the abundances of pure materials in the pixel, by the emergence of a new endmember or by their disappearance. Several hyperspectral change detection approaches using unmixing algorithms have been proposed in the literature, such as [111–115].

The main conclusion to be drawn from the above is that many hyperspectral imagery processing techniques may be performed using the same mathematical methods. For this reason, in this Thesis we have proposed an unmixing-based strategy based on orthogonal projections as a competitive strategy to fulfil the particular requirements imposed by the different image analysis tools in hyperspectral remote sensing. On this regard, we have defined a set of core operations based on the well-known Gram–Schmidt orthogonalization process that allows the extraction of useful information for many different hyperspectral imaging applications, while at the same time, ensures a low computational complexity and a high level of parallelism.

2.3 The Set of Core Operations

In this Thesis, we have dealt with the issue around the concurrent execution of multiple hyperspectral analysis techniques in restrictive environments such as onboard scenarios. On this basis, we have defined a set of common core operations that permits to extract features from the HSIs useful for many applications. As a novelty, it enables the execution of many different tasks at the same time with the advantage of sharing the most computationally intensive operations. To do this, our proposal is based on orthogonal projection techniques as many state-of-the-art approaches for analysing HSIs [74, 109, 116–121]. Concretely, the proposed methodology employs a modified version of the well-known Gram-Schmidt orthogonalization method [104, 105]. This methodology features low computational complexity since non-complex matrix calculation is involved and previously computed information is reused.

Additionally, the proposed set of core operations can be efficiently and independently applied on blocks of image pixels without requiring any specific spatial alignment. This feature makes our proposal a promising solution for real-time applications, especially when using hyperspectral sensors based on pushbroom/whiskbroom scanners, since lines of image pixels, also named hyperspectral frames, can be processed as soon as they are sensed. In addition, this possibility avoids storing and processing large amount of data,

thereby reducing the amount of required hardware resources and also speeding up the entire process.

2.3.1 The Gram-Schmidt Orthogonalisation Method

The Gram–Schmidt method calculates the orthogonal projection of a vector, $\mathbf{e}_i$, to a set of vectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_j]$, with $j < i$, by subtracting the portion of the vector $\mathbf{e}_i$ contained in the directions spanned by the vectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_j]$. Consequently, the Gram–Schmidt method performs the orthogonalization of a set of independent vectors $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p]$ and brings as a result a set of orthogonal vectors $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$ and their normalized vectors $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p]$. Figure 2.1 displays the orthogonalization process performed by the Gram-Schmidt method for two vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$.

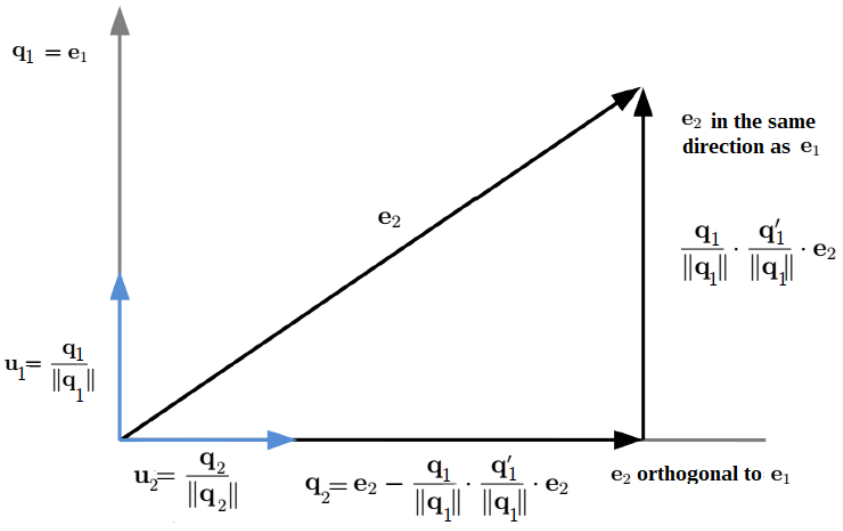


FIGURE 2.1: Example of the Gram-Schmidt orthogonalization for vector $\mathbf{e}_2$ with respect to vector $\mathbf{e}_1$; $\mathbf{q}_2$ represents the amount of information in $\mathbf{e}_2$ which is not contained in $\mathbf{e}_1$; $\mathbf{u}_1$ and $\mathbf{u}_2$ are unitary vectors with the directions spanned by $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively.

In this work, we employed a modified version of the Gram–Schmidt method [122] where vectors $\mathbf{u}_p$ are normalized dividing by the squared of its l^2-*norm*. In this regard, suppose two vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, the orthogonal component of $\mathbf{e}_2$ with respect to $\mathbf{e}_1$ is performed as shown Equation 2.2, where $\|.\|$ represents the l^2-*norm* and $\mathbf{u}^*$ is a unit vector. This equation is equivalent as that shown in Equation 2.3, where vector $\mathbf{u}$ is not now a unit

vector but is equivalent to $\mathbf{u}^*$. This modified version of the Gram–Schmidt method features low computational complexity since simple matrix operations are involved and also allows the reuse of previously computed information, which is reflected in speeding up the overall process.

$$\mathbf{e}_2 - \frac{\mathbf{q}}{\|\mathbf{q}\|} \cdot \frac{\mathbf{q}'}{\|\mathbf{q}\|} \cdot \mathbf{e}_2 = \mathbf{e}_2 - \mathbf{u}^* \cdot \mathbf{u}^{*'} \cdot \mathbf{e}_2, \quad \mathbf{q} = \mathbf{e}_1, \quad \mathbf{u}^* = \frac{\mathbf{q}}{\|\mathbf{q}\|} \tag{2.2}$$

$$\mathbf{e}_2 - \frac{\mathbf{q}}{\|\mathbf{q}\|} \cdot \frac{\mathbf{q}'}{\|\mathbf{q}\|} \cdot \mathbf{e}_2 = \mathbf{e}_2 - \frac{\mathbf{q} \cdot \mathbf{q}' \cdot \mathbf{e}_2}{\|\mathbf{q}\|^2} = \mathbf{e}_2 - \mathbf{q} \cdot \mathbf{u}' \cdot \mathbf{e}_2, \quad \mathbf{q} = \mathbf{e}_1, \quad \mathbf{u} = \frac{\mathbf{q}}{\|\mathbf{q}\|^2} \tag{2.3}$$

The pseudocode of the modified version of the Gram–Schmidt method is shown in Algorithm 1, where "′" represents the transpose of a vector. As it can be seen, the first orthogonal vector within $\mathbf{Q}$ is $\mathbf{e}_1$. In the second iteration, $n = 2$, $\mathbf{e}_2$ is projected on the direction spanned by $\mathbf{q}_1 = \mathbf{e}_1$, giving the projection vector $\mathbf{v}_1$ in Line 4. Then, it is subtracted from $\mathbf{q}_2 = \mathbf{e}_2$ in Line 5. As a result, $\mathbf{q}_2$ contains the spectral information spanned by $\mathbf{e}_2$ that cannot be represented by $\mathbf{e}_1$, i.e., orthogonal to $\mathbf{e}_1$. The process is repeated until the p vectors are orthogonal to each other.

Algorithm 1 Modified version of the Gram–Schmidt Orthogonalization

Inputs:
$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p]$
Outputs:
$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$ {Orthogonalized vector}; $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p]$ {Orthonormalized vector};
Algorithm:

1: **for** $n = 1$ **to** p **do**
2: $\quad \mathbf{q}_n = \mathbf{e}_n$
3: $\quad$ **for** $j = 1$ **to** $n - 1$ **do**
4: $\quad\quad \mathbf{v}_j = \mathbf{u}_j' \cdot \mathbf{e}_n$;
5: $\quad\quad \mathbf{q}_n = \mathbf{q}_n - \mathbf{q}_n \cdot \mathbf{v}_j$;
6: $\quad$ **end for**
7: $\quad \mathbf{u}_n = \mathbf{q}_n / (\mathbf{q}_n' \cdot \mathbf{q}_n)$;
8: **end for**

2.3.2 General Notations

In order to ease the understanding of the concepts introduced in the following Sections, it is needed to previously define some variables fully employed along the remainder of this Chapter 2. In the following, $\mathbf{HI} = \{\mathbf{F}_i, i = 1, ..., nr\}$ is a sequence of nr hyperspectral frames or lines of pixels, $\mathbf{F}_i$, comprised by nc pixels with nb spectral bands. Pixels within **HI** are grouped in blocks of BS pixels, $\mathbf{M}_k = \{\mathbf{r}_j, j = 1, ..., BS\}$, being normally BS equal to nc, or multiple of it, and k spans from 1 to $\frac{nr \cdot nc}{BS}$. $\hat{\boldsymbol{\mu}}$ is the average pixel or centroid pixel of each $\mathbf{M}_k$ block. $\mathbf{C} = \{\mathbf{c}_j, j = 1, ..., BS\}$ represents the centralized version of the input image block, $\mathbf{M}_k$. $\mathbf{E} = \{\mathbf{e}_n, n = 1, ..., p\}$ saves the p most different hyperspectral pixels extracted from each $\mathbf{M}_k$ block. $\mathbf{V} = \{\mathbf{v}_n, n = 1, ..., p\}$ comprises p vectors of BS elements where each $\mathbf{v}_n$ vector corresponds to the projection of the BS pixels within $\mathbf{M}_k$ onto the corresponding n extracted pixel, $\mathbf{e}_n$. $\mathbf{Q} = \{\mathbf{q}_n, n = 1, ..., p\}$ and $\mathbf{U} = \{\mathbf{u}_n, n = 1, ..., p\}$ save p pixels of nb bands that are orthogonal among them.

2.3.3 Description of the proposed Set of Core Operations

The proposed set of core operations is based on the aforementioned Gram-Schmidt method for extracting the p most representative hyperspectral pixels in a scene and also identifying the amount of spectral information that can be represented by them. It is worth stressing that these operations may be precisely performed by using the entire image or blocks of not spatially-connected image pixels since only the spectral information is analysed.

The first characteristic pixel, $\mathbf{e}_1$, to be extracted is the image pixel with the highest deviation from the average pixel, $\hat{\boldsymbol{\mu}}$. Afterwards, the orthogonal projection of each image pixel with respect to $\mathbf{e}_1$ is performed using the aforementioned Gram-Schmidt method. At this point, image pixels just retain the information that is not contained by $\mathbf{e}_1$ and thus, that is orthogonal to it. Once the first representative pixel has been selected, the proposed set of core operations sequentially extracts new characteristic pixels by selecting the pixels with the largest orthogonal projections to the pixels already extracted. With it, we achieve to select the most different pixels in each iteration, understanding as it, those pixels that cannot be well represented by previously selected pixels.

The proposed set of core operations is displayed in Algorithm 2 for an image block, $\mathbf{M}_k$. Operations from Lines 3 to 13 are repeated p times to extract the p most different pixels. First of all, the HSI block, $\mathbf{M}_k$, is centered in Line 2, obtaining $\mathbf{C}$ matrix by subtracting

the average pixel, $\hat{\boldsymbol{\mu}}$, to all pixels within $\mathbf{M}_k$. Secondly, pixels are sequentially extracted from Lines 3 to 13. In this process, the dot product of each frame pixel with itself is first calculated from Lines 4 to 6. In the remainder of this document, it is referred to as brightness of a pixel. Thirdly, the extracted pixels, $\mathbf{e}_n$, are selected as those pixels from $\mathbf{M}_k$ that correspond to the highest brightness in matrix $\mathbf{C}$, as shown in Line 8. Then, the orthogonal projection vectors, $\mathbf{q}_n$ and $\mathbf{u}_n$, are accordingly obtained as shown in Lines 9 and 10, respectively. After that, the information that can be spanned by the defined $\mathbf{q}_n$ and $\mathbf{u}_n$ orthogonal vectors is stored in the projected image vector $\mathbf{v}_n$ and subtracted to $\mathbf{C}$ in Lines 11 and 12. As it can be seen, Lines 2 and 7 of Algorithm 1 corresponds to Lines 9 and 10 of Algorithm 2, respectively. Similarly, operations shown in Lines 4 and 5 of Algorithm 1 corresponds to Lines 11 and 12 of Algorithm 2.

Algorithm 2 Set of core operations

Inputs:

$\mathbf{M}_k = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{BS}]$

Outputs:

$\hat{\boldsymbol{\mu}}$ {Average Pixel}; $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p]$ {Characteristic pixels}; $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$ {Orthogonalized vectors}; $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p]$ {Orthonormalized vectors}; $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p]$ {Projection vectors}

Algorithm:

1: Average pixel: $\hat{\boldsymbol{\mu}}$;
2: Centralization: $\mathbf{C} = \mathbf{M}_k - \hat{\boldsymbol{\mu}}$;
3: **for** $n = 1$ **to** p **do**
4: **for** $j = 1$ **to** BS **do**
5: Brightness Calculation: $\mathbf{b}_j = \mathbf{c}'_j \cdot \mathbf{c}_j$;
6: **end for**
7: Maximum Brightness: $j_{max} = \text{argmax}(\mathbf{b}_j)$;
8: Extracted pixels: $\mathbf{e}_n = \mathbf{r}_{j_{max}}$;
9: $\mathbf{q}_n = \mathbf{c}_{j_{max}}$;
10: $\mathbf{u}_n = \mathbf{q}_n / b_{j_{max}}$;
11: Projection: $\mathbf{v}_n = \mathbf{u}'_n \cdot \mathbf{C}$;
12: Subtraction: $\mathbf{C} = \mathbf{C} - \mathbf{q}_n \cdot \mathbf{v}_n$;
13: **end for**

In short, the proposed set of core operations are listed below:

1. Average pixel calculation, $\hat{\boldsymbol{\mu}}$:

Firstly, the average pixel, $\hat{\boldsymbol{\mu}}$, of the input image block, $\mathbf{M}_k$, is calculated (Line 1 of Algorithm 2).

2. Centralization:

   Then, $\mathbf{M}_k$ is centralized subtracting $\hat{\boldsymbol{\mu}}$ from each image pixel, obtaining the centralized version of the input image, $\mathbf{C}$ (Line 2 of Algorithm 2).

3. Brightness calculation, $\mathbf{b}_j$:

   The selected pixels are those with the highest dot product with itself in each iteration, also referred to as brightness of a pixel (Lines 4 to 6 of Algorithm 2). In this sense, $\mathbf{e}_n$ represents the selected pixel within the original image, $\mathbf{M}_k$ (Line 8 of Algorithm 2), $\mathbf{q}_n$ is its counterpart in $\mathbf{C}$ (Line 9 of Algorithm 2) and $\mathbf{u}_n$ is the normalized version of $\mathbf{q}_n$ (Line 10 of Algorithm 2).

4. Projection vector calculation, $\mathbf{v}_n$:

   Then, all vectors within $\mathbf{C}$ are projected onto $\mathbf{u}_n$, obtaining the projection vector, $\mathbf{v}_n$ (Line 11 of Algorithm 2).

5. Subtraction:

   To finish, the spectral information within $\mathbf{C}$ that cannot be represented by the selected pixel in the actual iteration, n, and that in consequence is orthogonal to it, is retained in $\mathbf{C}$ for the next iteration (Line 12 of Algorithm 2). For this reason, pixels within $\mathbf{Q}$ and $\mathbf{U}$ are orthogonal to each other.

2.4 Computational Complexity of the Set of Core Operations

In this section, the computational complexity of the proposed set of core operations is evaluated in terms of the number of operations (OPs) involved in each step of Algorithm 2. For clarity, OPs are simple calculations such as additions, subtractions, multiplications, and divisions. Additionally, Table 2.1 collects in summary the overall number of OPs required to process one block of image pixels, $\mathbf{M}_k$, composed of BS pixels.

In general terms, the proposed set of core operations encompasses 5 steps:

1. Average pixel calculation, $\hat{\boldsymbol{\mu}}$, (Line 1 of Algorithm 2):

   The calculation of the average pixel, $\hat{\boldsymbol{\mu}}$, involves the sum of each pixel element, $\mathbf{r}_{ji}$, for each particular spectral band, i, and then the division of each individual result by the total number of pixels, BS. This gives a set of $nb \cdot (BS + 1)$ OPs.

2. Centralization (Line 2 of Algorithm 2):

   In this step, the spectral information enclosed in the average pixel, $\hat{\boldsymbol{\mu}}$, is removed from each pixel within $\mathbf{M}_k$, which results in a total of $BS \cdot nb$ subtractions.

3. Brightness calculation, $\mathbf{b}_j$, (Lines 4 to 6 of Algorithm 2):

   The calculation of the brightness of one pixel corresponds to the inner product between two vectors of nb components. It is the most troublesome operation since it involves nb products and nb additions. This applied to all BS pixels of an image block, $\mathbf{M}_k$, leads to a total of $2 \cdot BS \cdot nb$ OPs.

4. Normalized vector calculation, $\mathbf{u}_n$ (Line 10 of Algorithm 2):

   The computation of each $\mathbf{u}_n$ vector requires to divide each nb component of the vector $\mathbf{q}_n$ by its brightness, which results in nb OPs.

5. Projection vector calculation, $\mathbf{v}_n$, (Line 11 of Algorithm 2):

   The calculation of each projection vector, $\mathbf{v}_n$, also corresponds to the inner product between two vectors of nb components, which implies $2 \cdot nb$ OPs. This applied to the BS pixels within an image block, $\mathbf{M}_k$, produces a total of $2 \cdot BS \cdot nb$ OPs.

6. Subtraction (Line 12 of Algorithm 2):

   The update of matrix $\mathbf{C}$ consists of firstly multiplying the projection vector, $\mathbf{v}_n$, of BS components with vector $\mathbf{q}_n$ of nb components, obtaining a matrix of $nb \times BS$ components, and secondly subtracting it to the remaining $\mathbf{C}$ matrix. Both steps require a total of $2 \cdot BS \cdot nb$ OPs.

Additionally, the set of operations involved from 3 to 6 are repeated p times, one for each reference pixel extracted per image block, $\mathbf{M}_k$, reaching a total of $p \cdot (6 \cdot BS \cdot nb + nb) + 2 \cdot nb \cdot BS + nb$ OPs. However, the main advantage of the proposed set of core operations is the use of simple mathematical operations that eases the posterior hardware implementation. In this context, this methodology performs exactly the same operations on each image pixel in each iteration, without needing any spatial or spectral information

from neighbouring pixels, thus avoiding creating data dependencies. Besides that, blocks of image pixels can be independently processed, thus reducing the amount of data to be processed and transferred and the computing hardware resources as well, such as memory, power and computational capabilities.

Steps	Number of OPs	Complexity
Average Pixel, $\hat{\boldsymbol{\mu}}$	$nb \cdot (BS + 1)$	$O(nb \cdot BS)$
Centralization	$nb \cdot BS$	$O(nb \cdot BS)$
Brightness, $\mathbf{b}_j$	$2 \cdot p \cdot BS \cdot nb$	$O(p \cdot BS \cdot nb)$
Normalized vector, $\mathbf{u}_n$	$p \cdot nb$	$O(p \cdot nb)$
Projection vector, $\mathbf{v}_n$	$2 \cdot p \cdot BS \cdot nb$	$O(p \cdot BS \cdot nb)$
Orthogonal information subtraction	$2 \cdot p \cdot BS \cdot nb$	$O(p \cdot BS \cdot nb)$
Total number of OPs	$p \cdot (6 \cdot BS \cdot nb + nb) + 2 \cdot nb \cdot BS + nb$	$O(p \cdot BS \cdot nb)$

TABLE 2.1: Number of OPs and computational complexity of the proposed set of core operations defined in Algorithm 2.

2.5 Data types and precision evaluation

One of the major benefits of the introduced core operations lies in the definition of a set of variables whose values are always within numeric ranges known before-hand. Consequently, it allows to fix in advance the maximum and minimum values of the results obtained in each operation. This feature makes possible to use the fixed-point concept in a custom way using integer arithmetic and bit shifting for representing the integer and decimal parts of the numbers. Therefore, the proposed set of core operations can be easily adapted for being more suitable for those hardware devices that are more efficient executing integer operations than floating point operations, such as FPGA devices.

In this section, we analyse the range of possible values to be reached by the variables involved in each operation in order to determine the required number of bits to be used for representing them with fixed-point notation. In summary, such variables are listed in Table 2.2. In addition, four versions of the set of core operations are also considered according to the data type and precision for representing image values stored in the centralized version of the input image block, $\mathbf{C}$. Efforts have been focused on this variable since it implies the major hardware resources consumption for its representation, more specially, $nr \cdot nc \cdot nb \cdot bd$ bits, being bd the number of bits used for storing each element

within $\mathbf{C}$. The proposed algorithm versions are referred to as *Float32*, *Int32*, *Int16* and *Int16-rd*, respectively. The *Float32*, *Int32* and *Int16* versions are developed for working with HSIs whose element values could be represented with up to 16 bits per pixel per band and 256 spectral bands as maximum. In particular, the *Float32* version employs single precision floating point arithmetic ($bd = 32$ bits) for storing $\mathbf{C}$. On the contrary, the *Int32* and *Int16* versions use customized fixed-point notation for representing fractional values within $\mathbf{C}$ with $bd = 32$ and 16 bits, respectively. Although the *Int16* version turns into a very interesting option for applications with limited available hardware resources, above all in terms of RAM memory, some precision losses are introduced in the operations, which affects the quality of the results. For this reason, we have made some performance-enhancing improvements to the *Int16* version, introducing the *Int16-rd* model. In this context, we have assumed that the available capturing sensor measures the incoming radiation using a resolution up to 12 bits per pixel per band (bpppb), though each image value is stored using $bd = 16$ bpppb, padding four zeros at the beginning of each sample. It is a common scenario in remote sensing applications [123, 124], besides, most of the hyperspectral data used as test-bench in subsequent Chapters of this document are packed in the same way as above mentioned.

In order to determine the number of bits to be used for representing the integer and decimal parts of the variables involved in the *Int32*, *Int16* and *Int16-rd* versions of the set of core operations, the range of their possible outcomes are analysed in detail in the following lines.

1. Average Pixel, $\boldsymbol{\mu}$:

   First of all, the set of core operations estimates the average spectrum, $\boldsymbol{\mu}$, of pixels within the input image block, $\mathbf{M}_k$, whose elements are saved using 12 or 16 bits depending on the resolution of the sensor that acquired the data. Hence, elements of $\boldsymbol{\mu}$ are in the same range that input image values, and so, it could be represented using the same number of bits, considering no decimal part.

2. Centralized image block, $\mathbf{C}$:

   Centralized image block, $\mathbf{C}$, represents an auxiliary copy of the input image block, $\mathbf{M}_k$, where the average spectral information has been removed. Accordingly, initial values stored in $\mathbf{C}$ are also in the same range that elements within $\mathbf{M}_k$ and $\boldsymbol{\mu}$. However, the set of core operations extracts pixels and the information that they are able to represent in every iteration, n, and hence, values of $\mathbf{C}$ elements tend to

decrease with the exception of certain unlikely situations in which they may increase. In order to set an example, consider two vectors a = *[0, 0.80, 0.80, 0.80, -0.80, -0.80, -0.80]* and b = *[0, 0.25, -0.25, 0.25, -0.25, 0.25 , -0.25]*, in which a is projected onto b and (-1,1] are the maximum and minimum values that can be represented with the available dynamic range. After the projection process, the orthogonal component of a with respect to b results in c = *[0, 0.53, 1.06, 0.53, -0.53, -1.06, -0.53]*. As it can be seen, overflow occurs since some elements exceed the limit values. For clarity, Figure 2.2 visually displays the example described above.

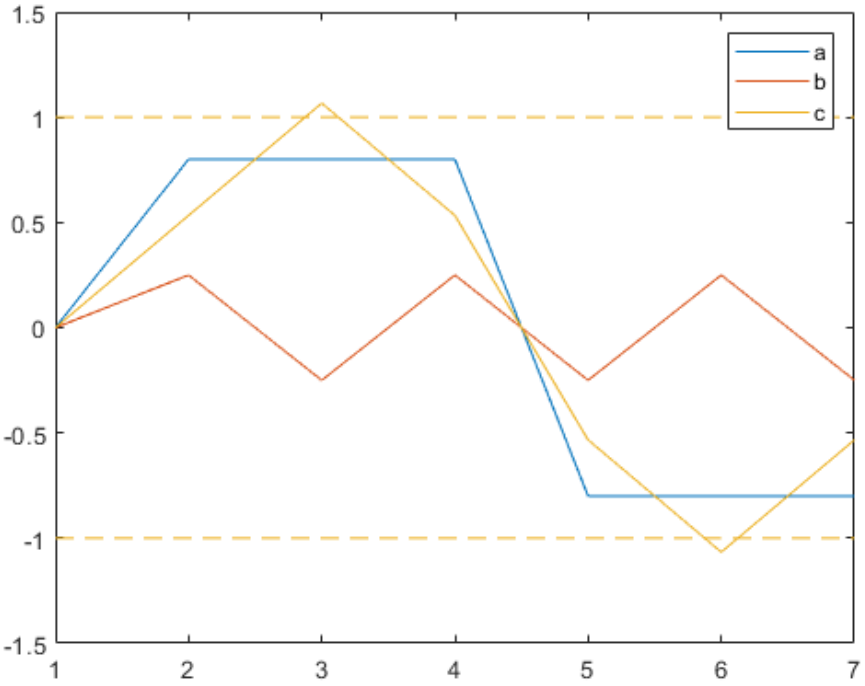


FIGURE 2.2: Example of unlikely situations where some vector elements increase their value after the orthogonal projection process. Vector a = [0, 0.80, 0.80, 0.80, -0.80, -0.80, -0.80] is projected onto vector b = [0, 0.25, -0.25, 0.25, -0.25, 0.25 , -0.25] and the maximum and minimum values to be represented with the available dynamic range is (-1, 1]. The result is vector c = [0, 0.53, 1.06, 0.53, -0.53, -1.06, -0.53], where some elements exceed the limit values

Due to this reason and in order to guarantee that the range of **C** values can be represented, 20 bits are used for representing the integer part of matrix **C** in the *Int32* version, which means that there would be a margin of 4 bits for those unlikely cases. Additionally, in order to be able to work with decimal numbers and increase the precision of the results, 12 bits are used for the decimal part, thus providing

a resolution of 2^{-12}. Accordingly, a total of 32 bits are used for representing the values of $\mathbf{C}$ in the *Int32* version.

In the *Int16* version, initial values of $\mathbf{C}$ are divided by 2 in order to avoid overflowing, what directly decreases the precision in one bit. On the contrary, this fact is discarded in the *Int16-rd* version since 2 extra bits are featured for these improbable situations. As it can noticed from Table 2.2, $\mathbf{C}$ image is stored employing 16 bits in the *Int16-rd* version as well as in the *Int16* version. However, it is assumed that image values are coded employing 12 bits as maximum instead of 16 bits. It permits to have 2 extra bits for representing the integer part of the fixed-point values of $\mathbf{C}$ elements. As a consequence, image precision is not altered as in previous *Int16* version for dealing with possible overflowing scenarios. Additionally, this new version also allows to have 2 bits for representing the decimal part of the fixed-point values of matrix $\mathbf{C}$, which is not possible in the *Int16* version.

3. Brightness, $\mathbf{b}_j$:

   The calculation of the brightness is the most troublesome operation, since it involves n_b products and $n_b - 1$ addition operations. In order to be able to accomplish this operation without overflowing, the brightness of each pixel, $\mathbf{b}_j$, is represented using 64 bits, 48 of them for the integer part and 16 for the decimal part, in the tree fixed-point versions.

4. $\mathbf{q}_n$ vector:

   Vectors $\mathbf{q}_n$ are directly selected as one pixel of $\mathbf{C}$. Hence, its values can be represented using the same number of bits as $\mathbf{C}$ in the *Int32* version, 20 for the integer part and 12 for the decimal part. Additionally, this configuration is also followed by both the *Int16* and *Int16-rd* versions in order to not lose precision and avoid overflowing in those operations that involve $\mathbf{q}_n$, such as those defined in Lines 10 and 12 of Algorithm 2.

5. Normalized vector, $\mathbf{u}_n$ vector:

   Vectors $\mathbf{u}_n$ are obtained by dividing the corresponding vector $\mathbf{q}_n$ by the maximum brightness value, $b_{j_{\max}}$ in each algorithm iteration, n, as it can be seen in Line 10 of Algorithm 2. Hence, its values are in the range (-1, 1] and therefore, elements of this vector may be represented using 32 bits, 2 bits for the integer part and 30 for the decimal part, obtaining a resolution of 2^{-30} for the three proposed fixed-point versions of the core operations.

6. Projection vector, $\mathbf{v}_n$:

   Vectors $\mathbf{v}_n$ contain the projection of the image pixels into the space spanned by the different orthogonal projection vector $\mathbf{u}_n$ in each iteration. Hence, their values are in the range (-1, 1] as $\mathbf{u}_n$ vectors. Therefore, its elements may be also represented using 32 bits, 2 bits for the integer part and 30 for the decimal part, for the proposed algorithm versions.

It is also worth to mention that the analysis and methodology presented in this work for generating the *Int32*, *Int16* and *Int16-rd* versions can be applied for generating any other integer version of the set of core operations, using between 16 and 32 bits, in order to achieve an optimal solution according to the specific characteristics of the sensor and the hardware device available for the image processing. Additionally, a higher number of bits can be also used if it is necessary to deal with images with a larger number of bands and/or with a higher bit depth.

Variable	Integer part			Decimal part			Total			
	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Float32
C	20	16	14	12	00	02	32	16	16	32
$\boldsymbol{\mu}$	16	16	12	00	00	00	16	16	12	32
b	48	48	48	16	16	16	64	64	64	64
q	20	16	14	12	00	02	32	16	16	32
u	02	02	02	30	30	30	32	32	32	32
v	02	02	02	30	30	30	32	32	32	32

TABLE 2.2: Number of bits used for representing the integer and decimal parts of the variables involved in the proposed set of core operations.

2.6 Conclusions

In this chapter, the issue around the real-time processing of HSIs has been approached from a new algorithmic perspective. Concretely, a set of core operations that extracts information from HSIs useful for many hyperspectral analysis techniques has been proposed. This methodology employs an orthogonal projection strategy and in particular, it is based on a modified version of the well-known Gram-Schmidt orthogonalization method. The

proposed method allows performing the simultaneous extraction of the p most representative hyperspectral pixels in a scene and identifying the amount of the image spectral information that can be represented by them. As it will be further analysed in next Chapters, the estimation of these data encourages the performance of many other hyperspectral processes, such as unmixing, compression, classification, anomaly detection and target detection.

In general, the methodology proposed in this Thesis entails the following advantages and benefits, above all in the field of onboard hyperspectral imaging processing:

- Line-by-Line performance.

  The proposed set of core operations is able to efficiently and independently process blocks of image pixels without requiring any specific spatial alignment. This feature makes this proposal a promising solution for real-time applications, especially those based on pushbroom/whiskbroom scanners since hyperspectral frames can be processed as soon as they are sensed. Additionally, it also reduces the amount of data to be stored and processed, thereby minimizing the required hardware resources and also speeding up the process of data analysis.

- Low computational complexity and high level of parallelism of involved operations.

  The proposed methodology does not perform complex matrix operations, such as inverse matrix calculation or the extraction of eigenvalues and eigenvectors, which makes easier its ulterior hardware implementation and reduces the amount of required hardware resources. In addition, this methodology allows the reutilization of information and avoids data dependences since spatial information is not involved. All of this contributes to reduce the time and effort during the stage of hardware acceleration.

- Reduction in the computing hardware resources.

  Since several hyperspectral image processing techniques may be performed using the proposed set of core operations, it enables the coexistence of multiple applications at the same time with the advantage of sharing the most computationally costly operations. Consequently, the overall computational cost and the amount of required hardware resources needed for their execution are considerably less than if different state-of-the-art algorithms were independently implemented.

- Fixed-point and Floating-point notation.

  The proposed set of core operations takes into consideration the hardware-design characteristics of the most typically used computing platforms, such as FPGAs and GPUs. Accordingly, they can be easily adapted to the requirements imposed by the targeted devices and thereby, be seamlessly implemented using both fixed-point and floating-point notation. In this context, FPGA devices are in general more efficient dealing with integer operations with a close-to-hardware programming approach, while GPUs are optimised for parallel processing of floating-point operations using thousands of small cores.

Chapter 3

Hyperspectral Anomaly Detection

In this Chapter, the Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD) proposed in this Thesis work is fully described and documented. The LbL-FAD algorithm is a subspace-based anomaly detector that employs an orthogonal projection strategy, in particular the set of core operations extensively analysed in previous Chapter 2, for estimating the orthogonal subspace spanned by the background distribution where anomalous entities are better detectable. The goodness of the LbL-FAD algorithm for the detection of anomalous spectra has been evaluated in this Chapter using real hyperspectral images collected by different sensors. Additionally, the LbL-FAD method has been compared with other algorithms that conform the state-of-the-art for the anomaly detection problem. The obtained results clearly support the benefits of the proposed methodology, in terms of both the accuracy of the detection performance and the inherent computational complexity.

3.1 Rationale

In the recent years, anomaly detection has experienced a steady surge in popularity in hyperspectral data analysis. Generally speaking, anomalies are considered as groups of rare and not abundant pixels whose spectral signature significantly differs from their surroundings. Thereby, anomaly detection is a generalization of target detection where there is no prior information about the desired target signature [58], [59], [125], [126]. Concretely, the anomaly detection issue could be seen as a binary classification problem where the background class is predominant and the existence of anomalous patterns may be indicative of abnormal or suspicious behaviour. This feature of uncertainty turns the anomaly detection into an essential matter in military and civilian applications, such as defense and surveillance, ecosystem disturbances, rare mineral discovery, among others.

In the recent decades, several anomaly detection algorithms have been proposed. Generally speaking, anomaly detection methods model the background distribution and classify those pixels with significantly different spectral signatures from the pattern as suspicious entities. To do so, anomaly detectors mark every image pixel with a certain score and point out those with the highest values as rare pixels. Therefore, anomalies could be seen as outliers in a distribution [127]. The most widely studied methods on hyperspectral anomaly detection are based on statistical approaches under the assumption of Gaussian multivariate distribution. Alternatively, sparsity and compressed sensing models have been recently emerged in the field of hyperspectral image processing. They do not assume a normal probability function or estimating a covariance matrix. Other anomaly detection methods are based on dimension reduction and feature extraction in order to remove inter-band correlations among spectral bands. Recently, with the development of deep learning methods, hyperspectral image processing and deep feature extraction have also made great progress [128].

Among this wide range of approaches found in the literature, the linear mixing model (LMM) appears as an interesting alternative to deal with the anomaly detection problem. As it was introduced in Chapter 2, the LMM is an extensively used tool in hiperspectral imagery analysis. It is based on the idea that a generic pixel spectrum, $\mathbf{r}_j$, can be represented as a linear combination of a set of reference spectral signatures, $\mathbf{E} = \{\mathbf{e}_n, n = 1, ..., p\}$, and the fractional parts covered by each $\mathbf{e}_n$ in $\mathbf{r}_j$, $a_{\mathrm{j,n}}$. Built on the premise that the anomalous target and the background signals lie into a different lower dimensional subspace in the field of subspace-based anomaly detection, the LMM can be rewritten as:

$$\mathbf{r}_{\mathrm{j}} = \sum_{n=1}^{p} \mathbf{b}_{\mathrm{n}} \cdot a_{\mathrm{j,n}} + \mathbf{s} \cdot a_{\mathbf{s}\mathrm{j}} + \mathbf{n}_{\mathrm{j}} \tag{3.1}$$

where $\mathbf{b}_{\mathrm{n}}$ represents the n background signal, $\mathbf{s}$ is the desired target signature, $a_{\mathbf{s}}$ is the abundance factor of $\mathbf{s}$ in the pixel $\mathbf{r}_{\mathrm{j}}$ and $\mathbf{n}_{\mathrm{j}}$ represents the noise contained in the image pixel $\mathbf{r}_j$.

A distinguishing feature of the anomaly detection problem is that the desired target signature, $\mathbf{s}$, to be detected is unknown beforehand but those image pixels spanned by $\mathbf{s}$ direction cannot be well represented by the background spectra. Therefore, the LMM may be effectively used to perform anomaly detection through the modelling of the background distribution and its subtraction from every image pixel by means of orthogonal subspace projections (OSPs). For this reason, subspace-based anomaly detection consists in projecting image pixels onto the subspace that is orthogonal to the one spanned by the background samples. In this context, the projection separation index for an image pixel $\mathbf{r}_{\mathrm{j}}$ is calculated as [58]:

$$\mathbf{d} = (\mathbf{r}_{\mathrm{j}} - \hat{\boldsymbol{\mu}}_b)' \cdot \mathbf{P} \cdot (\mathbf{r}_{\mathrm{j}} - \hat{\boldsymbol{\mu}}_b) \tag{3.2}$$

where $\hat{\boldsymbol{\mu}}_b$ is the estimated average pixel of the background samples and $\mathbf{P}$ is the matrix that projects the data onto the orthogonal subspace to one spanned by the background samples. In this case, $\mathbf{P}$ is defined by:

$$\mathbf{P} = \mathbf{I} - \mathbf{W}(\mathbf{W}'\mathbf{W})^{-1}\mathbf{W}' \tag{3.3}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{W} = \{\mathbf{w}_n, n = 1, ...p\}$ is a matrix whose columns are the p projection basis obtained from the background samples. An anomalous pixel is detected if its projection onto this orthogonal subspace is greater than a threshold.

The most common methods in the state-of-the-art to identify the background basis are based on linear transformations such as Singular Value Decomposition (SVD), Principal Component Analysis (PCA) or Independent Component Analysis (ICA) where the first eigenvectors of the background covariance matrix are representative of the background subspace. Another approaches are based on unmixing techniques to extract the background endmembers in an unsupervised manner [58], as it is proposed in this Thesis work

with the use of the proposed set of core operations introduced in Chapter 2. To this end, a new algorithm has been developed in this Thesis, named A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD) [129].

3.2 State-of-the-art in hyperspectral anomaly detection

Hyperspectral anomaly detection has been brought to the scientific community attention in the last years. Accordingly, several algorithmic solutions have been proposed in the literature. The well-known Reed-Xiaoli (RX) [60] is regarded as the benchmark in this field to which other newest algorithms are compared. The RX anomaly detector is based on the assumption that the background pattern follows a normal Gaussian distribution. Consequently, it employs the Mahalanobis distance to detect the anomalies by means of the computation of the background sample covariance matrix. Several variants of the RX detector have been investigated to improve its detection performance. For instance, the Local RX (LRX) locally estimates the background statistics by the definition of a dual concentric sliding window around each image pixel [130]. The Uniform Target Detector (UTD) and the RX work exactly the same, but the former uses the unit vector instead of a sample vector as the matched signature [44]. The impact of a regularization term added to the covariance matrix computation is analysed in [61] in pursuit of more stable background models. The Weighted-RXD and the Linear Filter-Based RXD [131] improve the estimation of the background statistics by removing some anomalous signatures and the noise when the background mean and the covariance matrix are computed. Additionally, kernel-based methods [132] have been also proposed in order to exploit the non-linear characteristic of the hyperspectral images (HSIs) and the highest-order correlation between the spectral bands.

One aspect to bear in mind is that the methodology followed by the RX-like solutions could be seen as an inverse operation of the Principal Component Analysis (PCA) [44]. As a matter of fact, the background distribution can be accurately modelled by the first PCA/SVD (Singular Value Decomposition) components defined by the eigenvalue decomposition of the background covariance matrix. On this basis, the subspace-based anomaly detectors emerge [58]. For instance, the Subspace RX (SSRX) [133] discards the first PCA/SVD bands and then globally applies the RX to the remaining subspace. On the

contrary, Orthogonal Subspace after RX (OSPRX) [134] projects the data onto the orthogonal subspace spanned by them before applying the RX.

Nonetheless, the background pattern could be, broadly speaking, very heterogeneous and thus, it is complicated to be represented assuming a normal probability function in some scenarios. Alternatively, different non-RX-based methods have been successfully applied to hyperspectral anomaly detection. Concretely, the unsupervised nearest regularized subspace (UNRS) [135], the collaborative-representation-based detector (CRD) [64], the low-rank representation models and the sparse-representation-based methods [62, 63] have been extensively researched in the field of hyperspectral anomaly detection. Due to the strong interband and spatial correlations of the observed data, these methods are based on the assumption that the background samples can be approximately represented by the spatial surrounding neighbourhood pixels or as combination of few reference vectors. For instance, the Low Rank and Sparse Representation anomaly detector (LRASR) [136] infers, on the one hand, that the background class could be modelled as the lowest rank representation of the HSI and, on the other hand, the anomalies can be potentially distinguished by the remaining information, which will be sparse since a tiny part of image pixels belongs to the anomalous class. On this basis, a background dictionary composed of representative spectral signatures extracted by a previously applied clustering method is built. Similarly, the LRASR-based Mahalanobis detector (LSMAD) [63] constructs a Mahalanobis-distance-based anomaly detection algorithm after recovering both the low-rank background matrix and the sparse matrix with the anomalous entities. With it, LSMAD algorithm tries to alleviate the anomaly contamination problem when the background mean and the covariance matrix are estimated. On the contrary, the CRD [137] exploits the concept that each background pixel can be approximately represented by its spatial neighborhoods whereas anomalies cannot be. To do this, a weight vector that collects the collaboration of these neighbourhood pixels is defined by the reinforce of its $l2\text{-}norm$ minimization, in which a Euclidean distance-weighted regularization matrix is included to adjust the contribution of each surrounding pixel.

Other state-of-the-art anomaly detection methods are based on projection techniques in order to carry out a dimensionality reduction for feature extraction, being the PCA and the Independent Component Analysis (ICA) the most commonly applied methods for this purpose. Global Iterative Principal Component Analysis Reconstruction-Error-based anomaly detector (GIPREBAD) [138] employs the PCA projections to examine anomalies by computing residual errors. As a matter of fact, it starts from the idea that the anomalies can be poorly reconstructed by the first principal components representative of

the predominant class in the image, the background, but dominate the trailing components with low variance. Autonomous Global Anomaly detector (AutoGAD) [139] follows an unmixing-like strategy to separate the independent mixed sources present in the image. In this sense, the data are projected onto a new set of independent axes given by the ICA and then, the abundance maps that possibly contain anomalous pixels are selected computing two measures of target characteristics.

In the last years, deep learning and tensor theory have drawn increasing attention for hyperspectral image processing, above all in the field of supervised classification [140–143] and to a minor extent for anomaly detection. As far as we know, the work presented in [144] was the first approach that applied deep convolutional neural network (CNN) for supervised hyperspectral anomaly detection. In this proposal, a reference data set with labelled samples is employed to train the CNN. Then, for each testing pixel, differences between pixel pairs constructed using its neighbouring pixels are classified by the trained CNN with the results of similarity measurement. However, supervised anomaly methods are not as applicable as unsupervised or semisupervised alternatives due to the lack of available labelled training data sets. In the field of unsupervised hyperspectral anomaly detection, the strategy of adaptive weighted coding for decreasing the effect of local anomalous pixels contamination using Deep Belief Network (DBN) model with auto-encoder structure is proposed in [145]. Additionally, a spectral adversarial feature learning (SAFL) model is presented in [146] to extract distinctive features in deep latent space whose quality is evaluated by means of reconstruction errors. Nonetheless, unsupervised techniques are very sensitive to noise and data corruption and thereby, the detection performance is diminished. In this scenario, semisupervised methods are adopted in this field. A generative adversarial network (GAN)-based model is proposed in [147] to estimate the background distribution in the spectral domain and also adopts a morphological attribute filter to generate an initial feature in the spatial domain. Alternatively, algorithmic solutions that fuse different computing strategies have also been emerging, for instance, [148] introduces a novel anomaly detection algorithm based on CNN, low-rank representation (LRR) and unsupervised clustering.

Nevertheless, most of the aforementioned proposals improve their detection accuracy by raising the intensity of the computation complexity, which is usually reflected in intensive memory requirements, high implementation costs and non-scalability. Just to list some of the most commonly used operations, they include covariance matrix, inverse matrix computation and eigenvalue decomposition. All of this poses some important limitations

that prevent the use of these methodologies for the onboard real-time processing of hyperspectral anomaly detection. In contrast, lowering the heavy computational costs is indeed taken into account by certain researchers through the use of alternative mathematical methods, such as QR and Cholesky decomposition [44–47], or through parallel computing via high-performance architectures. However, the latter is a very time-consuming task due to the low parallelizable nature of the involved operations.

In addition to the above issues, most of the mentioned methods must be fed by the entire HSIs to perform the anomaly detection process. This feature becomes them not competitive solutions for most applications based on pushbroom/whiskbroom scanners, in which the image is sensed in a line-by-line fashion, perpendicular to the flight direction of the capturing platform. Therefore, for paving the way for real-time detection performance, it is expected the search for anomalies as soon as new hyperspectral frames are sensed. This scenario imposes two necessities. On one side, it is mandatory to use causal anomaly detection algorithms, where only the data samples up to the pixel being processed is used for the analysis while future data are not involved [44]. On the other side, the proposed algorithmic solutions must be hardware-friendly in order to be easily executed in parallel computing devices. In this regard, the Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD) is proposed in this Thesis work with the goal of fulfilling the aforementioned constraints in terms of real-time performance and the causality required by applications based on pushbroom scanners.

3.3 Proposed anomaly detection algorithm: A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD)

The LbL-FAD algorithm is a subspace-based anomaly detection algorithm designed to fulfil the constraints imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners. In this regard, the LbL-FAD algorithm is able to independently process blocks of hyperspectral pixels with not taking into consideration any spatial alignment requirement. As a result, the methodology followed by the LbL-FAD algorithm is well aligned with the needs imposed by the aforementioned applications since the detection of anomalous pixels could be conducted in a line-by-line fashion.

The LbL-FAD algorithm is based on the concept of orthogonal subspace projections. As it was introduced in Section 3.1, anomalous spectral signatures significantly differ from the background pattern. Accordingly, they cannot be accurately represented by the subspace conformed by the background samples. On this basis, the methodology followed by the LbL-FAD algorithm focuses on the calculation of an orthogonal subspace to the one spanned by the background distribution in which the anomalous spectra are better distinguishable. To do so, the LbL-FAD algorithm firstly estimates a set of orthogonal vectors representative of the background model following an orthogonal projection strategy and, more specifically, by means of the set of core operations introduced in this Thesis work. Secondly, the LbL-FAD algorithm computes the orthogonal subspace matrix, $\mathbf{P}$, defined by Equation 3.3, that projects image pixels onto the orthogonal subspace to the one spanned by the background pattern. In this regard, two low computational complexity strategies are proposed, thereby avoiding the use of traditional linear transformation methods, such as the PCA or the SVD that are high computational complex in nature, nor the inverse of big data matrices.

Moreover, yet another contribution of this algorithm is the computation of an automatic thresholding that enables the real-time discrimination of anomalous pixels as soon as each hyperspectral frame is processed. In general, other causal state-of-the-art algorithms [44–47, 149], provide gray-scale maps as output where the anomalous target detection is usually carried out by visual inspection or thresholding after the whole image is processed. Unlike them, the LbL-FAD algorithm outputs a line-by-line binary map where anomalous pixels are segmented from the background, avoiding the subjective human factor.

To illustrate the workflow followed by the LbL-FAD algorithm, Figure 3.1 shows a graphic representation of the involved computing stages, which will be explained in detail in next sections.

3.3.1 Line-by-Line extraction of the background reference spectra

The first stage of the LbL-FAD algorithm, named *Stage 1* in Figure 3.1, consists in the feature selection of a set of reference spectra representative of the background distribution. To do so, the LbL-FAD detector employs the set of core operations described in Chapter 2, and more concretely in Algorithm 2, for the first n_f hyperspectral frames captured by the hyperspectral scanner. It is done under the assumption that these image blocks

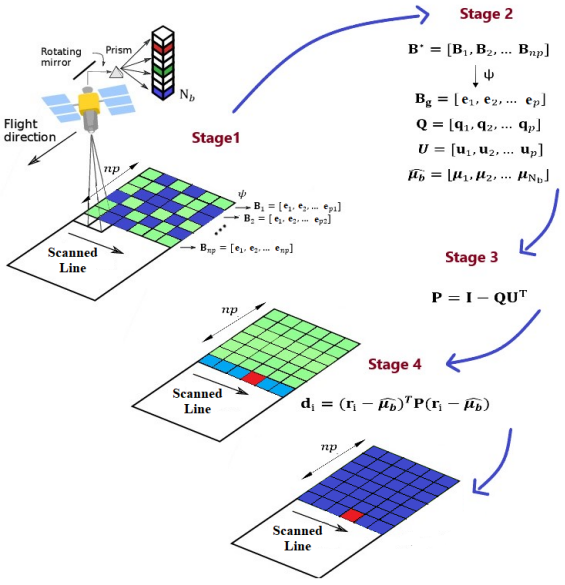


FIGURE 3.1: Diagram of the LbL-FAD stages. Stage 1: Line-by-line background spectra extraction. Stage 2: Overall background subspace estimation. Stage 3: Orthogonal subspace calculation. Stage 4: Detection of anomalies.

are fully representative of the background distribution and therefore, free of anomalous signatures. Therefore, the final goal of this algorithm stage is to select the most different pixels within each hyperspectral frame from the time they are captured.

Nevertheless, the key point of this stage is to accurately select the number of p pixels to be extracted from each hyperspectral frame. As it was explained in Chapter 2, each time that a pixel $\mathbf{e}_n$ is selected, the spectral information that could not be represented by the already extracted pixels remains in image matrix $\mathbf{C}$. It means that if the image is represented using the selected $\mathbf{e}_n$ pixels, according to the LMM, a small part of the spectral information is lost when the image is reconstructed using the p selected $\mathbf{e}_n$ pixels and besides, equal to the remaining information contained in $\mathbf{C}$. In this sense, the maximum brightness, $b_{j_{\max}}$, after the $\mathbf{e}_p$ vectors have been selected may be representative of the spectral losses introduced by the unmixing process and consequently, it could be used to set p. To this end, it is defined and extra input parameter, α, that represents the percentage of the spectral information that will be considered as noise. In this context, the pixel extraction process finishes when the loss, in percentage terms, is less than α, as it can be seen in Equation 3.4 where $(\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})$ represents the initial value of $\mathbf{r}_{j_{\max}}$ in $\mathbf{C}$. In general, experience has shown that a stop factor, α, fixed to 1% is sufficient since smaller values would mean that a greater number of characteristic pixels are unnecessarily extracted, and as a consequence, large computation times are promoted.

Finally, it is important to mention that enough n_f hyperspectral frames must be taken to ensure that all spectral variability is covered and thus, to generate a truthful background model. In addition, background samples obtained in previous flights may be used instead of obtaining them from the first n_f frames.

$$\frac{b_{j_{\max}}}{(\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})' \cdot (\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})} \cdot 100 < \alpha \rightarrow \text{Stop selecting } p \text{ } e_n \text{ pixels} \tag{3.4}$$

3.3.2 Overall background subspace estimation

One of the main novelties of the methodology followed by the proposed set of core operations and thus, by the LbL-FAD algorithm, is that image blocks, $\mathbf{M}_k$, are independently processed ruling out any spatial alignment restriction. For this reason, many pixels extracted from previous n_f hyperspectral frames actually represent the same entities. Therefore, it is required to obtain a subset of the most purest reference vectors that better define the background distribution. To this end, the set of core operations is applied once again, though input matrix, $\mathbf{M}_k$, is now replaced by a matrix $\mathbf{B^*} = \{\mathbf{E}_k, k = 1, ..., n_f\}$ whose columns collect the background reference vectors extracted from each first n_f frames.

Consequently, a subset of the p most representative background pixels $\mathbf{B_g} = \{\mathbf{e}_n, n = 1, ...p\}$ is obtained, as well as the orthogonal vector matrices $\mathbf{Q} = \{\mathbf{q}_n, n = 1, ...p\}$ and $\mathbf{U} = \{\mathbf{u}_n, n = 1, ...p\}$. Deriving from this, some background information is lost at the end of this step since $\mathbf{B_g}$ is actually a subspace of $\mathbf{B^*}$. As it was analysed in Section 3.3.1, the remaining maximum brightness, $b_{j_{\max}}$, in $\mathbf{C}$ could be an indicator of the amount of background information that is not well represented by the reference vectors within $\mathbf{B_g}$. Therefore, $b_{j_{\max}}$ obtained at the end of this stage could be potentially used as a benchmark to identify anomalous pixels. In later Sections, this parameter will be referred to as τ.

For the sake of clarity, this stage corresponds to *Stage 2* in Figure 3.1.

3.3.3 Orthogonal Subspace to the one spanned by the background samples

One of the fundamental issues posed by the anomaly detection process is the identification of desired spectral targets that are unknown in advance. Nevertheless, anomaly detection

is based on the premise that these rare signatures are notoriously different from the background pattern. For this reason, it is assumed that anomalous pixels should have a higher projection onto the subspace orthogonal to the background distribution. Consequently, the third stage of the LbL-FAD algorithm, named as *Stage 3* in Figure 3.1, focuses on the computation of the orthogonal subspace matrix, $\mathbf{P}$, defined by Equation 3.3. However, this calculation is computationally expensive since it implies matrix inverse whose dimension directly depends on the number of background samples p. On this basis, we propose two lower computational burden alternatives to address $\mathbf{P}$ computation. Both of them employ the orthogonal vectors $\mathbf{Q}$ and $\mathbf{U}$ estimated in Stage 2 of the algorithm. In particular, in this Section we discuss the first approximation while the second proposal will be later analysed in Section 3.4.

In this sense, when the Gram-Schmidt method is applied to a set of vectors, here $\mathbf{B_g} = \{\mathbf{e}_n, n = 1, ...p\}$, the orthogonalization process is performed by subtracting from each vector the spectral information that is not contained in the vectors already orthogonalized. For instance, $\mathbf{q}_3$ and $\mathbf{u}_3$ retain the spectral information present in $\mathbf{e}_3$ that is not part of $\mathbf{e}_1$ and $\mathbf{e}_2$. Nonetheless, $\mathbf{q}_2$ and $\mathbf{u}_2$ is also spanned by the direction of $\mathbf{e}_3$ but not by $\mathbf{e}_1$. As a result, $\mathbf{q}_p$ and $\mathbf{u}_p$ vectors only hold the spectral information of $\mathbf{e}_p$ that is not included in previously selected background reference vectors $\mathbf{B_g} = \{\mathbf{e}_n, n = 1, ...p-1\}$. On this basis, $\mathbf{Q}$ is equivalent to $\mathbf{W}$ in Equation 3.3 and $\mathbf{U}$ to $\mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}$. Hence, the first approximation for computing the orthogonal projection matrix, $\mathbf{P}$, can be defined as:

$$\mathbf{P} = \mathbf{I} - \mathbf{Q}\mathbf{U}' \tag{3.5}$$

3.3.4 Detection of anomalies

As a matter of fact, the detection of anomalous pixels is carried out in the fourth and the last stage of the LbL-FAD algorithm, labelled as *Stage 4* in Figure 3.1. To do so, pixels, $\mathbf{r}_\mathrm{j}$, within new captured frames, $\mathbf{M}_k, k > n_f$, are projected onto the orthogonal subspace spanned by the background model, represented by the orthogonal projection matrix, $\mathbf{P}$, and their projection, d, is measured as it is depicted in Equation 3.2.

On the basis that an anomaly represents a spectrally different material from those comprised by the background distribution, anomalous pixels should have a large projection onto the subspace spanned by Equation 3.2. In order to quantify how big must be this

projection, scalar τ obtained in Section 3.3.2 is used. As mentioned above, τ stands for the error committed by pixels within $\mathbf{B_g}$ to represent the background. Accordingly, τ is also the maximum brightness of the remaining spectral information within $\mathbf{C}$, so that, its value is already scaled in the orthogonal subspace spanned by $\mathbf{Q}$ and $\mathbf{U}$ vectors and therefore, by $\mathbf{P}$. As a result, an anomalous entity should have a larger projection, d, than τ.

As it was already mentioned in Section 3.1, an anomaly could be seen as an outlier in a distribution. An outlier is an observation whose value lies far away from the rest of the data. In a sense, its definition is fully connected to what is considered an abnormal distance from the overall pattern of a distribution. In this context, box plot diagrams, also termed as whiskers′ plots, are widely used in the literature to identify outliers. They display the behaviour of the data in the middle and extreme boundaries of the distribution. For doing so, they use the lower and upper quartiles and the interquartile range (IQ) for identifying extreme values, or fences, in the tails of the distribution beyond which abnormal values are considered. In this regard, a point beyond $1.5{\cdot}IQ$ on either side, termed as inner fence, is considered a mild outlier while a point beyond $3{\cdot}IQ$ on either side, termed as outer fence, is considered an extreme outlier. Following an analogous methodology in our particular application, an anomalous pixel could be defined as a mild outlier whose projection d is bigger than $1.5{\cdot}\tau$. This way, we are able to also propose an automatic thresholding method based on the background data distribution that allows to segment the anomalous targets just after a hyperspectral frame is processed.

The overall description of the LbL-FAD algorithm is summarized in Algorithm 3. The three-dimensional (3D) input hyperspectral cube, $\mathbf{HI} = \{\mathbf{F}_i, i = 1, ..., nr\}$, is a sequence of nr hyperspectral frames or lines of pixels, $\mathbf{F}_i$, comprised by nc pixels with nb spectral bands. Pixels within $\mathbf{HI}$ are grouped in blocks of BS pixels, $\mathbf{M}_k = \{\mathbf{r}_j, j = 1, ..., BS\}$, being normally BS equal to nc, or multiple of it, and k spans from 1 to $\frac{nr \cdot nc}{BS}$. The output is a binary map $\mathbf{X} = \left\{\mathbf{x}_{kj}, k = 1, ..., \frac{nr \cdot nc}{BS}, j = 1, ..., BS\right\}$ where anomalies are marked as class 1 and the background pixels as class 0. Since each frame is processed in real-time and in a line-by-line fashion, each row of the binary map, $\mathbf{X}$, is also provided in real-time without requiring to receive the whole image to apply the thresholding.

Additionally, Algorithm 4 shows the modifications made to the proposed set of core operations displayed in Algorithm 2. For the sake of clarity, they have been highlighted in blue. As it can be seen, α is also set as an input parameter and it is used in Line 9 to

Algorithm 3 The LbL-FAD algorithm.

Inputs:
$\mathbf{HI} = [\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_k], n_f, \alpha$
Outputs:
$\mathbf{X} = [\mathbf{x}_{11}, \mathbf{x}_{12}, ..., \mathbf{x}_{kj}]$
Algorithm:
Stage 1:
for k = 1 **to** n_f **do**
    $\mathbf{E}_\mathrm{k}$ = **Set of core operations**$(\mathbf{M}_\mathrm{k}, \alpha)$;
    $\mathbf{B^*} = [\mathbf{B^*}, \mathbf{E}_\mathrm{k}]$;
end for
Stage 2:
$[\hat{\boldsymbol{\mu}}_b, \mathbf{B}_\mathrm{g}, \mathbf{Q}, \mathbf{U}, \tau]$ = **Set of core operations**$(\mathbf{B^*}, \alpha)$;
Stage 3:
$\mathbf{P} = \mathbf{I} - \mathbf{Q}\mathbf{U}'$;
Stage 4:
for k = $n_f + 1$ **to** $\frac{nr \cdot nc}{BS}$ **do**
    for j = 1 **to** BS **do**
        $\mathbf{d}_\mathrm{j} = (\mathbf{r}_\mathrm{kj} - \hat{\boldsymbol{\mu}}_b)' \cdot \mathbf{P} \cdot (\mathbf{r}_\mathrm{kj} - \hat{\boldsymbol{\mu}}_b)$;
        if $\mathbf{d}_\mathrm{j} \leq 1.5 \cdot \tau$ **then**
            $\mathbf{x}_\mathrm{kj} = 0$;
        else
            $\mathbf{x}_\mathrm{kj} = 1$;
        end if
    end for
end for

establish a stopping condition during the p vector selection process. Moreover, τ scalar is also added to the list of algorithm outputs.

3.4 Hardware-Friendly LbL-FAD (HW-LbL-FAD)

The first and second stages of the LbL-FAD method, in which the background subspace is modelled, are the less computationally demanding parts of the algorithm. On the one hand, they are implemented in a small group of hyperspectral frames. On the other

Algorithm 4 LbL-FAD. Set of core operations

Inputs:

$\mathbf{M}_k = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{BS}], \alpha$

Outputs:

$\hat{\boldsymbol{\mu}}$ {Average Pixel}; $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p]$ {Characteristic pixels}; $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$ {Orthogonalized vectors};$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p]$ {Orthonormalized vectors}; $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p]$ {Projection vectors}; τ {Threshold}

Algorithm:

Average pixel: $\hat{\boldsymbol{\mu}}$;
Centralization: $\mathbf{C} = \mathbf{M}_k - \hat{\boldsymbol{\mu}}$;
exit $= 0$
while exit $= 0$ **do**
    for $j = 1$ **to** BS **do**
        Brightness Calculation: $\mathbf{b}_j = \mathbf{c}'_j \cdot \mathbf{c}_j$;
    end for
    Maximum Brightness: $j_{max} = \text{argmax}(\mathbf{b}_j)$;
    if $\frac{b_{j\max}}{(\mathbf{r}_{j\max} - \hat{\boldsymbol{\mu}})' \cdot (\mathbf{r}_{j\max} - \hat{\boldsymbol{\mu}})} \cdot 100 < \alpha$ **then**
        Stop condition: exit $= 1$
    else
        Extracted pixels: $\mathbf{e}_n = \mathbf{r}_{j_{max}}$;
        $\mathbf{q}_n = \mathbf{c}_{j_{max}}$;
        $\mathbf{u}_n = \mathbf{q}_n / b_{j_{max}}$;
        Projection: $\mathbf{v}_n = \mathbf{u}'_n \cdot \mathbf{C}$;
        Subtraction: $\mathbf{C} = \mathbf{C} - \mathbf{q}_n \cdot \mathbf{v}_n$;
        $\tau = b_{j\max}$
end if
end while

hand, they could be potentially performed off-line using hyperspectral data from previous flights over the same terrain area. Nonetheless, the last two LbL-FAD stages, in which the projection matrix, $\mathbf{P}$, is computed and the potential anomalous pixels are detected, bear most of the complexity and computational burden of the entire anomaly detection process and indeed, must be carried out aboard in order to provide a real-time monitoring. Consequently, a great of effort has been invested to optimize them for being later accelerated in a more efficient way.

In this sense, the Hardware-Friendly LbL-FAD (HW-LbL-FAD) [150] emerges as an alternative algorithmic solution that employs mathematically equivalent operations but at a lower computational burden. The major difference is that the orthogonal projection matrix, $\mathbf{P}$, is not explicitly calculated. In this regard, the projection separation index, d, as it is defined in Equation 3.2, in overall calculates the brightness, or the squared l^2-*norm*, of the orthogonal component of each image pixel, $\mathbf{r}_j$, to the subspace spanned by the background distribution. Accordingly, the modified Gram-Schmidt orthogonalization process, described in detail in Section 2.3.1, may be used for calculating the orthogonal projection, d, of each hyperspectral pixels, $\mathbf{r}_j$. It also facilitates the algorithm execution using customized integer arithmetic at different levels of precision that can be adapted for achieving the best relation between detection accuracy and computational burden.

The sequence of operations involved in this alternative solution of the LbL-FAD algorithm is collected in the pseudocode displayed in Algorithm 5. As it can be seen, the background average pixel, $\hat{\boldsymbol{\mu}}_b$, which is obtained in *Stage 2* of the algorithm, is firstly subtracted from each image pixel, $\mathbf{r}_j$ in Line 2. Afterwards, the spectral information of $\mathbf{r}_j$ that can be spanned by the background samples is removed using $\mathbf{Q}$ and $\mathbf{U}$ vectors also outputted in *Stage 2*, as it is shown in Lines 3 to 6. Finally, the remaining spectral information of each image pixel, $\mathbf{r}_j$, which is in fact orthogonal to the space spanned by the background samples, is measured in line 7. If this value is higher than the specified threshold ($1.5 \cdot \tau$), the pixel is labelled as an anomaly (Lines 8 to 12 of Algorithm 5).

The HW-LbL-FAD algorithm provides several advantages in relation to the process followed by the original LbL-FAD:

- *Less memory requirements*:

  With this new proposal, it is not needed to keep matrix $\mathbf{P}$ in memory, whose size is $nb \cdot nb$. Instead, the set of p spectral vectors within $\mathbf{Q}$ and $\mathbf{U}$ vectors are just needed. Normally, p is much lower than nb.

- *Less number of computing operations and complexity.*

  On one side, once that the $\mathbf{Q}$ and $\mathbf{U}$ vectors have been extracted in the second stage of the algorithm (see Section 3.3.2), it is not needed to compute the calculation of the orthogonal matrix, $\mathbf{P}$. Furthermore, the process followed by the HW-LbL-FAD algorithm for scanning the hyperspectral frames looking for possible anomalies, based on the Q and U vectors, requires much less number of operations (OPs) than

Algorithm 5 Stage 3 and Stage 4 performed by the HW-LbL-FAD for each image block, $\mathbf{M}_k$

Inputs:

$\mathbf{M}_k = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{BS}], \mathbf{Q}, \mathbf{U}, \hat{\mu}_b, \tau$

Outputs:

$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{BS}]$

Algorithm:

for j $= 1$ **to** BS **do**
  Centralization: $\mathbf{r}_j = \mathbf{r}_i - \hat{\mu}_b$;
  for k $= 1$ **to** p **do**
    Projection. $\mathbf{v}_n = \mathbf{U}'_k \cdot \mathbf{r}_j$;
    Subtraction: $\mathbf{r}_j = \mathbf{r}_j - \mathbf{Q}_k \cdot v$;
  end for
  Brightness Calculation: $\mathbf{d}_j = \mathbf{r}'_j \cdot \mathbf{r}_j$;
  if $\mathbf{d}_j \leq 1.5 \cdot \tau$ **then**
    $\mathbf{x}_j = 0$;
  else
    $\mathbf{x}_j = 1$;
  end if
end for

the process followed by the LbL-FAD using matrix $\mathbf{P}$. Indeed, the number of OPs involved by both the LbL-FAD and the HW-LbL-FAD methods for computing the projection separation index, $\mathbf{d}$, for the BS pixels within a HSI block, $\mathbf{M}_k$, is shown in Table 3.1.

- *Data types optimization.* Unlike the $\mathbf{P}$ computation carried out by the original LbL-FAD algorithm, the range of values that can be obtained by the operations involved by the modified Gram-Schmidt orthogonalization process described in Algorithm 5 can be known in advance. This makes possible the use of fixed-point arithmetic instead of floating-point notation, making the HW-LbL-FAD algorithm more suitable for being executed in some hardware devices such as space-graded FPGAs [151].

Algorithm	FLOPs	Complexity
LbL-FAD	$BS \cdot (2 \cdot N_b^2 + 2 \cdot N_b)$	$O(BS \cdot N_b^2)$
HW-LbL-FAD	$BS \cdot [p \cdot 4 \cdot N_b + 2 \cdot N_b]$	$O(BS \cdot p \cdot N_b)$

TABLE 3.1: Number of OPs performed by the LbL-FAD algorithm and the HW-LbL-FAD method for computing the projection separation index, $\mathbf{d}$, for the BS pixels within a HSI block, $\mathbf{M}_k$.

3.5 Experimental Results

This section discloses the most significant results obtained by the LbL-FAD algorithm when it is utilized to perform the anomaly detection process. In order to evaluate the performance of the proposed algorithm in real scenarios, eight different hyperspectral data sets have been used as inputs to the algorithm. Some of them were collected by traditional remote sensing observation platforms, such as the Wildfire Airborne Sensor Program (WASP) Imaging System and the NASA JetPropulsion Laboratorys Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), while others were sensed by a new unmanned aerial vehicle (UAV) - based acquisition system. This test bench has been analysed using both floating-point and fixed-point notations with different levels of precision. Additionally, the performance of the LbL-FAD algorithm has been compared with some of the most relevant algorithms of the state-of-the-art, namely the Orthogonal Subspace after RX (OSPRX), the Low-rank and Sparse matrix decomposition-based Mahalanobis distance method (LSMAD) and the Progressive Line Processing of a Kernel version of the RX algorithm (PLP-KRXD).

3.5.1 Reference Hyperspectral Data

In this section, the hyperspectral data used for evaluating the performance of the LbL-FAD algorithm are introduced. This test bench is composed of two conventional data widely used in the literature to evaluate the performance of hyperspectral anomaly detectors, a computer-generated synthetic image and a new bunch of real hyperspectral data collected by one UAV available in our institutional facilities.

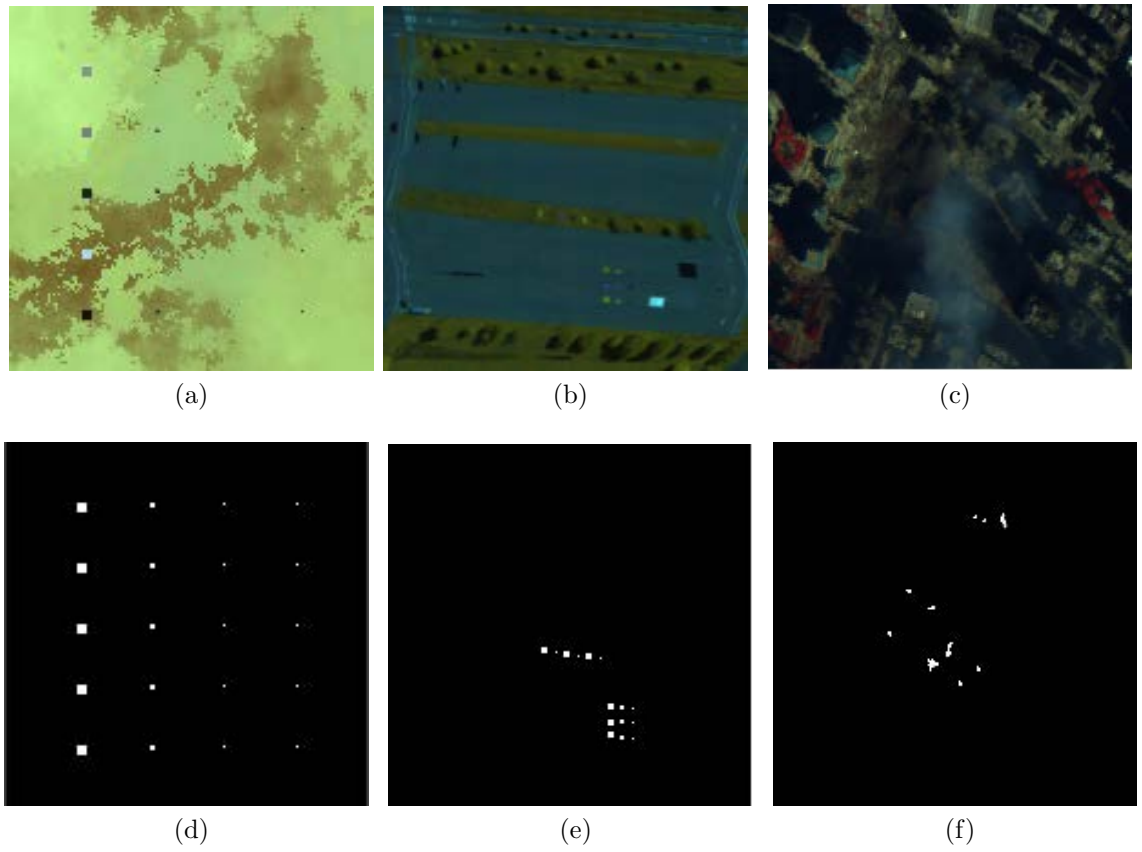
The synthetic data has a size of 150x150 hyperspectral pixels and 429 spectral bands. It was generated using a spectral library collected from the United States Geological Survey

(USGS) [152]. The background was simulated using four different spectral signatures whose abundances were generated using a Gaussian spherical distribution [153]. Twenty panels of various sizes arranged in a 5x4 matrix were introduced as anomalies. There are five 4x4 pure-pixel panels lined up in five rows in the first column, five 2x2 mixed-pixel panels in the second column, five subpixel panels combined with the background in a proportion of 50% in the third column and five subpixel panels blended with the background at 75%. Therefore, the simulated image has 110 anomaly pixels, a 0.49% of the image. A false color representation of this synthetic image is shown in Figure 3.2a while its ground-truth is shown in Figure 3.2d.

Between the couple of commonly used hyperspectral imagery, the first data set was taken over the Rochester Institute of Technology (RIT) by the WASP Imaging System [154]. A portion of the overall image taken over a parking lot with a size of 180x180 pixels and 120 bands has been used in this study, as it can be seen in Figure 3.2b where anomalies are fabric targets that are composed of 72 pixels and account for 0.22% of the image. Its corresponding ground-truth is shown in Figure 3.2e.

The second real data set was collected by the NASA Jet Propulsion AVIRIS over the World Trade Centre (WTC) area in New York City on September 16, 2001 [155]. The original data set has a size of 614x512 pixels and 224 spectral bands from 0.4 to 2.5 μm although a smaller region with size of 200x200 pixels was selected as data set. Anomalies are thermal hot spots that consist of 83 pixels and account for 0.21% of the image scene. Figure 3.2c shows a representation of this image while its corresponding ground-truth is shown in Figure 3.2f.

The six remaining HSIs used in the experiments were sensed by the aerial platform extensively analyzed in [89]. This acquisition system carries a *Specim FX10* pushbroom hyperspectral camera mounted on a DJI Matrice 600 drone. The hyperspectral sensor covers the visible near infrared (VNIR) of the electromagnetic spectrum, from 400 to 1000 nm, using 224 spectral bands with a spectral full width at half maximum (FWHM) of 5.5 nm and, 1024 spatial pixels per scanned cross-track line. In this work, the first 20 and the last 45 spectral bands are discarded due to the low spectral response of the hyperspectral sensor at those wavelengths, what results in just 160 spectral bands being retained. The data used for the experiments were collected by the aforementioned acquisition platform over different farming areas on the island of Gran Canaria (Spain), displayed in the Google Map picture shown in Figure 3.3a, during three different flight campaigns.



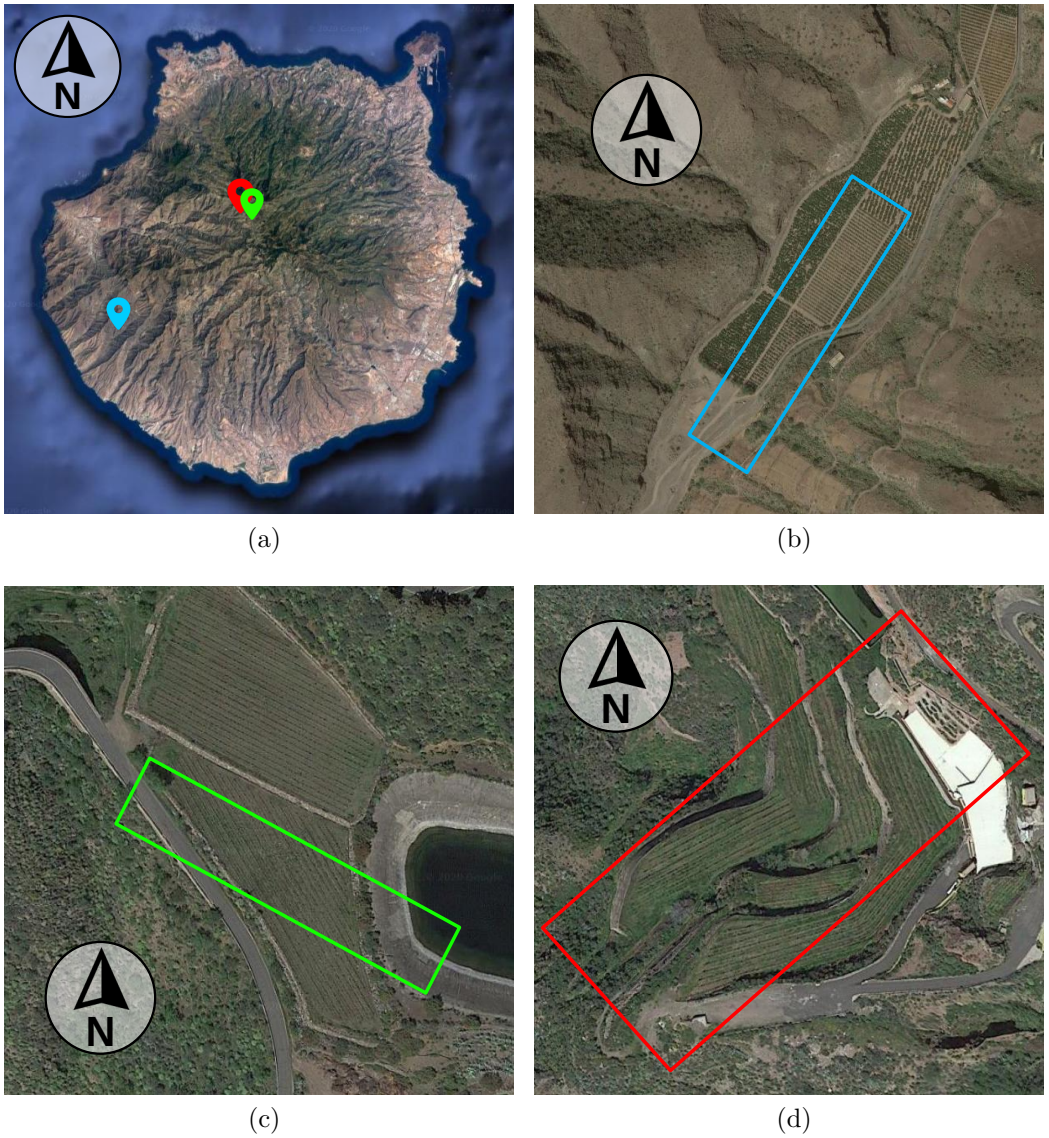
(a) (b) (c)

(d) (e) (f)

FIGURE 3.2: RGB representation of the HSIs used in the experiments. (a) Synthetic Image. (b) WASP RIT scene. (c) AVIRIS WTC scene. (d) ground-truth Synthetic Image. (e) ground-truth WASP RIT scene. (f) ground-truth AVIRIS WTC scene.

The first flight campaign, highlighted in blue color in Figure 3.3b, was carried out over a plantation of bananas in the south-west of the island, concretely in a village called Veneguera (27°52'17.4"N 15°45'44.2"W). The flight was performed at a height of 72 m over the ground at a speed of 6 m/s with the hyperspectral camera capturing frames at 125 frames per second (FPS), resulting in a ground sampling distance in line and across line of approximately 5 cm. This mission consisted of 6 waypoints that provided 3 swathes. The area covered by these swathes is highlighted in the Google Maps picture displayed in Figure 3.3b. Concretely, three portions of 825 hyperspectral frames, with their 1024 hyperspectral pixels, were cut out from these swathes and used for the experiments. A RGB representation of these HSI portions are displayed in Figures 3.5a-c, and their locations within the corresponding swath are displayed in Figures 3.4a-c, respectively.

The second and third flight campaigns were carried over some vineyard areas in a village called Tejeda located in the center of the island. Their exact coordinates are 27°59'35.6"N

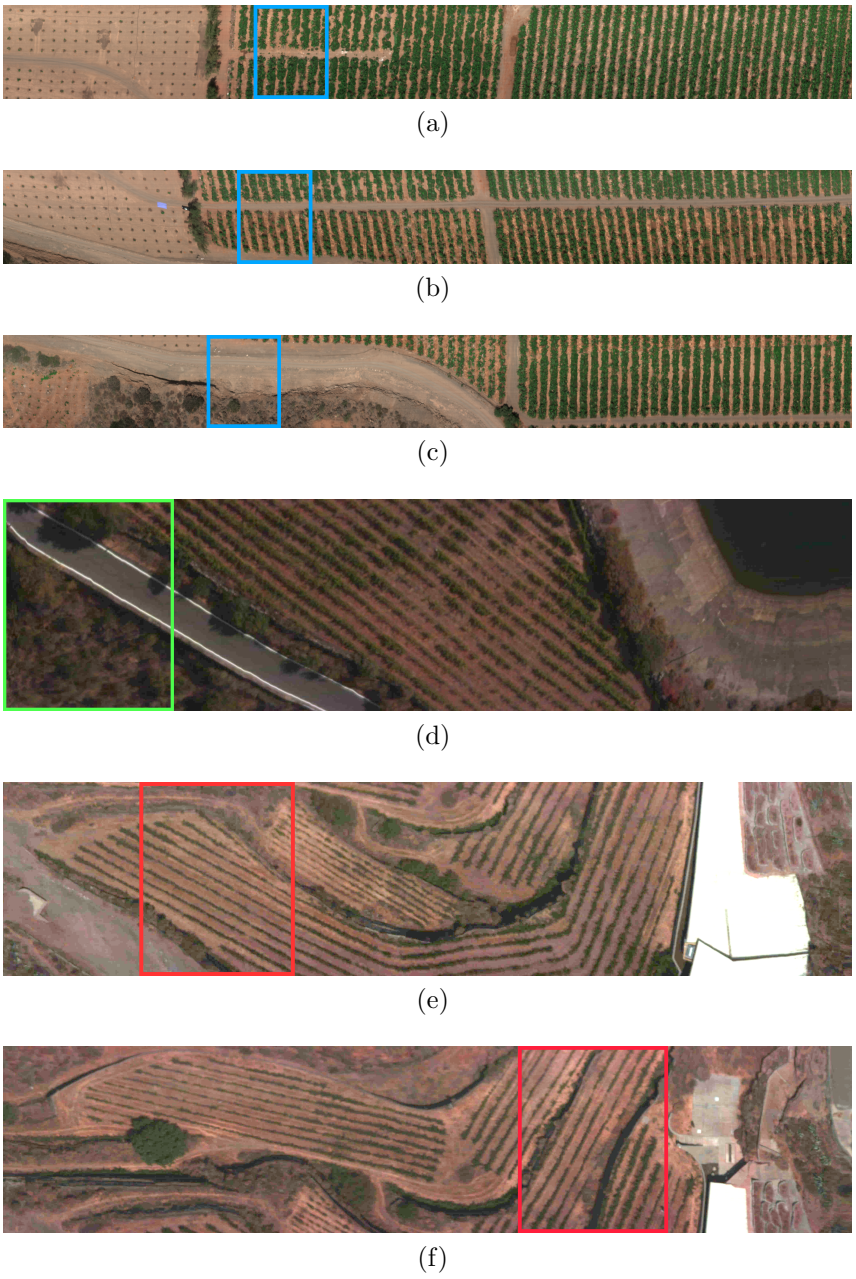


(a) (b)

(c) (d)

FIGURE 3.3: Google Maps pictures of the farming areas corresponding to the HSIs used in this work. (a) Location of the terrains on the island of Gran Canaria (Spain). (b) Area covered by the first flight campaign over a banana plantation. (c, d) Area covered by the second and third flight campaigns over different vineyards.

15°36'25.6"W and 27°59'15.2"N 15°35'51.9"W respectively and, they have been highlighted in color green and red in Figures 3.3c and 3.3d. In particular, the second flight campaign was performed at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS. This results in a ground sampling distance in line and across line of approximately 3 cm. This flight mission consisted of 12 waypoints that provided 6 swathes, but just one of them was used in the experiments carried out in this work. The ground area covered by this swath is highlighted in the Google Maps picture displayed in Figure 3.3b. One smaller portion of 825 hyperspectral



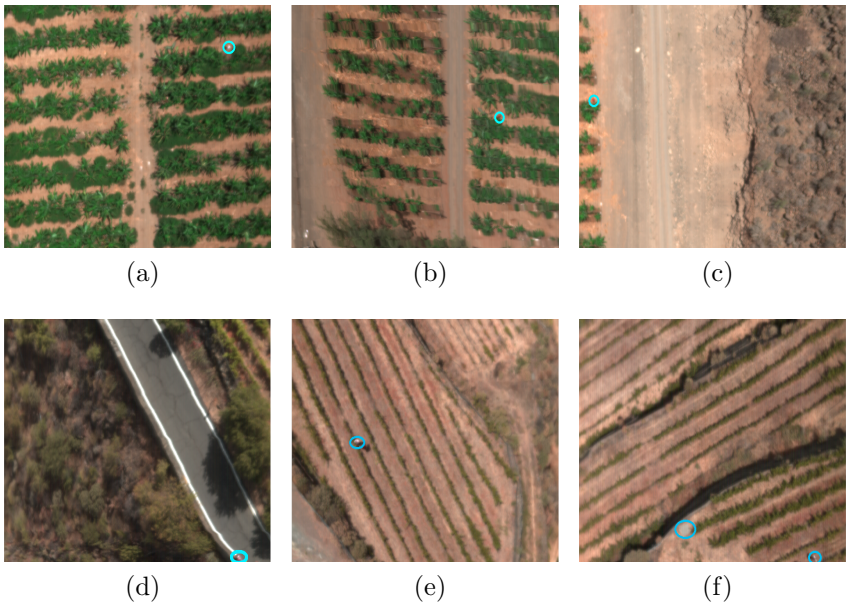
(a) (b) (c) (d) (e) (f)

FIGURE 3.4: RGB representation of the hyperspectral data acquired in each mission campaign swath that was used in this work. Color squares highlight the regions selected for the experiments.

frames with all their 1024 hyperspectral pixels was cut out from the entire swath image for the simulations. Figure 3.5d displays its RGB representation while its location within the entire frame is shown in Figure 3.4d.

In the third flight campaign, the terrain was scanned at a flight height of 45 m over the ground and at a speed of 6 m/s with the hyperspectral camera capturing at 200 FPS. The resulting ground sampling distance in line and across line was approximately 3 cm.

The entire flight mission consisted of 5 swathes, but just 2 of them were used for the experiments in this work. The ground area covered by these swathes is highlighted in the Google Maps picture displayed in Figure 3.3d. From them, two smaller portions of 825 hyperspectral frames with all their 1024 hyperspectral pixels were cut out for the simulations. Figures 3.5e-f display their RGB representations, while Figures 3.4e-f show their locations within the entire swathes.



(a) (b) (c)

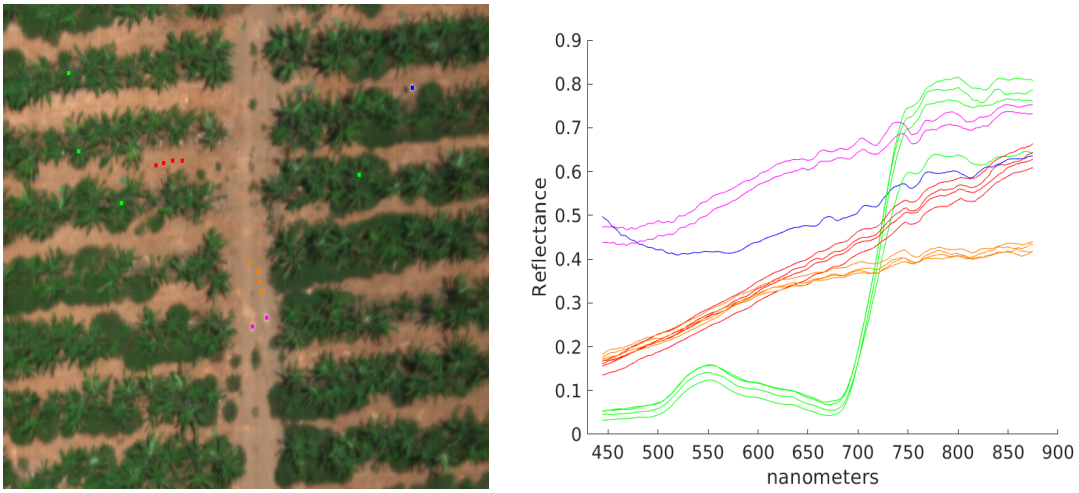
(d) (e) (f)

FIGURE 3.5: RGB representation of the employed test bench. Pixels enclosed in blue circles represent the anomalous entities to be detected. (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Drone Image 4. (e) Drone Image 5. (f) Drone Image 6.

The aforementioned images were calibrated using a white and dark calibration to obtain reflectance values. Some examples of calibrated signatures corresponding to different pixels are displayed in Figure 3.6. Nevertheless, either orthorectification or georeferencing processes have not been carried out for the acquired raw data. In this sense, the HSIs have been built up just by placing the subsequent captured hyperspectral frames one next to the other [156]. This does not degrade the quality of the experiments carried out in this work since the LbL-FAD algorithm does not use any kind of spatial information.

To evaluate the detection performance of the proposed method, the selected data set contains some artifacts that are considered to be anomalous pixels. Their locations within the test bench have been highlighted in Figure 3.5 using blue circles. These artifacts are people walking among the crop fields in Figures 3.5a-c, e. In Figure 3.5d, the anomalous

entities are two people standing next to the road. Finally, Figure 3.5f contains a person walking through the vineyard area and a concrete construction.



(a) (b)

FIGURE 3.6: Example of spectral signatures corresponding to some real pixels within Drone Image 1 (Figure 3.5 a). (a) Pixel locations. (b) Pixel spectral signatures.

3.5.2 Reference Algorithms

The performance of the LbL-FAD algorithm has been compared with three state-of-the-art anomaly detectors. Two of them are global detectors, which work with the entire image, while the third one is a line-by-line approach meant to work with hyperspectral imaging pushbroom sensors as our proposal does.

The first global detector is the OSPRX algorithm [58, 134]. It uses the first PCA/SVD components to define the background subspace and then projects the data onto the orthogonal subspace before applying the RX detector. In this case, the OSPRX algorithm employs the SVD method to estimate the first eigenvectors of the global covariance matrix. Then, it projects the data onto the orthogonal subspace to the one spanned by the resulted background eigenvectors in the same way as it is defined by Equations 3.2 and 3.3, being $\mathbf{W}$ an orthonormal matrix. This method just requires the number of SVD components to define the background subspace (d_{OSP}) as input parameter. As it can be seen, the employed methodology is similar to the one followed by our proposal. For this reason, the OSPRX algorithm has been selected among the other methods found in the literature to evaluate and compare the detection performance of the proposed LbL-FAD.

The second global detector is the LSMAD [63]. It is based on low-rank and sparse matrix decomposition techniques to decompose the HSI as the sum of three elements: a low-rank matrix, which captures the global background information, a sparse matrix representative of the sparse property of the anomalies and a noise matrix. Then, the LSMAD algorithm constructs a Mahalanobis-distance based anomaly detector utilizing the low-rank matrix. The problem of the inverse covariance matrix is overcome using the first largest eigenvalues and eigenvectors of the covariance matrix as it is further explained in [63]. This algorithm requires as input parameters the maximal rank of the background matrix, r, the cardinality of the sparse matrix, k, which is related to the ratio of anomalies in the image scene, and finally, a random matrix $\mathbf{A}_1 \in N_b \times r$. Depending on matrix $\mathbf{A}_1$, the results are different every time and due to this, the detection results shown in Section 3.5.5 reflect the average of 10 simulations for each test image.

Finally, the last state-of-the-art algorithm is an approach to the progressive line processing of a kernel version of the RX (PLP-KRXD) [149]. It employs a parallel causal sliding window to ensure the causality of the detection system [149]. The PLP-KRXD algorithm is updated recursively, which avoids having to recalculate the previously processed data frames. This algorithm requires as input parameters the polynomial kernel parameter, d, and the causal window size, $[a, b]$.

3.5.3 Assessment Metrics

The LbL-FAD algorithm results in a binary map where anomalous pixels are segmented from the background. For this reason, the True Positive Rate (TPR) and the False Positive Rate (FPR) will be used as assessment metrics to evaluate the detection performance of the proposed methodology. In this context, a true positive (TP) happens when an anomalous pixel is properly classified. Hence, the TPR is defined by the proportion of anomalous pixels correctly sorted among all anomalous pixels in the image, namely positive entities (P). On the contrary, a false positive (FP) occurs when a background pixel is declared as an anomaly. Therefore, the FPR is computed as the proportion of misclassified background pixels among all background image pixels, namely negative entities (N). These two metrics are displayed in Equations 3.6 and 3.7, respectively. It has been considered that anomalies are marked as 1 and the background pixels as 0 in the resulting binary map, $\mathbf{X}$, and in the ground-truth, $\mathbf{GT}$.

Nonetheless, the state-of-the-art anomaly detection solutions selected for benchmarking result in gray-scale maps where the detection of anomalous entities is made regarding the pixel intensity. Therefore, a thresholding is required for discriminating the anomalous pixels. In order to set this threshold and, at the same time, make a fair comparison in terms of detection performance between these reference anomaly detector and our proposal, it is measured for each data set the *FPR* reached by each reference detectors when the resulting detection maps are binarized using a threshold that generates the same *TPR* as the LbL-FAD algorithm. Similarly, it is also made for estimating the *TPR* but using a threshold that spawns the same *FPR* as the LbL-FAD algorithm. In the remainder of this Chapter 3, these assessment metrics are referred to as *C-FPR* and *C-TPR*, respectively.

$$TPR = \frac{\sum_{j=1}^{n_r \times n_c} \mathbf{X}_j = 1 \cap \mathbf{GT}_j = 1}{\sum_{j=1}^{n_r \times n_c} \mathbf{GT}_j = 1} = \frac{TPs}{P} \tag{3.6}$$

$$FPR = \frac{\sum_{j=1}^{n_r \times n_c} \mathbf{X}_j = 1 \cap \mathbf{GT}_j = 0}{\sum_{j=1}^{n_r \times n_c} \mathbf{GT}_j = 0} = \frac{FPs}{N} \tag{3.7}$$

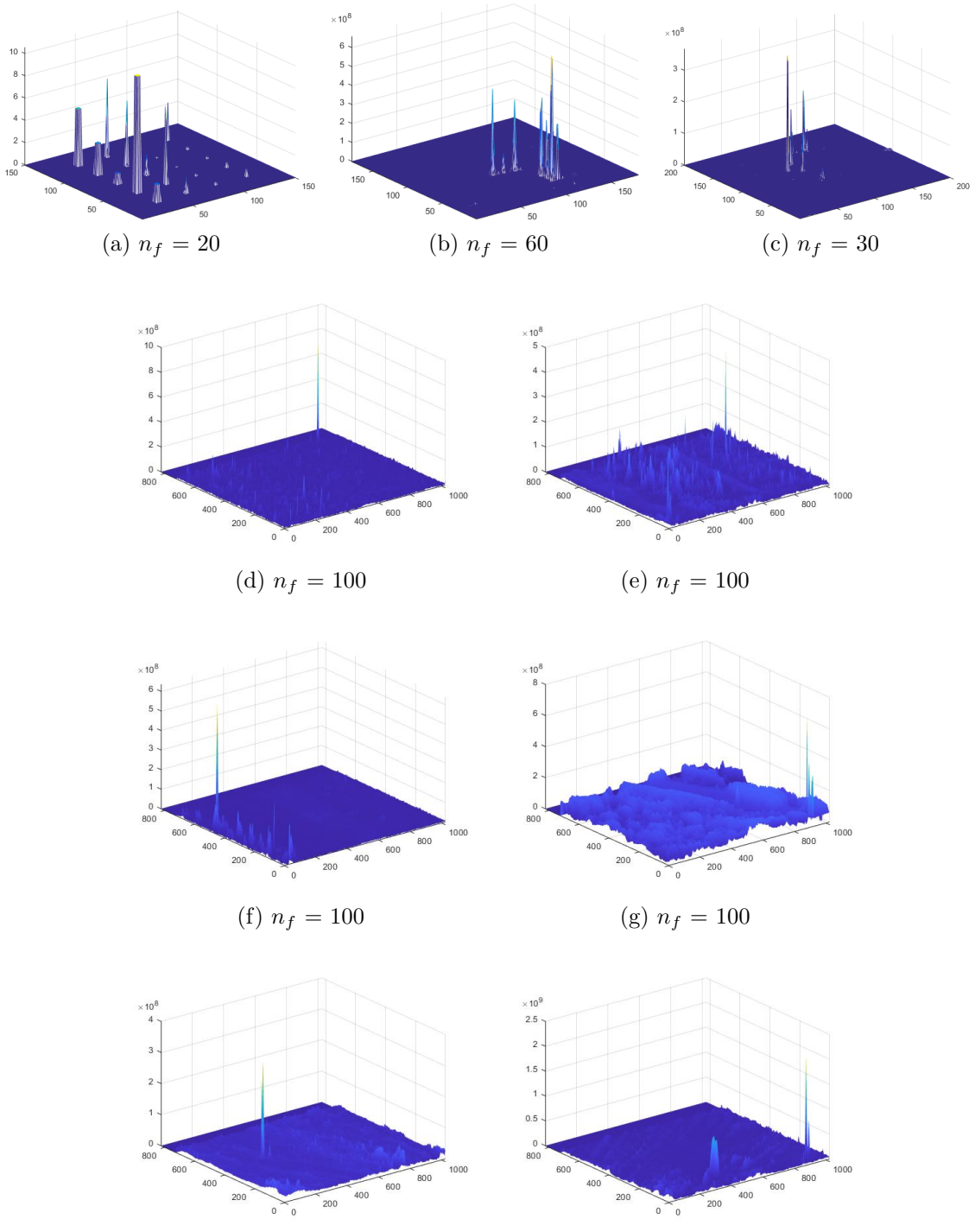
In any case, the aforementioned performance evaluation metrics required the availability of reliable ground-truths, in which the spatial resolution of anomalous objects is high enough to obtain accurate pixel-level ground-truths. For the hyperspectral test bench used for the experiments, we have only tested ground-truths for the *Synthetic Image*, the *WASP RIT* scene and the *AVIRIS WTC* scene, which are displayed in Figure 3.2. Hence, the aforementioned metrics will be only used to evaluate the algorithm performance for these particular images. In the case of the HSIs sensed by the UAV-based acquisition system (see Figure 3.5), the evaluation of the detection performance is visually made at object-level through the description of the resulting binary maps where anomalies and background elements are segmented. This is because the acquired images have been sensed at high altitudes and the exact position of the anomalies in the field was not measured. As a consequence, on the one hand, anomalous entities cover a very small number of image pixels and, on the other hand, pixels at object borders are mixed with the background. For this reason, it is very difficult to establish precise boundaries and thus, to generate accurate pixel-level ground-truths.

3.5.4 Detection performance of the LbL-FAD algorithm

In this section, the detection performance of the LbL-FAD algorithm is evaluated in terms of the quality of the detection results and the separability between anomalous and background classes. To ensure accurate detection results, it is recommendable that both anomaly and background classes are separated as much as possible in order to prevent misclassification during the thresholding process, which results in binary maps where anomalous entities are segmented. To this end, the anomalous pixels must be labelled with notable high scores as opposite to the background pixels. With it, the robustness of an algorithm to changes in the decision threshold is ensured and consequently, the probability of misclassification significantly decreases.

In order to assess the separability performance of the LbL-FAD algorithm, the 3D representations of the 2D binary maps given as output by the LbL-FAD detector are displayed in Figure 3.7 for the nine hyperspectral data sets. The third dimension represents the scores given by the LbL-FAD algorithm to each image pixel. For drawing these 3D representations, the internal thresholding done by the LbL-FAD method using the estimated threshold, τ, is discarded. Simulations have been carried out for blocks of nc pixels, that is, BS corresponds to the number of image columns, nc. With respect to the number of hyperspectral frames employed to estimate the background pattern, n_f, it has been specified at the bottom of each Figure. Analysing these 3D plots, we can conclude that the LbL-FAD detector assures a high separability between both anomaly and background classes. Anomalous pixels are characterized by steep peaks in contract to background pixels that are rated with smaller scores. The worst results in terms of the separability feature is for the HSIs sensed by the UAV-based acquisition system and more specifically, for *Drone Image 4* and *Drone Image 6* (see Figures 3.7g and 3.7i). The main reason is that these images have not been preprocessed in order to reduce the distortions stemmed by the drone movement or to eliminate the presence of potential unwanted glosses.

Apart from the 3D plots displayed in Figure 3.7, Figures 3.8 and 3.9 show the 2D binary maps obtained by the proposed LbL-FAD algorithm as output. For the sake of clarity, they are superimposed on a panchromatic representation of the scenes to be analysed in order to make easier the result interpretation. In these displays, lines in blue color indicate the n_f frames employed to estimate the background pattern. Since ground-truths are available for images shown in Figure 3.2 and the assessment metrics described in Section 3.5.3 will be used to evaluate the performance of the LbL-FAD algorithm on these images, we have considered convenient to illustrate the anomalous pixels detected by our proposal



(a) $n_f = 20$ (b) $n_f = 60$ (c) $n_f = 30$

(d) $n_f = 100$ (e) $n_f = 100$

(f) $n_f = 100$ (g) $n_f = 100$

(h) $n_f = 100$ (i) $n_f = 100$

FIGURE 3.7: 3D plots of the detection results obtained by the LbL-FAD algorithm for the nine data sets. (a) Synthetic Image. (b) WASP RIT scene. (c) AVIRIS WTC scene. (d) Drone Image 1. (e) Drone Image 2. (f) Drone Image 3. (g) Drone Image 4. (h) Drone Image 5. (i) Drone Image 6.

at pixel-level. Therefore, those pixels identified as anomalous have been highlighted in red color in Figure 3.8. Unlike it, anomalous entities present in HSIs collected in Figure 3.5 take up few image pixels and hence, spatial lines corrupted by anomalous signatures have been highlighted in red color instead. In addition, those anomalous pixels detected by the LbL-FAD algorithm have been also locked up in red circles.

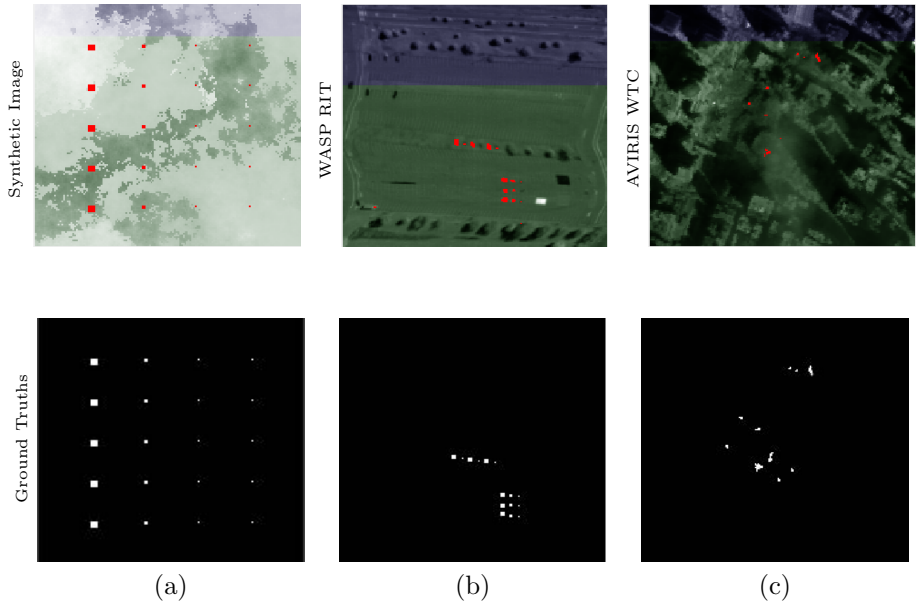


FIGURE 3.8: Anomaly detection results obtained by the LbL-FAD algorithm for the Synthetic Image, the WASP RIT scene and the AVIRIS WTC scene. Lines in blue color represent the n_f frames employed to estimate the background distribution. Pixels marked in red corresponds to the anomalous pixels detected by the LbL-FAD algorithm. (a) Synthetic Image. (b) WASP RIT scene. (c) AVIRIS WTC scene.

Table 3.2 collects the *TPR* and the *FPR* reached by the LbL-FAD detector for the anomaly maps displayed in Figure 3.8. On account of these results, we can conclude that the LbL-FAD method features a high detection capability. Regarding the *Synthetic Image*, the LbL-FAD method detects all anomalous pixels without misclassifying any pixel. For the *WASP RIT* scene, more than 93% of the anomalous pixels are detected while the *FPR* just ascends to 0.0958 % of the overall number of background pixels. As it can be seen from Figure 3.8b, just few mixed pixels located above all in the edges of the anomalous objects are not detected. In the case of the *AVIRIS WTC* scene, it is a very challenging image since rare targets are thermal hot spots that are also sub-pixel in size. As a matter of fact, the spatial resolution of a single pixel is 1.7 square meters. Although the *TPR* descends to 43.37% for this test case, at least some pixels of all targets have been detected with a very low *FPR* (just 0.01753 % of all background pixels).

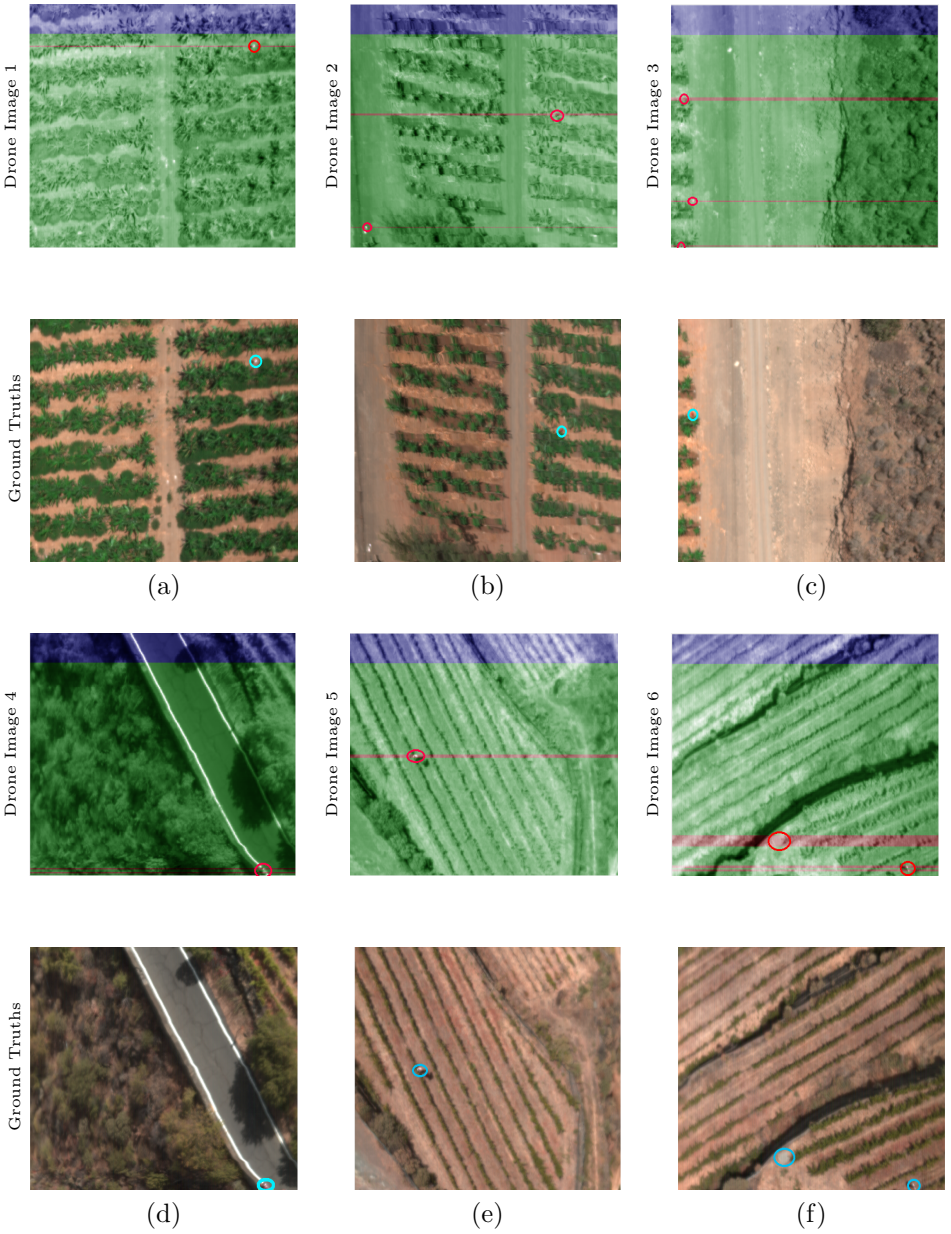


FIGURE 3.9: Anomaly detection results obtained by the LbL-FAD algorithm for the HSIs sensed by the UAV-sensed acquisition system. Lines in blue color represent the n_f frames employed to estimate the background distribution. Green lines represent the hyperspectral frames free of anomalies. Red lines highlight those frames identified by the LbL-FAD algorithm to be corrupted by anomalous pixels. Pixels enclosed in red circles represent the exact locations of detected anomalous pixels. (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Drone Image 4. (e) Drone Image 5. (f) Drone Image 6.

Finally, regarding the HSIs sensed by the drone-based system, the detection results obtained by the LbL-FAD algorithm are quite satisfactory as well. Compared to the RGB

representations displayed in Figure 3.5, anomalous objects detected in Figures 3.9a, 3.9g-3.9i perfectly match those anomalous entities highlighted in *Drone Image 1* and *Drone Images 4-6*. Regarding *Drone Image 2* and *Drone Image 3*, the anomalous bodies to be detected have been successfully identified but also other sparse brightness present in the scenes, as it can be seen in Figures 3.9b and 3.9c. In short, the anomaly detection results are in general quite accurate for these data sets although the spatial resolution covered by the anomalous entities is quite low and hence, they are composed of mixed spectral signatures.

Data Set	LbL-FAD			OSPRX			LSMAD				PLP-KRXD				
	TPR	FPR	n_f	C-FPR	C-TPR	d_{OSP}	C-FPR	C-TPR	r	k	C-FPR	C-TPR	a	b	d
Synthetic Image	**1.00**	**0,00**	20	**0,00**	**1,00**	5	**0,00**	**1,00**	3	0,005	1,00	0,00	15	5	1
WASP RIT scene	**0,93**	**9,59E-04**	60	3,40E-03	0,83	4	4,34E-02	0,64	3	0,003	6,22E-01	0,00	15	5	5
AVIRIS WTC scene	**0,43**	**1,75E-04**	30	1,80E-03	0,29	4	4,06E-02	0,088	3	0,002	5,07E-01	0.00	15	7	2

TABLE 3.2: Assessment metric summary about the quality of the anomaly detection results obtained by the LbL-FAD, the OSPRX, the LSMAD and the PLP-KRXD algorithms for the data set displayed in Figure 3.2.

3.5.5 Benchmarking against other state-of-the-art anomaly detectors

In this Section, the detection performance of the LbL-FAD method is compared with the three state-of-the-art algorithms briefly described in Section 3.5.2, which are: the OSPRX, the LSMAD and the PLP-KRXD methods. This analysis is made in a comparative framework using the *C-TPR* and the *C-FPR* assessment metrics for the *Synthetic Image*, the *WASP RIT* scene and the *AVIRIS WTC* scene. As a reminder, the *C-TPR* is the *TPR* reached by the reference detectors for the same *FPR* obtained by the LbL-FAD detector. On the contrary, the *C-FPR* is the *FPR* reached by the reference detectors for the same *TPR* achieved by the LbL-FAD method. Table 3.2 collects all these assessment metrics for the aforementioned detectors and data sets. To facilitate the understanding, since the *C-FPR* is related with the *TPR* obtained by the LbL-FAD algorithm, columns of Table 3.2 that collect this information are highlighted in dark gray. Similarly, columns that gather results of the *C-TPR* and the *FPR* reached by the LbL-FAD algorithm are highlighted in light dark. On this basis, the best algorithm in terms of detection performance is the one that gets the highest *TPR*, which means values of *TPR* close to 1, and the lowest *FPR*,

that is close to 0. Accordingly, the best results are also highlighted in bold in Table 3.2. In addition, Table 3.2 also summarizes the input parameters required by these methods to perform the anomaly detection process, whose descriptions were treated in Section 3.5.2.

Compared to the two global methods, the OSPRX and the LSMAD, the LbL-FAD algorithm gives very similar results, or even better in some cases, although it processes each hyperspectral frame entirely independently without requiring any spatial alignment. This fact is very noteworthy since the OSPRX and the LSMAD methods work with the whole HSI and thus, exploit both the spatial and the spectral features of it. With respect to the *Synthetic Image*, the three methods arise the same detection results, none of them misclassify any background or anomaly pixel. Nevertheless, the LbL-FAD algorithm notably outperforms the others global methods in terms of both TPR and FPR for the *WASP RIT* scene and the *AVIRIS WTC* scene. The greatest differences are found in the challenging *AVIRIS WTC* scene in terms of the TPR. In this particular case, the LbL-FAD algorithm detects the 43.37% of the thermal hot spots while this descends to 28.92% and 8.79% for the OSPRX and the LSMAD algorithms, respectively. Regarding other line-by-line methods, such as the PLP-KRXD method, the LbL-FAD algorithm clearly outperforms it. In this regard, the PLP-KRXD method is incapable of detecting any anomalous pixels for the same *FPR* obtained by the LbL-FAD algorithm for all data sets. In terms of the *FPR*, the PLP-KRXD algorithm obtains more than 1E+03 times the *FPR* reached by the LbL-FAD algorithm. In particular, the PLP-KRXD algorithm misclassifies the 100% of the background pixels for detecting all the anomalous entities for the *Synthetic Image*.

Besides the analysis made above, the 2D binary maps obtained by the aforementioned methods for the HSIs acquired by the UAV-based aerial platform have been qualitatively compared as well in the following lines. Similarly to the description made in Section 3.5.4, these detection maps are superimposed on a a panchromatic representation of the scenes to be analysed. In these displays, lines in red color indicate spatial hyperspectral frames corrupted by anomalous signatures. Within them, anomalous pixels are also highlighted enclosing them in red circles. Furthermore, since outputs of the three selected state-of-the-art detectors denote the likelihood of belonging to the anomaly class for each image pixel, the resulting anomaly maps have been transformed into binary maps after a detailed study of the data distribution. Figure 3.10 displays these color detection maps for the four compared algorithms. It has to be mentioned that the PLP-KRXD detection maps have been discarded except, for the *Drone Image 5*, since it has not been able to detect any of the anomalous entities regardless of the input parameter settings.

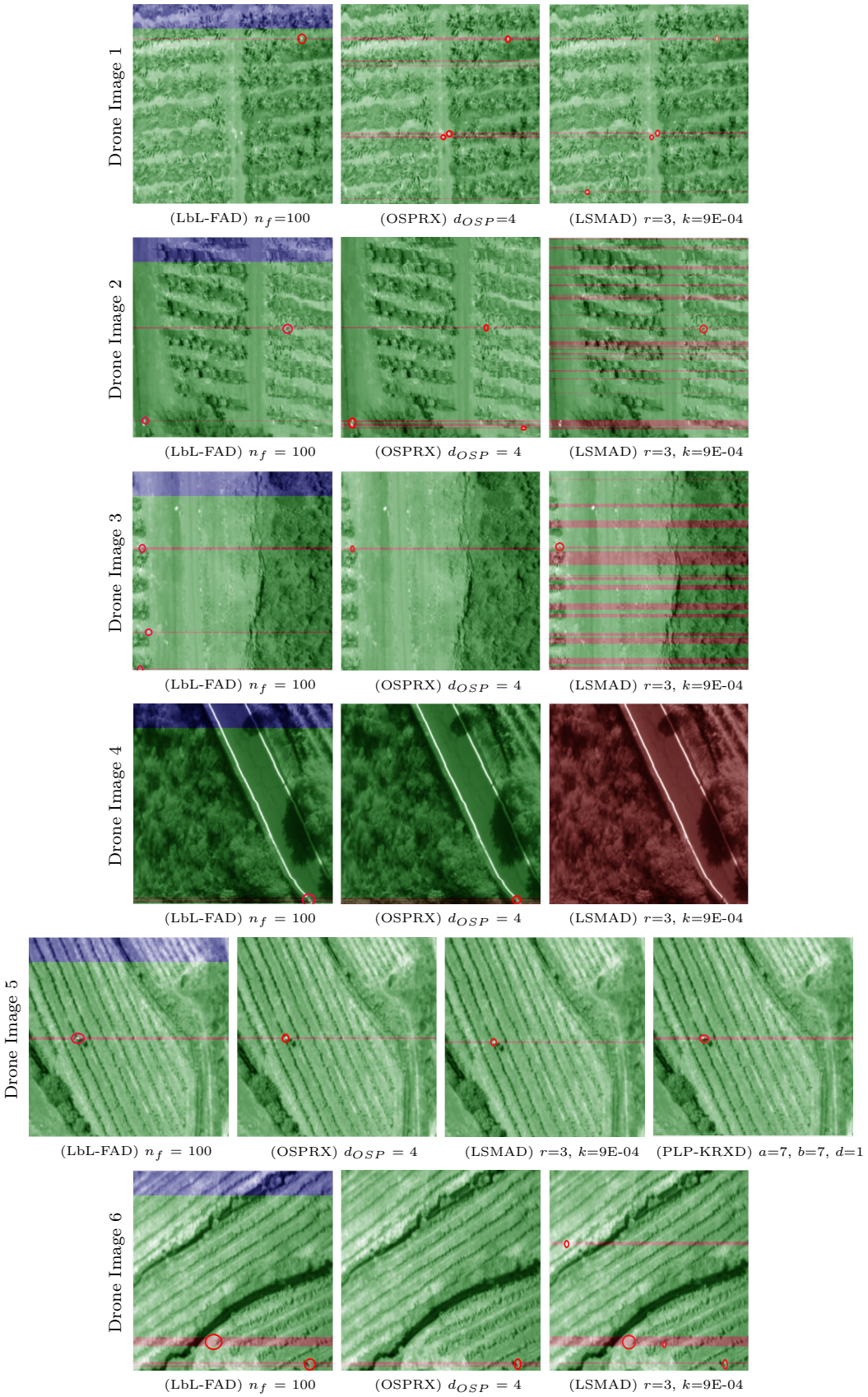


FIGURE 3.10: Anomaly detection results for the HSIs displayed in Figure 3.5 obtained by the LbL-FAD, the OSPRX, the LSMAD and the PLP-KRXD. Red lines highlight those frames identified by the different algorithms to be corrupted by anomalous pixels. Pixels enclosed in red circles represent the detected anomalous pixels.

As it can be seen, the detection performance of the LbL-FAD algorithm is in general very close to the OSPRX method. For *Drone Image 1*, the OSPRX method identifies the targeted anomalous entity, besides other sparse reflections present in the scene as the LSMAD method does. For the *Drone Image 2*, the anomalous targets have high response values for both the LbL-FAD and the OSPRX methods and some rare reflections in the right corner of the image are also detected by both methods. In the particular case of the LSMAD algorithm, much more sparse pixels are misclassified as abnormal targets. Something similar can be noticed with *Drone Image 3*. For the sake of clarity, it has been only pointed out the true anomalous object in these two particular cases for the LSMAD algorithm. Regarding *Drone Image 3*, the OSPRX algorithm is able to accurately discriminate some foreign objects in the right corner of the scene in contrast to the LbL-FAD algorithm. For the *Drone Image 4*, the LSMAD algorithm cannot suppress the background effectively and thus, it identifies as anomalous entities the white lines on both sides of the paved road. *Drone Image 5* is the only data set for which all mentioned methods have accurately discriminated the anomalous target. In the particular case of the PLP-KRXD method, it needs a local dual window to slide over the image, which badly influences the target detection and causes the loss of some anomalous pixels, as it happens for the other HSIs. Lastly, for *Drone Image 6*, all targets are detected by both the LbL-FAD and the LSMAD methods, although the former suppresses better the background. For this particular case, the OSPRX algorithm is not able to register the concrete construction.

After the above analysis, we can conclude that, in general terms, the LbL-FAD algorithm ensures good detection results working in a line-by-line fashion. In response, it is a suitable method to perform the anomaly detection process for applications that employ pushbroom scanners. In addition, it may be well asserted that the automatic thresholding method outlined in Section 3.3.4 is accurate enough for the targeted application, as well as the introduced methodology for extracting the background statistics.

Nonetheless, we would like to highlight the existing trade-off between the causality in line-by-line approaches and how to model the background distribution, which is the most important part in any adopted solution for anomaly detection. Actually, very few publications are made in this field where the anomaly detection problem is addressed in a line-by-line fashion [44, 129, 149, 157]. In the solution proposed in this Thesis work, the background distribution is estimated from several HSI blocks, n_f, under the assumption that they are free of anomalous signatures and hence, fully representative of the background distribution. On this basis, enough n_f hyperspectral frames must be taken to

ensure that all the spectral variability is covered and thus, to generate a trustworthy background model. However, this methodology could be not a feasible solution in very heterogeneous scenarios, which may be one limitation of the LbL-FAD detector.

3.5.6 Benchmarking performance among data types and precision: LbL-FAD vs HW-LbL-FAD

As it was comprehensively analysed in Chapter 2, the operations carried out by the modified Gram-Schmidt orthogonalization process and thus, by the proposed set of core operations, can be efficiently performed using both floating-point and integer arithmetic. Having studied the process followed by the HW-LbL-FAD algorithm, it can be concluded that the involved operations within the different stages of the algorithm correspond to the set of core operations or a subset of it. Accordingly, the HW-LbL-FAD algorithm can be implemented using the four algorithmic versions described in Section 2.5, namely *Float32*, *Int32*, *Int16* and *Int16-rd*. The first one uses floating-point arithmetic, as done in the original LbL-FAD algorithm, while the other two employ customized integer arithmetic by means of fixed-point notation with different levels of precision. As a reminder, these versions results according to the data type and precision for representing image values stored in the centralized version of the input image block, $\mathbf{C}$. The *Float32*, *Int32* and *Int16* versions are developed for working with HSIs whose element values could be represented with up to 16 bits per pixel per band and 256 spectral bands as maximum. In particular, the *Float32* version employs single precision floating point arithmetic (32 bits) for storing each element of $\mathbf{C}$. On the contrary, the *Int32* and *Int16* versions use customized fixed-point notation for representing fractional values within $\mathbf{C}$ with 32 and 16 bits, respectively. Although the *Int16* version turns into a very interesting option for applications with limited available hardware resources, above all in terms of RAM memory, some precision losses are introduced in the operations, which affects the quality of the results. For this reason, the *Int16-rd* model emerged as a performance-enhancing version derived from the original *Int16*. In this context, image values are saved with a resolution up to 12 bits per pixel per band (bpppb), though they are really stored using 16 bits per pixel per band (bpppb), padding four zeros at the beginning of each sample.

Table 3.3 gives an overview of the data precision, in number of bits, used for representing each algorithm variable for the four aforementioned versions of the HW-LbL-FAD algorithm. The main difference with respect to Table 2.2 introduced in Chapter 2 is that the

integer part of $\mathbf{v}$ is increased from 2 to 4 bits. This is because the brightness of image pixels from one hyperspectral frame to the other may be affected by the illumination conditions during the flight campaign. In this regard, $\mathbf{Q}$ and $\mathbf{U}$ vectors are extracted from the first n_f hyperspectral frames in the *Stage 2* of the algorithm. Under normal operating conditions, elements within the projection vectors, $\mathbf{v}_n$, are always in the range (-1,1]. Nonetheless, light conditions could fluctuate between new sensed frames during the flight mission. Therefore, it could derive in higher values of $\mathbf{v}_n$ elements when illumination conditions change during the acquisition of new hyperspectral frames whilst keeping $\mathbf{Q}$ and $\mathbf{U}$ vectors, which were estimated from previous frames obtained under other environmental conditions. Consequently, an increase of 2 bits for representing the integer part of $\mathbf{v}$ prevents overflowing even with an increment of the pixel brightness up to a factor of 4.

Variable	Integer part			Decimal part			Total			
	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Float32
$\mathbf{C}$	20	16	14	12	00	02	32	16	16	32
$\boldsymbol{\mu}$	16	16	12	00	00	00	16	16	12	32
$\mathbf{b}$	48	48	48	16	16	16	64	64	64	64
$\mathbf{q}$	20	16	14	12	00	02	32	16	16	32
$\mathbf{u}$	02	02	02	30	30	30	32	32	32	32
$\mathbf{v}$	04	04	04	28	28	28	32	32	32	32

TABLE 3.3: Number of bits used for representing the integer and decimal parts of the variables involved by the HW-LbL-FAD algorithm.

The detection performance of the proposed HW-LbL-FAD algorithm in its *Float32*, *Int32*, *Int16* and *Int16-rd* versions is evaluated in this Section. Additionally, the HW-LbL-FAD algorithm is also compared with the original LbL-FAD detector. Table 3.4 shows the obtained results in terms of TPR and FPR for the *Synthetic Image* (see Figure 3.2a), the *WASP RIT* scene (see Figure 3.2b) and the *AVIRIS WTC* scene (see Figure 3.2c). Although these images span the full dynamic range derived from 16 bit-depth, they have been scaled to reduce it to 12 bit-depth in order to be able to evaluate the *Int16-rd* version as well. Additionally, the input parameter n_f is also included in Table 3.4 since derived results depend on it.

Different conclusions can be drawn from the detection results displayed in Table 3.4. First of all, it can be observed that the HW-LbL-FAD algorithm executed using floating-point

arithmetic (*Float32*) results in the same outcomes than the original LbL-FAD algorithm (also using floating-point arithmetic). This verifies the fact that the operations performed by the HW-LbL-FAD algorithm are mathematically equivalent to the original LbL-FAD algorithm. Additionally, it is also confirmed that the *Int32* version of the HW-LbL-FAD algorithm is able to achieve the same performance than its *Float32* one, but being more suitable for those hardware devices that are more efficient using integer arithmetic. On the other hand, the *Int16* version of the HW-LbL-FAD algorithm generates slightly different results, but still achieving an accurate detection performance in relation to other state-of-the-art detectors, as those evaluated in Section 3.5.5, with the advantage of being a less computational demanding approach. By contrast, it is corroborated that the *Int16-rd* version offers significantly better quality compression results than previous *Int16* version, employing the same hardware resources. Additionally, deviations in the assessment metrics between the *Int32* and the *Int16-rd* versions are almost negligible, with the advantage of halving the memory space required for storing **C**.

Image	Inputs	LbL-FAD		HW-LbL-FAD							
				Float32		Int32		Int16		Int16-rd	
	n_f	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Synthetic Image	20	100	0,00	100	0	100	0	100	2,79	100	0
WASP RIT	60	93,06	0,09	93,06	0,09	93,06	0,09	79,17	0,06	93,06	0,09
AVIRIS WTC	30	43,37	0,02	43,37	0,02	43,37	0,02	44,58	0,02	43,37	0,02

TABLE 3.4: Detection performance of the original LbL-FAD algorithm and the four different versions proposed for the HW-LbL-FAD method. The *TPR* and the *FPR* are in percentage.

3.5.7 Computational complexity analysis

A comprehensive analysis about the computational complexity of the proposed set of core operations was made in Chapter 2. In this Section, it is extended for the particular case of the LbL-FAD algorithm, concretely the HW-LbL-FAD version, in order to evaluate the number of operations (OPs) required by each of its computing stages. Additionally, since the methodology followed by the OSPRX method and its detection performance are very similar to the HW-LbL-FAD algorithm, the overall number of OPs executed by these two anomaly detectors for processing the employed test bench has been also compared.

As it was explained in Section 3.3, the HW-LbL-FAD algorithm involves four main stages that can be actually grouped in two differentiated processes. The former can be seen as a *Training Stage* that covers the background modelling through the selection of the most representative pixels within this class and the computation of the orthogonal subspace to the one spanned by the background pattern (Stage 1 and 2 of the HW-LbL-FAD method). The latter can be seen as a *Detection Stage* where potential anomalous pixels are identified (Stage 3 and 4 of the HW-LbL-FAD method). Table 3.5 collects the number of OPs required by these two processes for the conventional images displayed in Figure 3.2 and for one of the images sensed by the UAV-based acquisition system, in particular the *Drone Image 1* shown in Figure 3.5a. It is worth stressing that the selection of the background representative vectors carried out under the *Training Stage* could be done off-line using the data from previous flights. Regarding this, the computational cost of the proposed HW-LbL-FAD algorithm drastically decreases since just the *Detection Stage* would be executed.

In order to compare the computational complexity of the HW-LbL-FAD method against the OSPRX detector, a brief description of the OSPRX method in terms of OPs is required. In general terms, overall operations performed by the OSPRX method could be divided into four main stages. Firstly, the covariance matrix of the entire HSI is computed. Considering that the image to be tested has nr rows, nc columns and nb bands, the overall number OPs required to compute the covariance matrix is $(2 \cdot nr \cdot nc + 1) \cdot (nb + nb^2)$. Secondly, the SVD problem is solved in order to decompose the covariance matrix into its eigenvectors and eigenvalues. The computation of the SVD has a long history and many improvements and techniques have been used to address it over the years. In this work, we have analysed the Divide and Conquer (D&C) process [158], which is one of the most widely used and fastest methods to settle the SVD for large matrices. According to [159], the overall number of FLOPS required by the D&C process to compute the SVD, including an initial QR decomposition, is $(6 \cdot m \cdot n^2 + 8 \cdot n^3\ )$ for a $m \times n$ matrix. For the particular case of the $nb \times nb$ covariance matrix, it ascends to $14 \cdot nb^3$. The third stage consists in the computation of the orthogonal subspace to the one spanned by the eigenvectors of the covariance matrix. On the basis that the SVD gives orthonormal vectors as output, this stage is equivalent to Stage 3 of the HW-LbL-FAD algorithm and the number of involved OPs ascends to $2 \cdot d_{OSP} \cdot nb^2$, being d_{OSP} the number of targeted first principal components. Finally, the fourth stage also matches with the Stage 4 of the HW-LbL-FAD algorithm whose overall number of computed OPs is $nr \cdot nc \cdot (2 \cdot nb^2 + nb)$.

Table 3.5 also collects the overall number of OPs undertaken by the HW-LbL-FAD algorithm and the OSPRX detector for the aforementioned data sets. For the particular case of the HW-LbL-FAD method, the number of extracted background reference vectors, p, is needed in order to evaluate its computational complexity. For the *Synthetic Image*, p ascends to 3; for the *WASP RIT* scene to 8 and for the *AVIRIS WTC* scene and the Drone Image 1 to 5. The last column of Table 3.5 shows the proportion of extra OPs required by the OSPRX algorithm compared with those required by the LbL-FAD algorithm, which is almost 370 times higher for the worst case and 15 times bigger at the best. As it can be seen, the differences are higher for images with larger number of nb.

Data Set	HW-LbL-FAD			OSPRX	$\frac{\text{OSPRX}}{\text{LbL}-\text{FAD}}$
	Training Stage	Detection Stage	Total		
Synthetic Image	2,63E+07	1,17E+08	1,43E+08	1,77E+10	123,42
WASP RIT	6,77E+07	8,81E+07	1,56E+08	1,90E+09	12,20
AVIRIS WTC	4,84E+07	2,06+E08	2,54E+08	9,90E+09	38,93
Drone Image 1	5,27E+08	2,61E+09	3,14E+09	8,70E+10	27,70

TABLE 3.5: Number of OPs required by the OSPRX and the HW-LbL-FAD methods for the Synthetic Image, the WASP RIT scene, the AVIRIS WTC scene and the Drone Image1.

3.6 Conclusions

A new real-time anomaly detector, named LbL-FAD, has been introduced in this Chapter for the purpose to fulfil the requirements imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners. In this regard, sensed hyperspectral frames are processed absolutely independently since no spatial connection between them has to be taken into consideration. The LbL-FAD algorithm works under the assumption that anomalous pixels cannot be well represented by the background distribution. Employing the set of core operations extensively analysed in this Thesis work, the LbL-FAD method focuses on the estimation of an orthogonal subspace to the one spanned by the background samples in which anomalous entities could be better distinguishable. For doing so, the LbL-FAD algorithm uses the aforementioned orthogonal projection strategy for extracting a set of reference background pixels from the first sensed hyperspectral frames, which are noted to be free of anomalous spectra. Afterwards, the LbL-FAD method exploits the computational advantages derived from the orthogonality to propose a low-complexity

alternative to compute the orthogonal subspace projection matrix, $\mathbf{P}$. Lastly, unlike other state-of-the-art anomaly detectors, the LbL-FAD algorithm also comes up with an automatic thresholding method that permits to detect the anomalous pixels in a line-by-line fashion as soon as each hyperspectral frame is processed. Consequently, it results in a binary map where anomalous targets are segmented from the background in real-time.

Furthermore, an alternative version of the LbL-FAD algorithm is also proposed in this Chapter, referred to as HW-LbL-FAD. This second approach makes a more efficient use of the introduced set of core operations, resulting in a mathematically equivalent algorithm to the LbL-FAD method, but resulting in a lower computational burden in terms of both OPs and memory requirements. Additionally, the HW-LbL-FAD algorithm can be executed using customized integer arithmetic at different precision levels that can be adapted for achieving the best relation between detection accuracy and computational burden. Concretely, four versions of the HW-LbL-FAD algorithm are proposed in this work. The first one (*Float32*) uses floating point arithmetic as the original LbL-FAD does, while the other three (*Int32*, *Int16* and *Int16-rd*) employ integer arithmetic and the fixed-point notation. From experiments made, it has been confirmed that the *Int32* and the *Int16-rd* versions lead to very similar results to the *Float32* version, but being more suitable for those hardware devices that are more suitable for working with integers. In the particular cases of the *Int16-rd* and the *Int16* versions, they also halves the memory space required for storing the input data.

Several simulations have been carried out using both synthetic and real hyperspectral data to verify the well detection performance and the low computational complexity of the LbL-FAD detector. To this end, the detection performance of the LbL-FAD algorithm has been compared with other state-of-the-art reference methods using statistical measures, such as the *TPR* and the *FPR*, and also under the detailed description of the resulting binary maps. Regarding the computational burden, it has also been evaluated in terms of the number of operations to be executed. The experimental results assert the superiority of the LbL-FAD detector in terms of both separability between classes and accuracy in the detection performance using a considerable less number of OPs and much simpler operations that can be easily parallelized. As a matter of fact, the LbL-FAD algorithm is able to implement between 12% and 100% less number of OPs than the well-known OSPRX algorithm, being ever higher as the number of spectral bands increases.

To sump up, the LbL-FAD algorithm shows the following advantages with respect to other state-of-the-art solutions for anomaly detection in hyperspectral imagery:

1. The LbL-FAD detector is conceived to perform anomaly detection in each hyperspectral frame acquired by a pushbroom sensor as soon as they are sensed. Therefore, each line of hyperspectral pixels can be independently processed without requiring any spatial alignment between pixels. In addition, the LbL-FAD algorithm shows low computational complexity and high level of parallelism. In this regard, the involved operations can be easily accelerated in parallel computing platforms ensuring a real-time performance, as it has been previously analysed in [33, 122, 160].

2. The LbL-FAD algorithm presents a hardware-friendly alternative to perform the orthogonal subspace projection matrix, $\mathbf{P}$, which discards any matrix inverse calculation by providing a set of orthogonal vectors. Alternatively, a second version of the algorithm, named HW-LbL-FAD, has been also introduced as a potential solution for those hardware devices that are more efficient dealing with integer operations. Consequently, this new equivalent approach is described using fixed-point number representation at different precision levels.

3. Unlike other state-of-the-art algorithms [138, 139, 161–163], the LbL-FAD algorithm does not use traditional linear transformation methods such as PCA, SVD or ICA to obtain the components that contain the most of the background information. These methods show important disadvantages that prevent their use in real-time applications due to the high computational cost, intensive memory requirements, high implementation costs and a non-scalable nature of the involved operations, such as eigenvalue decomposition, covariance matrix computation and matrix inverse computation [159, 164–168].

4. Several state-of-the-art anomaly detectors are based on the traditional RX detector whose main bottleneck is the performance of the inverse covariance/correlation matrix due to the dimensions of the matrices involved in the multiplications. For this reason, great efforts have been made to reduce the heavy computational load that requires its computation [44–47, 149]. The LbL-FAD method goes one step further, moving away from this problematic extensively discussed in the literature and, brings a hardware-friendly alternative. In this regard, elements involved in the multiplications to compute the orthogonal subspace projection matrix, $\mathbf{P}$, with the same dimension as the covariance matrix, is much smaller.

5. In general, the state-of-the-art algorithms result in gray-scale maps of probabilities where the anomalous target discrimination is usually carried out by visual inspection or thresholding after the whole image is processed. In contrast, the LbL-FAD

algorithm proposes an automatic thresholding method in which the threshold is obtained from the background data distribution. It enables the discrimination of the anomalous pixels in a line-by-line fashion as soon as each hyperspectral frame is received. Therefore, the LbL-FAD algorithm provides a binary map as output where anomalous entities are segmented from the background, avoiding the subjective human factor. In addition, an unitary map where all pixels are marked as background is given if no anomalous pixel are present.

Chapter 4

Hyperspectral Lossy Compression

In this Chapter, a performance-enhancing version of the state-of-the-art Lossy Compression Algorithm for Hyperspectral Image Systems (HyperLCA) is proposed for the spectral decorrelation and compression of hyperspectral images. In this regard, the original version of the HyperLCA algorithm is modified in order to employ the set of core operations proposed in this Thesis work for the data transformation. Additionally, it also widens the definition of the operations performed by the *HyperLCA Transform* with the purpose of analysing their suitability for being executed using integer arithmetic. The goodness of the HyperLCA algorithm for the lossy compression of hyperspectral images is evaluated using different quality metrics, such as the SNR, the MAD and the RMSE, among others, for different settings of the algorithm input parameters. Finally, it is demonstrated that the HyperLCA algorithm is able to preserve the most different pixels after the compression-decompression process, which is crucial for many hyperspectral imaging applications such as anomaly detection.

4.1 Rationale

Hyperspectral technology has gained relevance in the last years, mainly due to the emergence of compact-size aerial platforms, such as unmanned aerial vehicles (UAVs) and new space-borne missions. Its expansion and growing recognition have been largely propelled by the great richness of spectral information collected by this type of sensors. In exchange, one of the main challenges faced by this technology is the efficient handling of large amount of data that, on the one hand, jeopardizes the real time performance of making-decision applications and, on the other hand, demands large on-board storage resources. In addition, the latest technological advances are promoting to put into the market hyperspectral cameras with higher spectral and spatial resolutions, making it harder the efficient data management from an on-board processing, communication, and storage points of view [102, 103]. For all these reasons, an efficient on-board compression of hyperspectral images (HSIs) is mandatory, due to the huge amount of sensed data in order to save bandwidth and storage space. To face this issue, two main strategies may be followed: (1) proposing new hardware-friendly algorithms and (2) presenting the technologies and strategies to execute the compression in the available onboard hardware devices, minimizing the complexity and consequently, the resources usage and power consumption [107].

Traditionally, hyperspectral imagery sensed by spaceborne Earth-observation platforms are not onboard processed. As a matter of fact, they are onboard stored until their transmission to the ground segment where they are off-line processed by the end-users on supercomputing systems based on Central Processing Units (CPUs), Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), or heterogeneous architectures [33]. The main rationale behind this is the restrictions in the onboard power capacity and storage space, which promotes the use of low-power computing devices normally technology generations behind their commercial counterparts [34, 35, 38, 39, 70, 169]. Regarding airborne platforms, sensed data are normally onboard stored and thus, they cannot be accessed until the flight mission is over [170]. UAVs have also experienced a growing popularity, becoming one of the most powerful tools for the Earth observation due to their lower-cost and their more flexible revisit time than above mentioned acquisition systems. In this field, data handling is addressed similarly to airborne platforms, even though considerable efforts are being made to transmit them in real-time to the ground segment [171, 172].

Regrettably, the data transmission introduces important delays mainly related to the large data volume to be transferred, amounted to thousands of Giga-bytes (GBs), and the limited communication bandwidth of the data link, which in fact has been kept relatively stable over the years [38, 55, 173]. All of this, jointly with the steadily growing data-rate of the latest-generation sensors, make it compulsory to develop new low-complexity approaches for hyperspectral image compression that see high compression ratios with satisfactory rate-distortion performance. It prevents the unnecessary accumulation of high amount of uncompressed data and facilitates the efficient data handling and transfers.

Under this restricted scenario, it is becoming necessary to move from traditionally preferred lossless or near-lossless compression approaches to lossy compression techniques. Despite most of the former solutions bring a quite satisfactory rate-distortion performance, they provide very moderate compression ratios of about 2∼3:1 [65, 66] that nowadays are insufficient to deal with the high data-rate of the newest-generation sensors. Consequently, all of this has motivated the appearance of research works targeting lossy compression [106, 174–177].

In general terms, hyperspectral image compression algorithms exploit the redundancies in the spatial and/or spectral domains for reducing the amount of data with or without loosing information. The traditional scheme for performing the hyperspectral imagery compression consists of a spatial and/or spectral decorrelator, a quantization stage and an entropy coder. In this regard, the decorrelator can be transform-based or prediction-based. Transform-based approaches [106], such as Discrete Wavelet Transform (DWT) [174, 178] or the Karhunen-Loève Transform (KLT) [175, 176], are generally preferred to spatially/spectrally decorrelate images in lossy compression. By contrast, lossless compression is more efficiently performed by prediction-based methods [177], which must retain nearby or neighbouring sample spectra for estimating the prediction errors of the pixel under test. Consequently, it requires at least maintaining saved in memory the spectral information of a few spatial lines of hyperspectral pixels.

Although most of the state-of-the-art lossy compressors achieve very satisfactory performance in terms of rate-distortion, they are characterized by extremely high computational costs, intensive memory requirements and a non-scalable nature. It is because many solutions found in the literature are generalizations of existing two-dimensional (2D) image or video compression algorithms [179]. Due to all these reasons, current solutions are not appropriate for power-constrained applications with limited hardware resources, such as onboard compression [180, 181]. Against this background, low-complexity compression

schemes stand as the most practical solution for such restricted environments [177, 182–184].

In this context, the Lossy Compression Algorithm for Hyperspectral Image Systems (HyperLCA) [107] arose in response of the aforementioned existing limitations. The HyperLCA algorithm is a low-computational complexity transform-based alternative that provides high compression ratios with a good compression performance at a reasonable computational burden. As a further advantage, the HyperLCA algorithm permits compressing blocks of image pixels independently. This feature promotes, on the one hand, the reduction of the data to be managed at once besides the hardware resources to be allocated. On the other hand, the HyperLCA algorithm becomes a very competitive solution for most applications based on pushbroom/whiskbroom scanners, paving the way for real-time compression performance. This algorithm was first introduced in [2] but this previous work has been widened in this Thesis in order to use the proposed set of core operations for decorrelating the spectral information. In addition, the performance of the HyperLCA algorithm using integer arithmetic is also analysed in detail with the purpose of adapting it for those devices that are more efficient dealing with integer operations, such as FPGAs.

4.2 State-of-the-art in hyperspectral data compression

Currently, a broad range of solutions for the compression of HSIs can be found in the specialized literature. The hyperspectral compression techniques base their operating model on exploding the redundancies in the image samples in both the spatial and spectral domains. The ultimate objective is to find the relations that permit to represent the sensed information in another way using less amount of data. For doing so, two working tools exist: 2D coding, which only looks for redundancies in the spatial or the spectral domain, and three-dimensional (3D) coding, which analyses the hyperspectral cubes as a whole.

Additionally, loss of information could be introduced during the compression process. This gives rise to two types of algorithms: lossless and lossy methods. Lossless algorithms are normally preferred since data are completely preserved for later scientific purposes. Nonetheless, these strategies are able to reach very limited compression ratios that are

insufficient to cope with the higher data rates of the latest generation sensors. Consequently, the necessity of obtaining higher compression ratios will be critical in the near future. In this sense, lossy or near-lossless compression techniques could be a potential solution. However, whenever lossy or near-lossless methods are employed, it is necessary to evaluate the impact of the losses in the reconstructed data. In this sense, it has been observed that low average distortions after the compression process are not indicative of high quality reconstructed images when they are used in ulterior applications, such as classification, unmixing or anomaly detection [69, 185–188]. It is because lossy compressors normally behave as low-pass filters that, on the one hand, reduce the intrinsic image noise but, on the other hand, may remove some atypical elements of the image, negatively affecting the performance of the aforementioned hyperspectral analysis techniques.

Different techniques for hyperspectral imagery compression exist based on vector quantization [189–192], or more recently compressive sensing and learning-based methods [193–198], but prediction-based and transform-based methods prevail in the literature. Predictive coding consists of predicting the value of each element within a hyperspectral pixel from past data, generally neighbouring pixels in the spatial (intra-band) and/or spectral (inter-band) domains, quantizing the prediction error and entropy coding it. This compression type is mostly preferred for lossless compression although near-lossless or lossy compression may be also addressed by means of an appropriate quantization method [3]. Several methods based on prediction can be found in the literature but the Consultative Committee for Space Data Systems (CCSDS) has standardized the CCSDS 123-B-1 recommendation for the onboard lossless compression for multispectral and hyperspectral compression [199]. It is based on the Fast Lossless (FL) algorithm [200], which is a low-complexity adaptive filtering for predictive compression of HSIs based on a variant of the Least Mean Square (LMS) method [201]. Recently, this algorithm has been extended to perform near-lossless, denoting the new solution as Fast Lossless EXtended (FLEX) [202] that is the basis of the new CCSDS 123-B-2 standard for hyperspectral image compression [203].

In the field of lossy compression, transform-based decorrelators, such as the Discrete Wavelet Transform (DWT), the Discrete Cosine Transform (DCT), the Karhunen-Loève Transform (KLT) or the Principal Component Analysis (PCA), are generally preferred to spatially/spectrally decorrelate the HSIs. This transformation stage is followed by the quantization and encoding of the resulting coefficients. Nevertheless, transform-based hyperspectral imagery algorithms are usually extensions of 2D compression techniques. 3D alternatives involve concatenating a spatial transform with a transform in the spectral

domain. A popular approach is to apply a one-dimensional spectral decorrelator, such as KLT [175, 176] or the DWT [178], followed by the JPEG2000 standard for decorrelating the spatial information and performing the quantization and the entropy coding [204].

Among the existing transforms, the DCT is a decorrelator widely used in image and video compression standards, such as the JPEG and the H.264 respectively. This transform is advantageous as it is applied to an image in a block-by-block basis, similarly to the HyperLCA compressor proposed in this Thesis work. However, it also makes that the correlation across the block boundaries cannot be eliminated. Consequently, this results in apparent to the unaided eye blocking artifacts, particularly at low bit rates [205]. Alternatively, the DWT has become a benchmark in image compression. This decorrelator normally provides higher data reduction and better quality results. Unlike the above transforms, the DWT is applied to the whole image although it is computationally more complex from an implementation point of view [206].

Nevertheless, the transform-based lossy compression approaches based on the KLT [207], including the PCA transform, have been proven to yield the best results in terms of rate-distortion as well as in preserving the relevant information for the ulterior hyperspectral analysis [43, 67, 106, 208]. Despite the suitable results presented by the KLT approaches, they also show some deficiencies that prevent their use in restricted environment such as onboard scenarios. These drawbacks include high implementation costs, intensive memory requirements and a non-scalable nature. The PCA behaves similarly to the KLT but preserves better the data integrity after the compression process [67, 67, 106, 208]. Nonetheless, the PCA exposes the same disadvantages in terms of computational complexity than the KLT due to mainly the eigenvalue and eigenvector calculation process, whose complexity increases with the number of bands in the hyperspectral image. Among other methodologies derived from the KLT, the Pairwise Orthogonal Transform (POT) provides the best trade-off between coding performance and complexity [209].

The CCSDS has also defined the CCSDS 122.1-B-1 spectral preprocessing transform for 3D image compression standard [210]. This approach is able to work in both lossless and lossy operation modes, though its behaviour is far better for the latter operating mode. In this regard, the standard defines four different spectral transforms: the identity transform, the Integer Wavelet Transform (IWT), the POT and the Arbitrary Affine Transform (AAT). Then, the resulting image is compressed by multiple instances of a 2D DWT-based encoder, one per transformed band.

4.3 Lossy hyperspectral image compression with the HyperLCA algorithm

The HyperLCA algorithm is a lossy transform-based compressor for HSIs originally planed for satisfying the requirements imposed by current remote sensing hyperspectral imaging applications. Among its most important characteristics, it is emphasized the compression of image pixel blocks individually with no spatial strings attached. Consequently, this feature allows for the streaming of the compression process performed on the single hyperspectral frames collected by applications based on pushbroom or whiskbroom sensors. Additionally, this strategy also eliminates the need of large data volume storage required by other non causal state-of-the-art solutions, besides other advantages, such as the reduction in the hardware resources and the high speed-up promoted by the low-complexity involved operations. Apart from that, the HyperLCA compressor also guarantees a minimum user-defined compression ratio, which allows for knowing in advance the maximum data rate that will be attained after the acquisition and compression processes in order to efficiently manage the data transfers and/or storage. Additionally, the HyperLCA compressor provides quite satisfactory rate-distortion results for higher compression ratios than those achievable by lossless compression approaches.

Regarding the compression process, the HyperLCA compressor follows an unmixing-like strategy for selecting the most different pixels from the data to be compressed. These pixels are later pre-processed and lossless codified, so they are fully preserved after the compression-decompression process. This is a distinguishing characteristic of the HyperLCA method since rare anomalous spectra are highly preserved in comparison with other state-of-the-art lossy compression solutions that typically behave as low-pass filters. As a matter of fact, this feature leads to a correct operation of ulterior applications such as anomaly/target detection, classification, change detection, spectral unmixing, among others.

The compression process carried out by the HyperLCA compressor mainly involves two distinct parts: a spectral transform and a lossless codification stage. The *HyperLCA Transform* is the most computationally demanding part, but on the other hand, the highest compression ratios are reached in this stage at a low computational burden and a high level of parallelism. For doing so, the *HyperLCA Transform* employs the set of core operations proposed in this Thesis work for the extraction of the most characteristic pixels, $\mathbf{E}$, and their corresponding projection vectors $\mathbf{V}$, which results in a spectral uncorrelated

version of the original hyperspectral data. The number of selected vectors, p, is previously estimated in an initial stage according to the user-defined minimum desirable compression ratio. The codification stage is in charge of entropy encoding each vector selected by the *HyperLCA Transform* after being preprocessed beforehand. This stage slightly increases the compression ratio achieved by the *HyperLCA Transform* at a very low computational cost and without introducing additional losses of information. Figure 4.1 shows a graphic representation of the main computing stages involved by the HyperLCA compressor, which will be further analysed in Section 4.3.2.

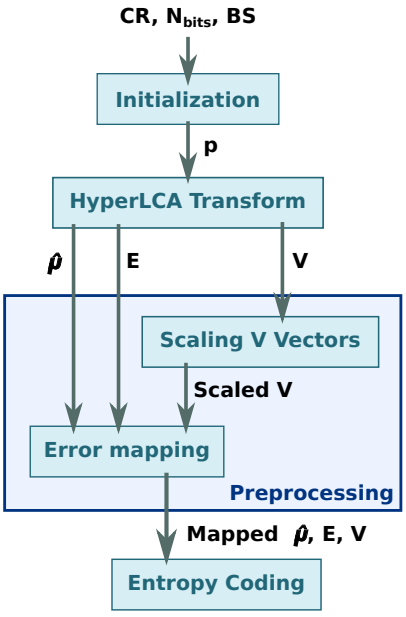


FIGURE 4.1: Data flow among the different computing stages of the HyperLCA compressor.

The HyperLCA algorithm was for the first time introduced in [2]. This previous work established the starting point towards the use of orthogonal projection techniques and, in particular the Gram-Schmidt method, for the spectral decorrelation and reduction of the hyperspectral data. In this sense, this first approximation focused more specifically on the *HyperLCA Transform* stage. As a novelty, the methodology described in [2] has been widened in this Thesis work in order to be adapted and, thereby, fallen within the proposed set of core operations. Moreover, the *Preprocessing* and *Entropy Coding* stages displayed in Figure 4.1 have been also included in order to obtain an efficient and comprehensive compression system. Finally, it has also analysed the feasibility of extending the initial definition of the *HyperLCA Transform* to be executed using integer arithmetic, in the interest of adapting it for those devices that are more efficient dealing with integer operations, such as FPGAs. For the sake of clarifying the novelties introduced

in this current work, a short overview about the contributions made by [2] is given in Section 4.3.1.

4.3.1 Background notions about the *HyperLCA Transform*

As it was already mentioned, the HyperLCA algorithm was first raised in [2]. This preliminary work analysed in detail the *HyperLCA Transform* in the interest of searching a competitive solution that would provide similar results to those obtained by the widely known PCA transform but, following a lower-complexity mathematical method. In this sense, its main purpose was to estimate a set of orthonormal vectors, $\mathbf{U*}$, in which to project the original HSI, $\mathbf{M}_k$, in order to obtained a spectrally decorrelated and reduced data set, $\mathbf{Mr}_k$. As it can be inferred, this set of orthonormal vectors, $\mathbf{U*}$, would act as the eigenvectors derived from the PCA analysis.

For doing this, the spectral transform defined by [2] was also based on the Gram-Schmidt method and indeed, follows the same sequence of the set of core operations proposed in this Thesis work. Nonetheless, unlike this previous work, the involved operations have been defined in this Thesis as a methodological point of view in the pursuit of contributing to the scientific community with a set of core operations that could be efficiently applied in other hyperspectral analysis fields as well, as those analysed along this document.

For the sequential extraction of the projection vectors, $\mathbf{u*}_n$, the hyperspectral data, $\mathbf{M}_k$, is firstly centered subtracting from all image pixels the average pixel, $\hat{\boldsymbol{\mu}}$, and thus obtaining matrix $\mathbf{C}$. Then, the p orthonormal vectors, $\mathbf{u*}_n$, are sequentially extracted as those with the highest brightness in each iteration. Nonetheless, the $\mathbf{u*}_n$ vector calculation is addressed as shows Equation 2.2 of Chapter 1, that is, dividing the selected pixel by its l^2-*norm*. In contrast, this fact has been modified in this current work for adapting it to the proposed set of core operations. Accordingly, $\mathbf{u}_n$ is not a unit vector in the proposed version of the HyperLCA compressor described in next Section 4.3.2 but, it is equivalent to $\mathbf{u}^*$ in the way as it was explained in Equation 2.3 of Chapter 1. In addition, this modification features a lower computational complexity. After that, the information that can be spanned by the selected $\mathbf{u*}_n$ vectors is subtracted from $\mathbf{C}$ as it is done by the Gram-Schmidt method.

Once that all projection vectors, $\mathbf{U*}$, have been calculated, the spectrally decorrelated and reduced data set, $\mathbf{Mr}_k$, is finally obtained as the projection of the real hyperspectral

image, $\mathbf{M}_k$, onto the subspace spanned by the set of estimated projection vectors as $\mathbf{Mr}_k = \mathbf{U*}' \cdot \mathbf{M}_k$. Therefore, the resulting image $\mathbf{Mr}_k$ is a compressed version of the original data since it retains the spectral information in p spectral bands, that is, the number of vectors in $\mathbf{U*}$, instead of using the initial nb wavelengths, being $nb >> p$. Finally, each pixel within $\mathbf{Mr}_k$ and $\mathbf{U*}$ is independently entropy coded using the standard CCSDS 121 lossless coder [211]. It is another differentiating point introduced by the performance-enhancing *HyperLCA Transform* version present in this Thesis work, as it is further analysed in the following Section 4.3.2.

4.3.2 Description of the extended version of the HyperLCA algorithm

In this Section, the different computing stages of the extended version of the HyperLCA algorithm proposed in this Thesis work are analysed in more detail. As mentioned above, these algorithm stages are displayed in Figure 4.1, in which the variables involved in the whole compression process are also represented.

4.3.2.1 HyperLCA Initialization

The number of the p most different pixels, $\mathbf{E}$, and projection vectors, $\mathbf{V}$, extracted from each image block, $\mathbf{M}_k$, directly depends on the compression ratio achieved by the HyperLCA algorithm. Additionally, this parameter is linearly dependant on some user-defined input parameters, as it can be seen from Equation 4.1. CR represents the minimum compression ratio to be reached for the targeted application, DR refers to the number of bits employed to represent each element of $\mathbf{M}_k$ and $\mathbf{E}$, and N_{bits} determines the number of bits used for representing the values of $\mathbf{V}$. As can it be deduced from Equation 4.1, the number of selected pixels, p, directly determines the maximum compression ratio to be reached with the selected algorithm configuration. Furthermore, bigger p results in better reconstructed images but lower compression ratios.

$$p \leq \frac{DR \cdot nb \cdot (BS - CR)}{CR \cdot (DR \cdot nb + N_{\text{bits}} \cdot BS)} \tag{4.1}$$

4.3.2.2 HyperLCA Transform

As mentioned before, the *HyperLCA Transform* employs the set of core operations described in Chapter 2, and more concretely in Algorithm 2, to obtain a compressed and uncorrelated image. As a reminder, the linear mixing model (LMM) is based on the idea that each pixel in a hyperspectral image, $\mathbf{r}_j$, can be represented as a linear combination of a set of p reference spectral signatures, $\mathbf{e}_n$, as shown in Equation 4.2, where $a_{\mathrm{j,n}}$ is the fractional area covered by each $\mathbf{e}_n$ in $\mathbf{r}_j$, and $\mathbf{n}_\mathrm{j}$ represents the noise contained in each image pixel, $\mathbf{r}_j$.

$$\mathbf{r}_\mathrm{j} = \sum_{n=1}^{p} \mathbf{e}_\mathrm{n} \cdot a_{\mathrm{j,n}} + \mathbf{n}_\mathrm{j} \tag{4.2}$$

On this basis, the p most different hyperspectral pixels, $\mathbf{E}$, and their corresponding projection vectors, $\mathbf{V}$, set as outputs of the set of core operations proposed in this Thesis, can be used to resolve to LMM issue. In this regard, a block of BS hyperspectral pixels, $\mathbf{M}_k$, may be also represented by $p \cdot (nb + np)$ elements, being $p << nb$. Consequently, we get a compressed image whose compression ratio proportionally depends on the number of selected pixels, p. Therefore, the smaller p extracted vectors, the higher compression ratios but, on the other hand, the bigger missing spectral information after the compression-decompression process. It is because $\mathbf{C}$ matrix, defined in Algorithm 2 of Chapter 2 where the set of core operations is displayed, retains the image spectral information that cannot be represented by the already selected p characteristic pixels. Indeed, this feature enables the possibility of easily providing a stopping condition according to different quality measures, such as the Signal-to-Noise Ratio (SNR) or the Maximum Single Error (MaxSE). In this scenario, the iteration process finishes either when the p $\mathbf{E}$ and $\mathbf{V}$ vectors are selected, or if this additional stopping condition is sooner accomplished.

In summary, the proposed set of core operations acts like a spectral transform where image pixels are projected onto a new subspace spanned by the p selected pixels, $\mathbf{E}$, and thus, it can be used to perform the lossy compression of hyperspectral perform. For the sake of clarity, Algorithm 6 displays the inverse *HyperLCA Transform* for reconstructing the compressed image blocks to obtain the decompressed hyperspectral image, $\mathbf{Mc}$. For this purpose, $\mathbf{Mc}$ is firstly initialized as the centroid pixel, $\hat{\boldsymbol{\mu}}$. This is because, the first step of the proposed set of core operations is the subtraction of $\hat{\boldsymbol{\mu}}$ from all pixels within $\mathbf{M}_k$ and hence, the projection vectors, $\mathbf{V}$, are also centered (see Line 1 of Algorithm 6).

For the same reason, $\mathbf{E}$ vectors are also centered in Line 2. The decompressed image, $\mathbf{Mc}$, is then obtained by sequentially adding the spectral information contained in $\mathbf{v}_n$ over each orthogonal vector, $\mathbf{q}$, as it can be seen from Lines 3-10. As it can be inferred analysing the addition shown in Line 6, the decompressed image, $\mathbf{Mc}$, is the result of a linear combination of the p selected pixels, $\mathbf{E}$, and the projection vectors, $\mathbf{V}$, similarly to the LMM.

As it can be also seen, the above described decompression process is clearly distinguishable from the initial methodology introduced by [2]. In this previous work, the decorrelated image, $\mathbf{Mr}_k$, was directly codified using the standard CCSDS 121 coder. Unlike it, only a limited set of vectors, $\mathbf{E}$, $\mathbf{V}$ and $\hat{\boldsymbol{\mu}}$, are handled in this current work, which also ensures the achievement of higher compression ratios than the aforementioned previous version.

Algorithm 6 Inverse HyperLCA Transform

Inputs:
$\hat{\boldsymbol{\mu}}$ {Average Pixel}; $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_p]$ {Characteristic pixels}; $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p]$ {Projection vectors}
Outputs:
$\mathbf{Mc}_k = [\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{BS}]$
Algorithm:

1: $\mathbf{Mc}_k = \hat{\boldsymbol{\mu}}$;
2: $\mathbf{E} = \mathbf{E} - \hat{\boldsymbol{\mu}}$;
3: **for** n = 1 **to** p **do**
4: $\quad \mathbf{q} = \mathbf{e}_n$;
5: $\quad \mathbf{u} = \mathbf{q}/(\mathbf{q}' \cdot \mathbf{q})$;
6: $\quad \mathbf{Mc}_k = \mathbf{Mc}_k + \mathbf{q} \cdot \mathbf{v}_n$;
7: $\quad$ **for** $j = n + 1$ **to** p **do**
8: $\quad\quad e_j = e_j - (u' \cdot e_j) \cdot q$
9: $\quad$ **end for**
10: **end for**

4.3.2.3 HyperLCA Preprocessing

Before entropy coding the *HyperLCA Transform* outputs, they must be firstly adapted to perform the posterior algorithm stages more efficiently. This transformation process is performed in two steps:

- Scaling $\mathbf{V}$ vectors:

$\mathbf{V}$ vectors actually collect the projection of each pixel within $\mathbf{M}_k$ over the direction spanned by each orthogonal vector, $\mathbf{u}_n$. Since the selected pixel in each iteration, $\mathbf{e}_n$, is the one worst represented by earlier extracted pixels, values of each element within $\mathbf{v}_n$ are typically in the range of $(-1, 1]$. Nonetheless, the following *Entropy-coding* stage is configured to deal only with integers. Therefore, elements within $\mathbf{V}$ must be scaled to fully exploit the dynamic range offered by the input parameter N_{bits}, as shown in Equation (4.3). After doing so, the scaled $\mathbf{V}$vectors are rounded up to the closest integer values.

$$v_{j_{\text{scaled}}} = (v_j + 1) \cdot (2^{N_{\text{bits}}-1} - 1) \tag{4.3}$$

- Error Mapping:

In general terms, the *HyperLCA Entropy Coding* stage exploits the redundancies among the data to be coded for assigning the shortest word length to the most common values. With it, the compression ratio achieved by the *HyperLCA Transform* is slightly increased. For this purpose, $\hat{\boldsymbol{\mu}}$ and each output vectors, $\mathbf{e}_n$ and $\mathbf{u}_n$, are individually lossless preprocessed and transformed in the *HyperLCA Error Mapping* stage to be exclusively composed of positive integer values closer to zero than the original ones. To that end, the HyperLCA algorithm uses the prediction error mapper described in the CCSDS recommended standard for lossless multispectral and hyperspectral image compression [211]. On the basis of the high spectral redundancies between contiguous spectral bands, the differences between adjacent elements ($\mathbf{y}_{j-1}$ - $\mathbf{y}_j$) of a generic hyperspectral pixel $\mathbf{y} = \{\mathbf{y}_j, j = 1, ..., nb\}$, such as $\hat{\boldsymbol{\mu}}$ or $\mathbf{e}_n$, is closer to zero than the $\mathbf{y}_{j-1}$ or $\mathbf{y}_j$ values themselves. Similarly it happens with single $\mathbf{v}_n$ vectors, although spatial redundancies are lower in hyperspectral imagery. The overall process works as follows. Firstly, the possible minimum and maximum values of each vector to be codified ($\mathbf{y_{min}}$ and $\mathbf{y_{max}}$) is estimated as $(-2^{(\text{DR}-1)}, 2^{(\text{DR}-1)} - 1)$ when $\mathbf{y}$ contains negative integer values, otherwise, $(0, 2^{\text{DR}} - 1)$, being DR the number of bits for representing $\mathbf{y}_j$ elements. Then, a threshold is calculated as $\theta_j = minimum(\mathbf{y}_{j-1} - \mathbf{y_{min}}, \mathbf{y_{max}} - \mathbf{y}_{j-1})$, which will be used to determine the way in which the prediction error, $\triangle_j = \mathbf{y}_j - \mathbf{y}_{j-1}$, is mapped as shown in Equation 4.4.

$$\mathbf{Y}_{j_{\text{mapped}}} = \begin{cases} 2\triangle_j & 0 \leq \triangle_j \leq \theta_j \\ 2\|\triangle_j\| - 1 & -\theta_j \leq \triangle_j < 0 \\ \theta_j + \|\triangle_j\| & \text{otherwise} \end{cases} \tag{4.4}$$

4.3.2.4 HyperLCA Entropy Coding

The *HyperLCA Entropy Coding* is the last stage of the HyperLCA compressor. It follows a lossless entropy-coding strategy based on the Golomb–Rice algorithm [212]. As in the *HyperLCA Error Mapping* stage, each single output vector is independently coded in the same order that they are obtained in the previous compression stages of the HyperLCA algorithm. This feature is a major advantage since it permits to pipeline the inputs and outputs of the different compression stages for a single block of pixels, $\mathbf{M}_k$.

For performing the codification process, the compression parameter, M, is estimated as the average value of the targeted vector. Afterwards, each of its elements is divided by M in order to obtain the results of the division, the quotient (q) and the remainder (r). On the one hand, the quotient, q, is codified using unary code. On the other hand, the remainder, r, could be coded using $b = log_2(M) + 1$ bits if M is power of 2. Nevertheless, M can actually be any positive integer. For this reason, the remainder, r, is coded as plain binary using $b - 1$ bits for r values smaller than $2^b - M$, otherwise it is coded as $r + 2^b - M$ using b bits.

4.3.2.5 Bitstream Generation

Finally, outputs of the previous compression stages are packed for each independently processed image block, $\mathbf{M}_k$, in order to generate the compressed bitstream that is sent to the Earth surface. The generated bitstream consists of two parts: the header, which contains the global information about the hyperspectral image that is needed to later decompress it, and $\frac{nr \cdot nc}{BS}$ data packages with the compressed data necessary to reconstruct each image block, $\mathbf{M}_k$.

Regarding the header, it collects the following data:

1. Size of the hyperspectral image; nc, nr and nb; representing the number of columns, rows and spectral bands respectively, coded as plain binary using 16 bits each.

2. The number of pixels within an image block $\mathbf{M}_k$, BS, coded as plain binary using 16 bits.

3. The number of hyperspectral pixels to be extracted during the compression process, p, coded as plain binary using 8 bits.

4. The number of bits per pixel per band used to save the sensed hyperspectral image, DR, coded as plain binary using 8 bits.

5. The number of bits per pixel per band used to save values of $\mathbf{V}$, N_{bits}, coded as plain binary using 8 bits.

6. One extra bit indicating if any additional stopping condition based on a quality metric has been used.

The second part of the bitstream that contains the information of each individual block of image pixels, $\mathbf{M}_k$, may be packed in two different ways, as shown in Figure 4.2c - 4.2d. The first option, graphically described in Figure 4.2c, is used when no additional stopping condition is used and fixed p $\mathbf{E}$ and $\mathbf{V}$ vectors are extracted for all pixel blocks, $\mathbf{M}_k$. On the contrary, the number of $\mathbf{E}$ and $\mathbf{V}$ vectors, p, in the second alternative may be different for each block of pixels, $\mathbf{M}_k$. In this situation, parameter p is also included in each data package, as shown in Figure 4.2d.

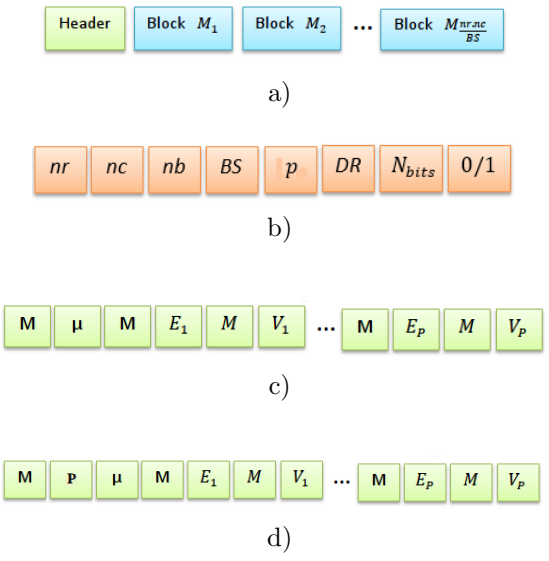


FIGURE 4.2: (a) General structure of the bitstream generated by the HyperLCA algorithm. (b) Header distribution. (c) Data package structure for non-stopping condition algorithm configuration. (d) Data package structure when a stopping condition based on quality metrics is enabled.

4.4 Experimental Results

In this Section, it is provided a concise and precise description of the experimental results carried out for evaluating the HyperLCA algorithm performance for the lossy compression of HSIs. The strengths and limitations of the proposed method have been extensively discussed through the analysis of a collection of quality metrics that assess the missing information introduced by the lossy compression process. For doing this, a set of six HSIs taken over real scenarios have been used. This test bench has been analysed using both floating-point and fixed-point notations with different levels of precision.

4.4.1 Reference Hyperspectral Data

The HyperLCA reliability for the lossy compression of HSIs has been evaluated using the bunch of real hyperspectral data collected by the acquisition platform described in [89] and further analysed in Section 3.4.1 of Chapter 3. For the sake of clarity, a brief summary about these image descriptions has been included in this Section, although we encourage the reader to see Chapter 3 to expand the details about the flight campaigns in which images were taken.

This data set was collected over multiple farming areas on the island of Gran Canaria (Spain) by a pushbroom sensor mounted on a UAV. In particular, the reference images are selected portions of some swaths within three different flight campaigns. These data cover the spectral information from 400 to 1000 nm using 160 spectral bands and consist of 825 lines height, each line comprising 1024 hyperspectral pixels with 12-bits depth. A RGB representation of these hyperspectral image portions are displayed in Figure 4.3. Images displayed in Figures 4.3 a–c were taken at a height of 72 m over the ground at a speed-rate of 6 m/s with a camera frame-rate of 125 frames per second (FPS), resulting in a ground sampling distance in line and across line of approximately 5 cm. Data shown in Figure 4.3 d was sensed in a second flight campaign performed at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm. Finally, frames exhibited in Figures 4.3 e–f were scanned at a flight height of 45 m over the ground and at a speed of 6 m/s with the hyperspectral camera capturing at 200 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm.

The aforementioned images were calibrated using a white and dark calibration to obtain reflectance values. Nonetheless, either orthorectification or georeferencing processes were not carried out for the acquired raw data. In this sense, images were built up just by placing the subsequent captured hyperspectral frames one next to the other [156]. This does not degrade the quality of the experiments carried out in this work since the tested algorithms do not use any kind of spatial information. A notable aspect of these images is the existence of some anomalous artefacts, such as some humans and concrete construction, which have been circled in blue in Figure 4.3.

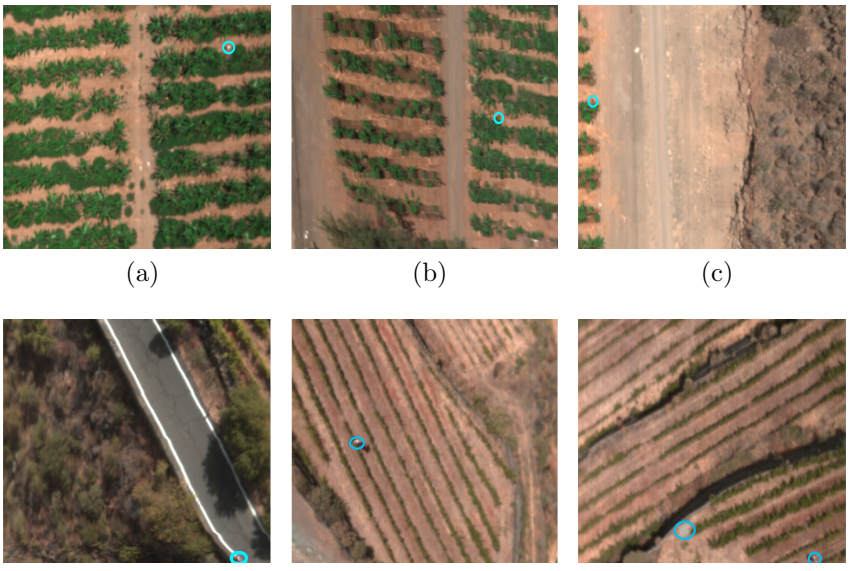


FIGURE 4.3: RGB representation of the employed test bench. Pixels enclosed in blue circles represent some anomalous spectra. (**a**) Drone Image 1. (**b**) Drone Image 2. (**c**) Drone Image 3. (**d**) Drone Image 4. (**e**) Drone Image 5. (**f**) Drone Image 6.

4.4.2 Assessment Metrics

The goodness of the HyperLCA algorithm for the compression of HSIs has been evaluated for its differing proposed versions in both floating-point and integer arithmetic. To this end, the hyperspectral imagery previously described in Section 4.4.1 has been compressed/decompressed using multiple settings of the HyperLCA input parameters (BS, N_{bits}, CR). As a matter of fact, the spectral distortions introduced by the missing information after the lossy compression process has been measured using different assessment metrics.

Firstly, the compression performance has been twofold evaluated by means of the achieved CR, measured as the ratio between the data volume before and after the compression, and the average number of bits per pixel per band, *bpppb*, used for representing the compressed images. Secondly, the missing data after the compression-decompression process have been also assessed using four different quality metrics: the *Signal-to-Noise Ratio (SNR)*, the *Root Mean Squared Error (RMSE)*, the *Maximum Absolute Difference (MAD)* and the *Structural Similarity Index (SSIM)*. These assessment metrics are calculated using the entire compressed-decompressed images (I and I_c) consisting of np pixels and nb bands each.

The *SNR* (Equation 4.5) evaluates the average missing information by means of the comparison between the average error and the maximum reached one. Accordingly, a higher *SNR* is indicative of a better compression performance. The *RMSE* (Equation 4.6) also analyses the average missing information although, unlike the *SNR*, a higher *RMSE* value means higher data losses and a weak compression performance. In contrast, the *MAD* (Equation 4.7) assesses the error for the worst reconstructed image element. Finally, *SSIM* (Equation 4.8) measures distortions as a combination of three factors: loss of correlation, luminance distortion and contrast distortion. Therefore, *SSIM* is defined as the product of the powers of these three similarities, as it can be seen in Equation 4.8 where α, β and τ have been set to 1. The dynamic range is between [-1,1], where 1 indicates a perfect structural similarity. Since *SSIM* is a quality assessment index originally designed for two-dimensional grey-scale images, we have applied it in a band-by-band manner to evaluate the quality of the HSIs. Therefore, a mean *SSIM* index over all spectral bands has been adopted in this work [213].

$$\mathrm{SNR} = 10 \cdot \log_{10}\left(\frac{\sum_{i=1}^{nb}\sum_{j=1}^{np}(I_{i,j})^2}{\sum_{i=1}^{nb}\sum_{j=1}^{np}(I_{i,j} - Ic_{i,j})^2}\right)(dB) \tag{4.5}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{np \cdot nb} \cdot \sum_{i=1}^{nb}\sum_{j=1}^{np}(I_{i,j} - Ic_{i,j})^2} \tag{4.6}$$

$$\mathrm{MAD} = \max(I_{i,j} - Ic_{i,j}) \tag{4.7}$$

$$SSIM = \frac{1}{nb}\sum_{i=1}^{nb}[(\frac{2\cdot \bar{I}_i \cdot \overline{Ic_i} + C1}{\bar{I}_i^2 + \overline{Ic_i}^2 + C1})^{\alpha} \cdot (\frac{2\cdot S_{I_i}\cdot S_{Ic_i} + C2}{S_{I_i}^2 \cdot S_{Ic_i}^2 + C2})^{\beta} \cdot (\frac{S_{IIc_i}^2 + C3}{S_{I_i}\cdot S_{Ic_i} + C3})^{\tau}] \quad (4.8)$$

$$\bar{I} = \frac{1}{np}\sum_{j=1}^{np} I_j \quad \overline{Ic} = \frac{1}{np}\sum_{j=1}^{np} Ic_j \quad S_I = \frac{1}{np-1}\cdot\sum_{j=1}^{np}(I_j - \bar{I})$$

$$S_{Ic} = \frac{1}{np-1}\cdot\sum_{j=1}^{np}(Ic_j - \overline{Ic}) \quad S_{IIc}^2 = \frac{1}{np-1}\sum_{j=1}^{np}(I_j - \bar{I})\cdot(Ic_j - \overline{Ic})$$

4.4.3 Benchmarking performance among data types and precision

Most of the compression performance achieved by the HyperLCA algorithm is obtained in the *HyperLCA Transform* stage, originally designed to be used with floating-point notation [2]. Nonetheless, FPGA devices are, in general, more efficient dealing with integer operations. Additionally, the execution of floating-point operations in different devices may produce slightly different results. For this reason, the performance of the HyperLCA algorithm using integer arithmetic is further discussed in this Section in order to adapt it for being more suitable for this kind of devices.

As a matter of fact, the operations performed by the *HyperLCA Transform* are those carried out by the modified Gram-Schmidt orthogonalization process and thus, through the set of core operations proposed in this Thesis work. As it was comprehensively analysed in Chapter 2, these operations may be efficiently executed using both floating-point and fixed-point integer arithmetic. Accordingly, the *HyperLCA Transform* stage can be implemented using the four algorithmic versions described in Section 2.5, namely *Float 32*, *Int32*, *Int16* and *Int16-rd*. The first one uses floating-point arithmetic, as done in the original version of the HyperLCA compressor, while the other three employ integer arithmetic with different levels of precision for representing the image values stored in the centralized version of the hyperspectral frame to be processed, **C**.

The *Float32*, *Int32* and *Int16* versions are developed for working with HSIs whose element values could be represented with up to 16 bits per pixel per band and 256 spectral bands as maximum. In particular, the *Float32* version employs single precision floating point arithmetic (32 bits) for storing each element of **C**. On the contrary, the *Int32* and the *Int16*

versions use customized fixed-point notation for representing fractional values within **C** with 32 and 16 bits, respectively. Although the *Int16* version turns into a very interesting option for applications with limited available hardware resources, above all in terms of RAM memory, some precision losses are introduced in the operations, which affects the quality of the results. For this reason, the *Int16-rd* model emerged as a performance-enhancing version derived from the original *Int16*. In this context, image values are saved with a resolution up to 12 bits per pixel per band (bpppb), though they are really stored using 16 bits per pixel per band (bpppb), padding four zeros at the beginning of each sample. Table 4.1 gives an overview of the data precision, in number of bits, used for representing each algorithm variable for the four aforementioned versions of the HyperLCA algorithm. It should be pointed out that when integer arithmetic is used, **V** vectors are directly represented using integer data types and do not need to be scaled and rounded to integer values in the *HyperLCA Preprocessing* stage.

Variable	Integer part			Decimal part			Total			
	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Int32	Int16	Int16-rd	Float32
C	20	16	14	12	00	02	32	16	16	32
$\boldsymbol{\mu}$	16	16	12	00	00	00	16	16	12	32
b	48	48	48	16	16	16	64	64	64	64
q	20	16	14	12	0	02	32	16	16	32
u	02	02	02	30	30	30	32	32	32	32
v	02	02	02	30	30	30	32	32	32	32

TABLE 4.1: Number of bits used for representing the integer and decimal parts of the variables involved by the *HyperLCA Transform*.

Figures 4.4 - 4.7 show a graphical representation of the average results obtained for each quality metric described in Section 4.4.2 for assessing the performance of the HyperLCA algorithm for the lossy compression of HSIs. The experiments have been made using different configurations of the algorithm input parameters, that is, $N_{bits} = (12, 8)$, $BS = (1024, 512, 256)$ and $CR = (12, 16, 20)$. Although these metrics will be analysed in more detail in next Section 4.4.4, we will now focus in particular on the comparative between the different proposed algorithm versions.

Some conclusions can be drawn from the aforementioned displays. In this regard, the quality of the compression results is very similar between the 32-bits fixed point proposal (*Int32*) and the single precision floating-point version (*Float32*) for all case studies, but being more suitable for those hardware devices that are more efficient using integer arithmetic. Additionally, it is corroborated that the *Int16-rd* version offers slightly better quality compression results than previous *Int16* version, employing the same hardware resources. Furthermore, deviations in the assessment metrics between the *Int32* and the *Int16-rd* versions are almost negligible, with the advantage of halving the memory space required for storing **C**. On the contrary, the compression performance obtained by the *Int16* version is not as competitive as the other three solutions, as it is analysed down below.

Regarding the performance of the *Int16* version, the most significant differences are observed for the *MAD* and the *RMSE* quality metrics displayed in Figures 4.5 - 4.6 and, even more for $N_{bits} = 8$. The rationale behind this is that initial values of **C** are divided by 2 in order to avoid overflowing, as it was described in Section 2.5 of Chapter 2, which directly decreases the precision in one bit. When N_{bits} is set to 8 bits, less number of bits is employed for scaling **V** vectors in the *Preprocessing* stage of the HyperLCA algorithm. Therefore, higher number of p characteristic vectors can be potentially selected according to Equation 4.1 described in Section 4.3.2.1. Nonetheless, the information contains in **C** matrix decreases in every iteration, n. Consequently, image values are increasingly smaller and differences among iterations can be disguised by the loss of precision introduced at the beginning. Therefore, no matter how many extra pixels are extracted since the missing information cannot be reconstructed during the decompression process. Unlike the *Int16* version, this fact is discarded in the *Int16-rd* version, although the same number of bits is used for representing the data, that is, **C** elements. It is because 2 extra bits for representing the integer part of the data are available in the *Int16-rd* version for these unlikely scenarios. Consequently, image precision is not altered as in previous *Int16* version for dealing with possible overflowing scenarios. Additionally, this new version also allows having 2 bits for representing the decimal part of the fixed-point values of matrix **C**, which is not possible in the *Int16* version.

Nevertheless, the *Int16* approach provides its best performance for very high compression ratios, small BS and images packaged using less than 16 bits per pixel per band. Indeed, the quality of the compression results for $BS = 256$ are closer to those provided by the other approaches regardless the compression ratio. This fact turns the *Int16* version of the *HyperLCA transform* in a very interesting option when the available hardware resources

are limited, since the main goal of this version is to reduce the amount of resources needed for the compression. Additionally, according to the quality metric values shown in Figures 4.4 - 4.7, it can be drawn that the compression performance of the *Int16* version is competitive enough, specially for power-constrained scenarios where high compression ration are demanded and hardware resources are restricted.

To sum up, it can be concluded that the HyperLCA compressor can be efficiently implemented using integer arithmetic, what is specially useful for the onboard spaceborne scenarios in order to carry out more efficient FPGA designs. The use of integer arithmetic is also useful in order to ensure that the *HyperLCA Transform* produces the same results whatever the computing device is employed, as recommended by the CCSDS 121.1-B-1 [210]. Additionally, the analysis carried out in this work for the different integer solutions can be extrapolated for other configurations. This allows to customize the HyperLCA algorithm according to the characteristics of the hyperspectral sensor acquiring the data and the hardware device available for the compression in order to achieve the most efficient solution.

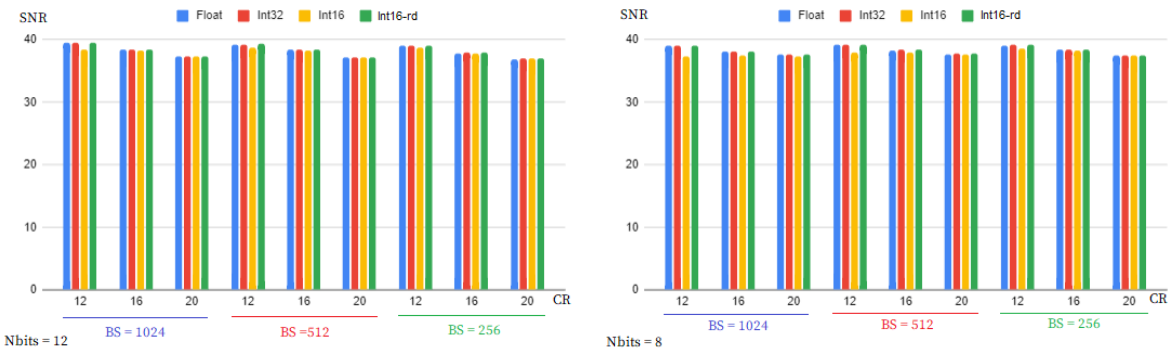
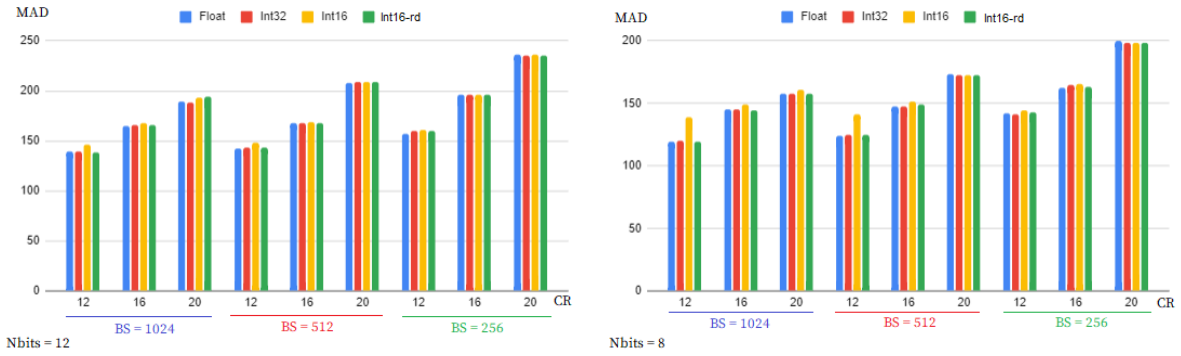


FIGURE 4.4: Average values of the *SNR* obtained for the *Float32, Int32, Int16* and *Int16-rd* versions of the HyperLCA algorithms. (a) $N_{bits} = 12$. (b) $N_{bits} = 8$.

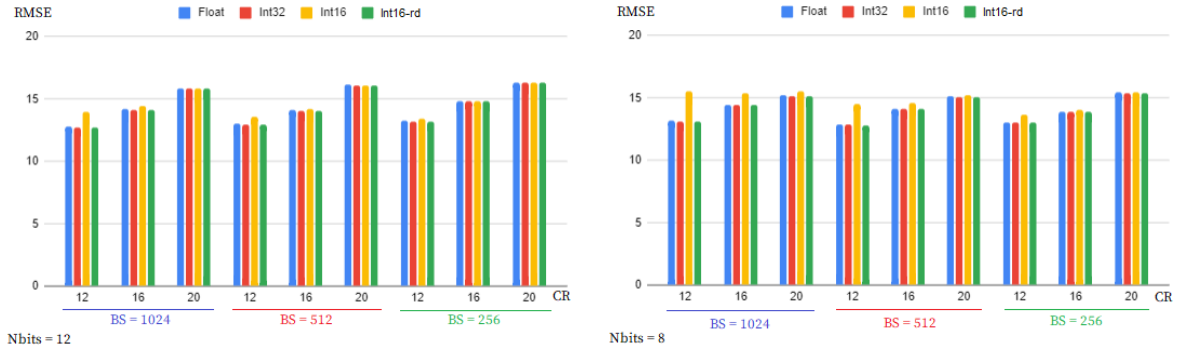
4.4.4 Compression performance of the HyperLCA algorithm

This section discloses the results obtained in all addressed experiments with the purpose of evaluating the goodness of the proposed HyperLCA algorithm for the lossy compression of HSIs. To this end, only the results obtained by the *Float32* version will be analysed for the sake of reducing the complexity of the evaluation.



(a) (b)

FIGURE 4.5: Average values of the *MAD* obtained for the *Float32, Int32, Int16* and *Int16-rd* versions of the HyperLCA algorithms. (a) $N_{bits} = 12$. (b) $N_{bits} = 8$.

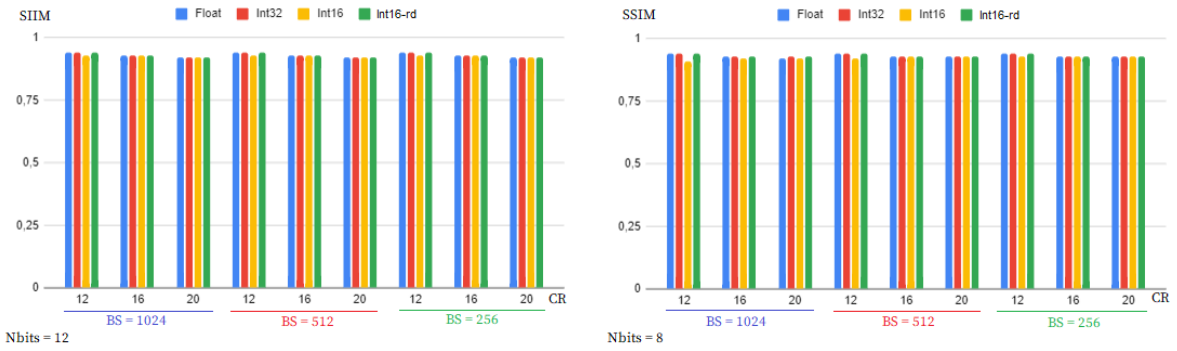


(a) (b)

FIGURE 4.6: Average values of the *RMSE* obtained for the *Float32, Int32, Int16* and *Int16-rd* versions of the HyperLCA algorithms. (a) $N_{bits} = 12$. (b) $N_{bits} = 8$.

4.4.4.1 Effect of the HyperLCA input parameters in the algorithm performance

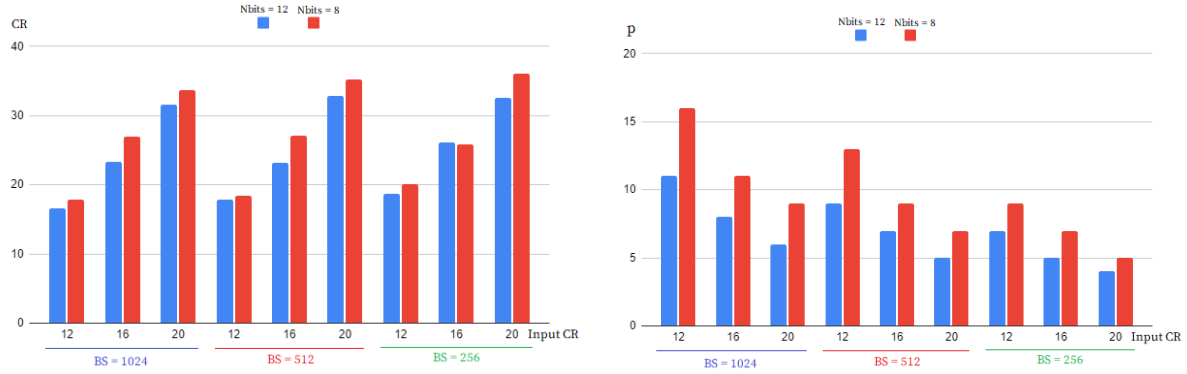
The HyperLCA algorithm has three main input parameters that may affect its compression performance: the number of pixels per block in which the image is divided, BS, the number of bits used for scaling the extracted $\mathbf{V}$ vectors, N_{bits}, and the minimum compression ratio to be desirable, CR. Different experiments have been done in order to evaluate the behaviour of the HyperLCA compressor when using different values for these parameters. In particular, the number of pixels per block, BS, has been set to 256, 512 and 1024 pixels, the dynamic range, N_{bits}, to 12 and 8 bits and, the minimum desirable compression ratio, CR, to 12, 16 and 20. On the one hand, Figure 4.8a graphically shows the distribution of the actual compression ratios achieved by the above mentioned input



(a) (b)

FIGURE 4.7: Average values of the *SSIM* obtained for the *Float32, Int32, Int16* and *Int16-rd* versions of the HyperLCA algorithms. (a) $N_{bits} = 12$; (b) $N_{bits} = 8$.

parameter settings. On the other hand, Figure 4.8b displays the tendency of the p parameter for the same case studies. The aforementioned plots represent the average results for the six data sets described in Section 4.4.1. The specific outcomes for each data set in particular are collected in Table 4.2.



(a) (b)

FIGURE 4.8: Evaluation of the HyperLCA compression results according to the average results in terms of (a) the reached compression ratios (CR) and (b) the number of extracted characteristic pixels (p) per image block, $\mathbf{M}_k$.

At a first glance, it can be noted that the HyperLCA algorithm always gives rise to higher compression ratios than those established as inputs. For instance, for an input $CR = 12$, the range of achieved compression ratios are between 16.30 and 20.07. For the most restricted scenario, input $CR = 20$, the achieved compression ratios ascend to 36 on average. In addition, all analysed algorithm settings employ less than 1 bpppb for representing each element within the compressed images.

Nbits	BS	CR	p	Drone Image 1		Drone Image 2		Drone Image 3		Drone Image 4		Drone Image 5		Drone Image 6	
				CR	*bpppb*	*CR*	*bpppb*	*CR*	*bpppb*	*CR*	*bpppb*	*CR*	*bpppb*	*CR*	*bpppb*
12	1024	12	11	16.30	0.74	16.47	0.73	16.49	0.73	17.56	0.68	16.47	0.73	16.31	0.74
		16	8	22.80	0.53	23.15	0.52	23.27	0.52	24.80	0.48	22.95	0.52	22.76	0.53
		20	6	30.85	0.39	31.48	0.38	31.88	0.38	33.94	0.35	30.93	0.39	30.81	0.39
	512	12	9	17.77	0.68	18.04	0.67	17.87	0.67	18.55	0.65	17.62	0.68	17.44	0.69
		16	7	23.04	0.52	23.47	0.51	23.23	0.52	24.16	0.50	22.77	0.53	22.56	0.53
		20	5	32.61	0.37	33.37	0.36	33.02	0.36	34.35	0.35	32.02	0.37	31.84	0.38
	256	12	7	18.72	0.64	19.10	0.63	19.18	0.63	19.08	0.63	18.12	0.66	17.99	0.67
		16	5	26.18	0.46	26.80	0.45	26.91	0.45	26.74	0.45	25.23	0.48	25.09	0.48
		20	4	32.61	0.37	33.47	0.36	33.63	0.36	33.34	0.36	31.32	0.38	31.21	0.38
8	1024	12	16	17.42	0.69	17.63	0.68	17.60	0.68	19.38	0.62	17.85	0.67	17.58	0.68
		16	11	26.17	0.46	26.61	0.45	26.63	0.45	29.50	0.41	26.61	0.45	26.19	0.46
		20	9	32.56	0.37	33.25	0.36	33.41	0.36	37.07	0.32	32.99	0.36	32.49	0.37
	512	12	13	18.15	0.66	18.49	0.65	18.32	0.66	19.30	0.62	18.10	0.66	17.82	0.67
		16	9	26.77	0.45	27.39	0.44	27.01	0.44	28.59	0.42	26.46	0.45	26.05	0.46
		20	7	34.89	0.34	35.88	0.33	35.29	0.34	37.46	0.32	34.28	0.35	33.80	0.36
	256	12	9	20.07	0.60	20.58	0.58	20.74	0.58	20.60	0.58	19.29	0.62	19.10	0.63
		16	7	25.86	0.46	26.59	0.45	26.74	0.45	26.55	0.45	24.74	0.49	24.50	0.49
		20	5	36.16	0.33	37.35	0.32	37.54	0.32	37.21	0.32	34.38	0.35	34.11	0.35

TABLE 4.2: Compression Results. Achieved CR and bppp for the six datasets.

According to the results shown in Figure 4.8b, $N_{bits} = 8$ results in higher p selected pixels than $N_{bits} = 12$. It makes sense since N_{bits} is part of the denominator in Equation 4.1. Although this may give the idea that smaller CR are reached with $N_{bits} = 8$, due to the fact that more p characteristics pixels are selected, in fact, the opposite is happened, as it can be seen from Figure 4.8a. It is because less number of bits are used for coding $\mathbf{V}$ vectors in the generated bitstream. In addition, the *HyperLCA Entropy Coding* also introduces some data compression additional to that obtained in the *HyperLCA Transform*. In this sense, the gaps among data that could be represented using $N_{bits} = 8$ bits is bigger than employing $N_{bits} = 12$ bits, in concrete 0.0039 vs 0.000244 in decimal notation respectively. It means that the *Entropy Coder* is able to code those vectors represented with $N_{bits} = 8$ using less number of bits than with $N_{bits} = 12$ bits.

Regarding BS, when the image is split up in smaller groups, smaller BS values, a set of orthogonal vectors is obtained for each one and hence, each image block has to be compressed using less p vectors for obtaining the same compression ratio, as can be seen from Figure 4.8b. In relation with the compression ratio, CR, little variations are shown among different BS settings, though a slightly higher CR are obtained for smaller BS due to the above mentioned reasons.

Nbits	BS	CR	Drone Image 1				Drone Image 2				Drone Image 3			
			SNR	*MAD*	*RMSE*	*SSIM*	*SNR*	*MAD*	*RMSE*	*SSIM*	*SNR*	*MAD*	*RMSE*	*SSIM*
12	1024	12	44.33	94.00	8.87	0.98	44.14	97.00	8.10	0.98	44.63	74.00	6.98	0.97
		16	42.60	141.00	10.82	0.98	42.59	115.00	9.68	0.97	43.45	85.00	8.00	0.97
		20	40.94	173.00	13.10	0.97	41.10	148.00	11.50	0.96	42.39	117.00	9.04	0.96
	512	12	43.97	98.00	9.24	0.98	43.64	94.00	8.58	0.97	44.32	76.00	7.24	0.97
		16	42.68	130.00	10.72	0.98	32.45	113.00	9.84	0.97	43.48	100.00	7.97	0.97
		20	40.48	182.00	13.81	0.97	40.44	147.00	12.41	0.96	42.22	148.00	9.22	0.96
	256	12	43.11	101.00	9.12	0.96	34.93	247.00	20.70	0.92	43.88	97.00	7.61	0.97
		16	41.28	167.00	12.60	0.97	41.26	165.00	11.29	0.97	42.74	153.00	8.68	0.96
		20	39.60	228.00	15.28	0.97	39.69	214.00	13.52	0.96	41.87	210.00	9.60	0.96
8	1024	12	44.36	80.00	8.84	0.98	43.76	70.00	8.47	0.98	43.40	63.00	8.05	0.97
		16	42.88	96.00	10.48	0.98	42.49	99.00	9.79	0.97	42.54	84.00	8.89	0.97
		20	42.07	115.00	11.50	0.98	41.80	104.00	10.61	0.97	42.07	90.00	9.38	0.97
	512	12	44.26	85.00	8.94	0.98	43.68	76.00	8.55	0.98	44.05	66.00	7.47	0.97
		16	42.86	105.00	10.50	0.98	42.41	93.00	9.89	0.97	43.11	84.00	8.32	0.97
		20	41.83	135.00	11.82	0.98	41.49	128.00	11.00	0.97	42.46	104.00	8.96	0.97
	256	12	43.61	101.00	9.63	0.86	43.28	95.00	8.95	0.98	44.09	91.00	7.43	0.97
		16	42.57	124.00	10.86	0.98	42.31	95.00	8.95	0.98	43.39	103.00	8.06	0.97
		20	40.83	166.00	13.26	0.97	40.72	170.00	12.01	0.96	42.35	159.00	9.08	0.96

TABLE 4.3: Compression Results. Achieved *SNR, MAD, RMSE* and *SSIM* for Drone Image 1, Drone Image 2 and Drone Image 3.

4.4.4.2 Evaluation of the HyperLCA performance for the lossy compression of HSIs

The HyperLCA algorithm spectrally decorrelates the information contained in the HSIs projecting the image pixels onto a smaller subspace composed of p projection vectors. The amount of the data that can be spanned by this subspace is directly proportional to the number of projection vectors extracted by the *HyperLCA Transform* and therefore, the extent of missing data after the compression-decompression process. In this Section, the quality of the compression results obtained after the spectral data reduction is evaluated in terms of the rate-distortion introduced by the HyperLCA compressor. For doing so, the HyperLCA performance is tested analysing the assessment metrics defined in Section 4.4.2, that is, the SNR, the MAD, the $RMSE$ and the $SSIM$, in a similar way to that was made in previous Section 4.4.4.1. The results obtained for these metrics are displayed in Tables 4.3 and 4.4 for the six data sets described in Section 4.4.1. For the sake of clarity, average results are also shown in Figure 4.9 for the different settings of the algorithm input parameters.

According to the obtained results, it can be observed that the proposed method is able to reach very high compression ratios with a good rate-distortion for all tested configurations. On average, the SNR ascends to 38.26 with the minimum and maximum values ranging

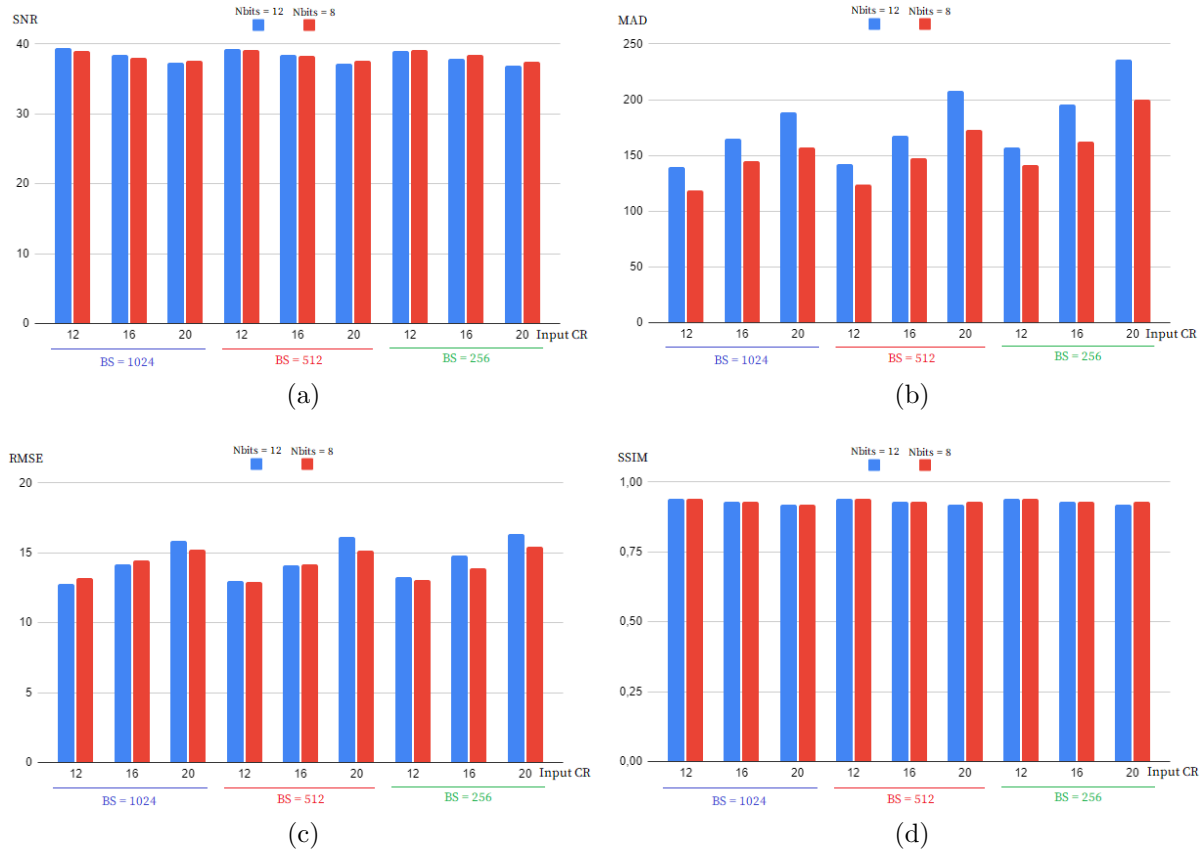


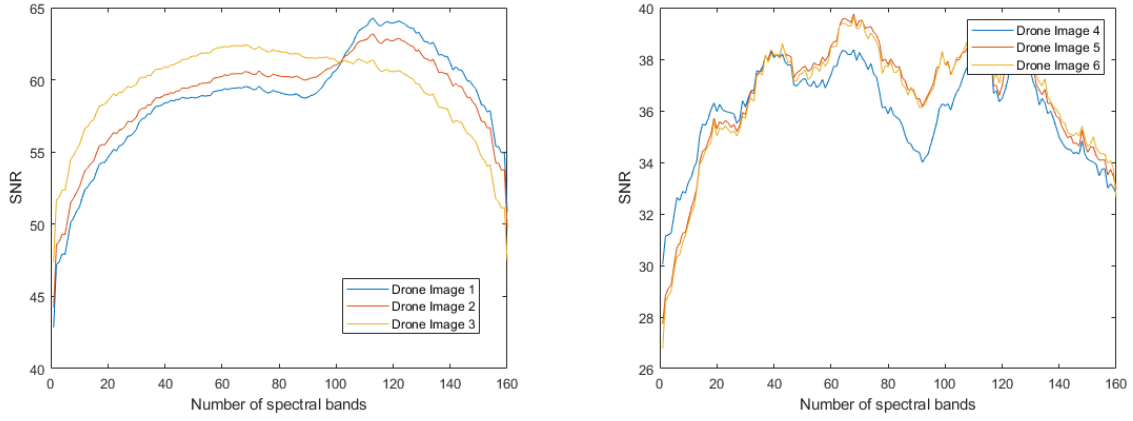
FIGURE 4.9: Evaluation of the HyperLCA compression results according to the average results in terms of (a) *SNR*, (b) *MAD*, (c) *RMSE* and (d) *SSIM*.

from 32.33 to 44.63. In terms of the MAD, the average result rounds the 4% of the maximum possible $(2^{12} - 1)$, ranging from 63 to 280. Regarding the $RMSE$, the average result is over 14.33, going from 6.98 to 25.59. In relation to the $SSIM$, the average result ascends to 0.93, ranging from 0.83 to 0.98. In general terms, the assessment metrics are better for *Drone Image 1* - *Drone Image 3* than for *Drone Image 4* - *Drone Image 6*. It is because the random noise inherent to the data set, which was acquired in three different flight missions. Figures 4.10a - 4.10b show the *SNR* distribution per band for the six HSIs, which was estimated as recommended by [81, 214]. As can be noted, the *SNR* is more compromised for the second set of HSIs.

In addition, the highest-quality results are obtained for the biggest BS. This is due to the fact that when the image is split up in smaller groups, smaller BS values, a set of orthogonal vectors is obtained for each image block, and hence, each block has to be compressed using less projection vectors for obtaining the same compression ratio. If there is little variability among pixels, it is possible to compress many of them, bigger BS values, with a single set of projection vectors with the same accuracy, obtaining a higher

Nbits	BS	CR	Drone Image 4				Drone Image 5				Drone Image 6			
			SNR	*MAD*	*RMSE*	*SSIM*	*SNR*	*MAD*	*RMSE*	*SSIM*	*SNR*	*MAD*	*RMSE*	*SSIM*
12	1024	12	34.50	124.00	11.12	0.87	34.99	212.00	20.55	0.92	34.26	234.00	20.95	0.90
		16	33.93	147.00	11.86	0.86	34.42	238.00	21.96	0.91	33.53	265.00	22.77	0.89
		20	33.18	177.00	12.93	0.85	33.80	250.00	23.58	0.90	32.73	269.00	24.98	0.88
	512	12	34.44	141.00	11.19	0.86	34.86	220.00	20.88	0.91	34.26	227.00	20.95	0.91
		16	33.98	156.00	11.80	0.86	34.41	264.00	21.99	0.91	33.74	278.00	24.33	0.89
		20	33.05	224.00	13.13	0.85	33.70	271.00	23.84	0.90	32.96	278.00	24.33	0.89
	256	12	34.44	138.00	11.19	0.86	34.93	247.00	20.70	0.92	33.32	243.00	20.81	0.91
		16	33.80	162.00	12.04	0.86	34.37	267.00	22.07	0.91	33.69	261.00	22.36	0.90
		20	33.22	220.00	12.88	0.85	33.94	280.00	23 19	0.91	33.23	264.00	25.59	0.90
8	1024	12	32.06	122.00	14.71	0.84	35.47	190.00	19.45	0.92	34.87	189.00	19.53	0.91
		16	31.73	138.00	15.28	0.83	34.83	216.00	20.94	0.92	34.09	237.00	21.36	0.90
		20	31.54	155.00	15.62	0.83	34.49	226.00	21.78	0.91	33.67	255.00	22.43	0.90
	512	12	33.04	118.00	13.14	0.85	35.35	199.00	19.72	0.92	34.83	199.00	19.63	0.91
		16	32.64	160.00	13.77	0.84	34.74	216.00	21.17	0.91	34.12	228.00	21.29	0.91
		20	32.33	166.00	14.27	0.84	34.30	259.00	22.26	0.91	33.62	247.00	22.55	0.90
	256	12	33.69	138.00	12.19	0.86	35.23	215.00	19.99	0.92	34.67	240.00	19.97	0.91
		16	33.37	141.00	12.66	0.86	34.84	251.00	20.92	0.92	34.23	246.00	21.03	0.91
		20	32.86	169.00	13.42	0.85	34.29	270.00	22.28	0.91	33.62	264.00	22.56	0.90

TABLE 4.4: Compression Results. Achieved *SNR, MAD RMSE* and *SSIM* for Drone Image 4, Drone Image 5 and Drone Image 6.



(a) (b)

FIGURE 4.10: SNR distribution per spectral band: (a) Drone Image 1, Drone Image 2 and Drone Image 3, (b) Drone Image 4, Drone Image 5 and Drone Image 6.

compression ratio. On the other hand, smaller BS settings may lead to better results when there is too much variability between pixels.

The MAD assessment metric measures the maximum reconstruction error among all image elements. As it can be noted from Figure 4.9b, MAD metric reaches bigger values for $N_{bits} = 12$. This makes totally sense since less p projection vectors are selected, as it was analysed in previous Section 4.4.4.1 and it can be plainly see in Figure 4.8b. It means

that the remaining **C** matrix defined within the set of core operations contains much more spectral information that cannot be reconstructed after the compression process.

Analysing the global distortions through the *SNR* and the *RMSE* metrics, results follow the logical tendency; the higher CR, the greater reconstruction errors. In addition, Figure 4.9c also reveals that although configurations with greater N_{bits} values obtain in general better decompressed images, this situation turns over for higher CR and smaller N_{bits} values.

In conclusion, the HyperLCA compressor stands as a quite competitive solution for the lossy compression of HSIs that offers high compression ratios introducing little losses of information. Additionally, its compression performance is also very solid and steady for the six different images, which contain a high quantity of random noise due to the fluctuations in the trajectory of the aerial acquisition system.

4.4.5 Evaluation of the distortions introduced by the lossy compression for the subsequent anomaly detection

Generally speaking, anomalies are considered as groups of rare and not abundant pixels whose spectral signature significantly differs from their surroundings. Normally, their existence may be indicative of abnormal or suspicious behaviour. For this reason, it is crucial the early detection of these atypical anomalous patterns. However, preserving these pixels through the lossy compression-decompression process represents a challenging task. It is because this kind of pixels covers a small fraction of the entire image and therefore, there is a high likelihood that anomalous spectra are lost after the compression/decompression process since most of existing lossy compression solutions behave as low-pass filters. Although the HyperLCA algorithm is also a lossy compressor, it has been specially designed for preserving the most different pixels within each image block to be independently processed, $\mathbf{M}_k$, and hence, retains the anomalous spectra.

On this basis, a detailed analysis of how the distortions introduced by the HyperLCA algorithm after the lossy compression process affects to the posterior anomaly detection. To this end, anomaly detection is performed over the images compressed by the HyperLCA algorithm for different input parameter settings. For doing so, the Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (LbL-FAD), described in Chapter 3 of this Thesis work, is used for the identification of the anomalous spectra. Figures 4.11 - 4.13

display the 2D binary maps obtained by the LbL-FAD algorithm employing input images compressed by the HyperLCA algorithm configured with N_{bits} = (12, 8), CR = (12, 16, 20) and BS = (1024, 512, 256), respectively. For the sake of clarity, these detection maps are superimposed on a panchromatic representation of the scenes in order to make easier the result interpretation. In these displays, lines in blue color indicate the n_f frames employed to estimate the background pattern, while red color highlights spatial lines corrupted by anomalous entities. In addition, those anomalous pixels detected by the LbL-FAD algorithm have been also locked up in red circles.

Compared to the RGB representations displayed in Figure 4.3 where the location of the anomalous objects has been pointed out inside blue circles, it can be seen that the HyperLCA compressor is able to accurately preserve these strange entities since all of them have been detected for all case studies. In general terms, the detection results are more accurate for *Drone Image 1*, *Drone Image 2* and *Drone Image 3* since random noise inherent to these images is more reduced than the other set of images, as it can be seen from Figure 4.10. A case in point is that in some noisy cases, such as *Drone Image 5*, higher CR improves the performance of the ulterior anomaly detection process since the random noise cannot be reconstructed. In this sense, the *HyperLCA Transform* acts as a noise filter. It is also worth mentioned that more accurate detection performance are obtained for bigger groups of pixels, BS = 1024, which matches with the behaviour of the HyperLCA compressor.

Finally, it may be concluded that the use of the same core operations for the definition of both the HyperLCA compressor and the LbL-FAD algorithm guarantees that the compression process does not seriously affect the posterior anomaly detection performance when it is off-board executed using the compressed/decompressed data. As a consequence, it permits tailoring to different scenarios that impose different requirements, ensuring the same results in all situations.

4.4.6 Computational complexity analysis

A comprehensive analysis about the computational complexity of the proposed set of core operations was made in Chapter 2. In this Section, it is extended for the particular case of the HyperLCA algorithm in order to evaluate the number of operations (OPs) required by each of its computing stages. Table 4.5 summarizes the number of OPs required by the *HyperLCA Transform* and the *HyperLCA Entropy Coding* stages to process a single block

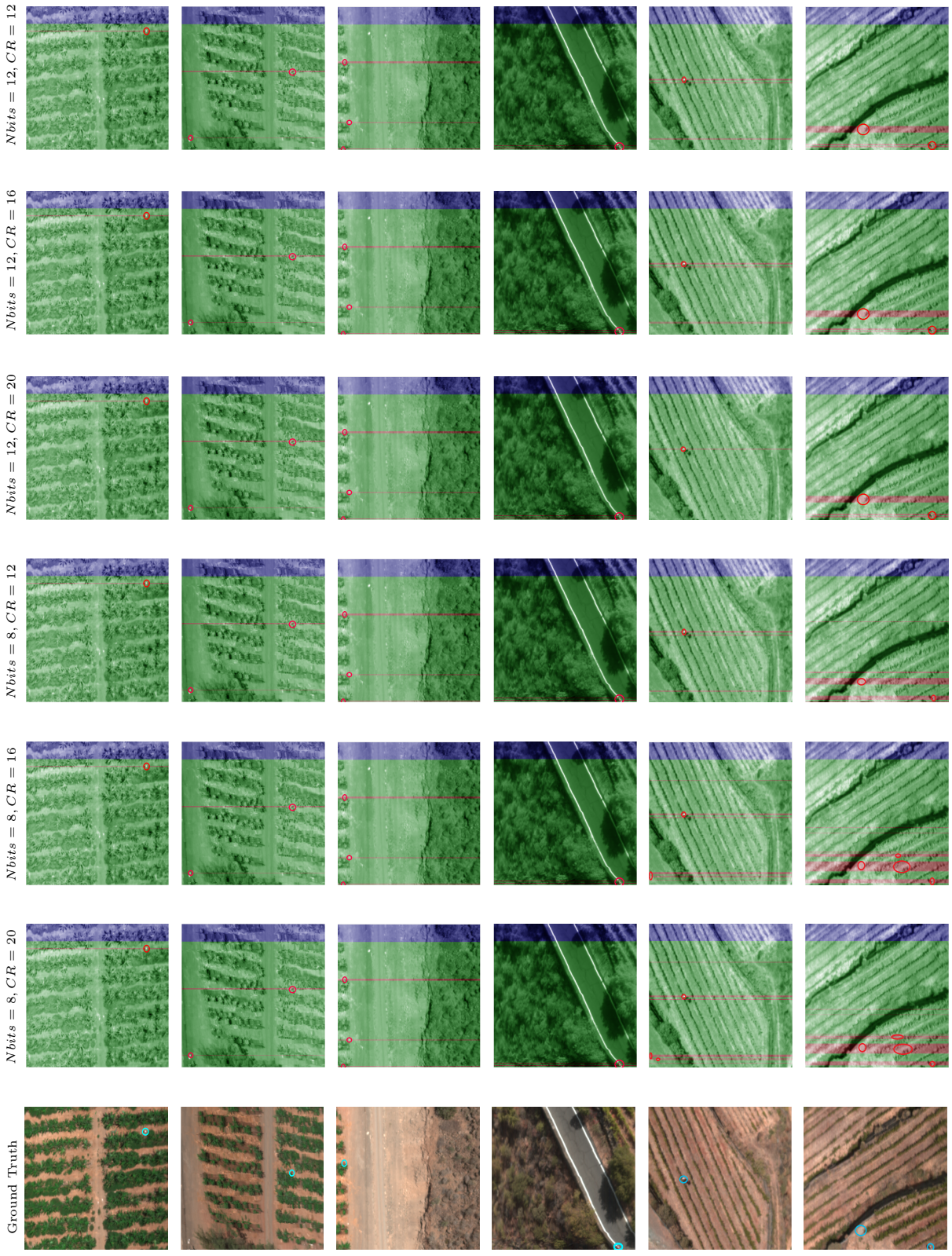


FIGURE 4.11: Anomaly detection results obtained by the LbL-FAD algorithm using the compressed images obtained by the HyperLCA algorithm for $BS = 1024$.

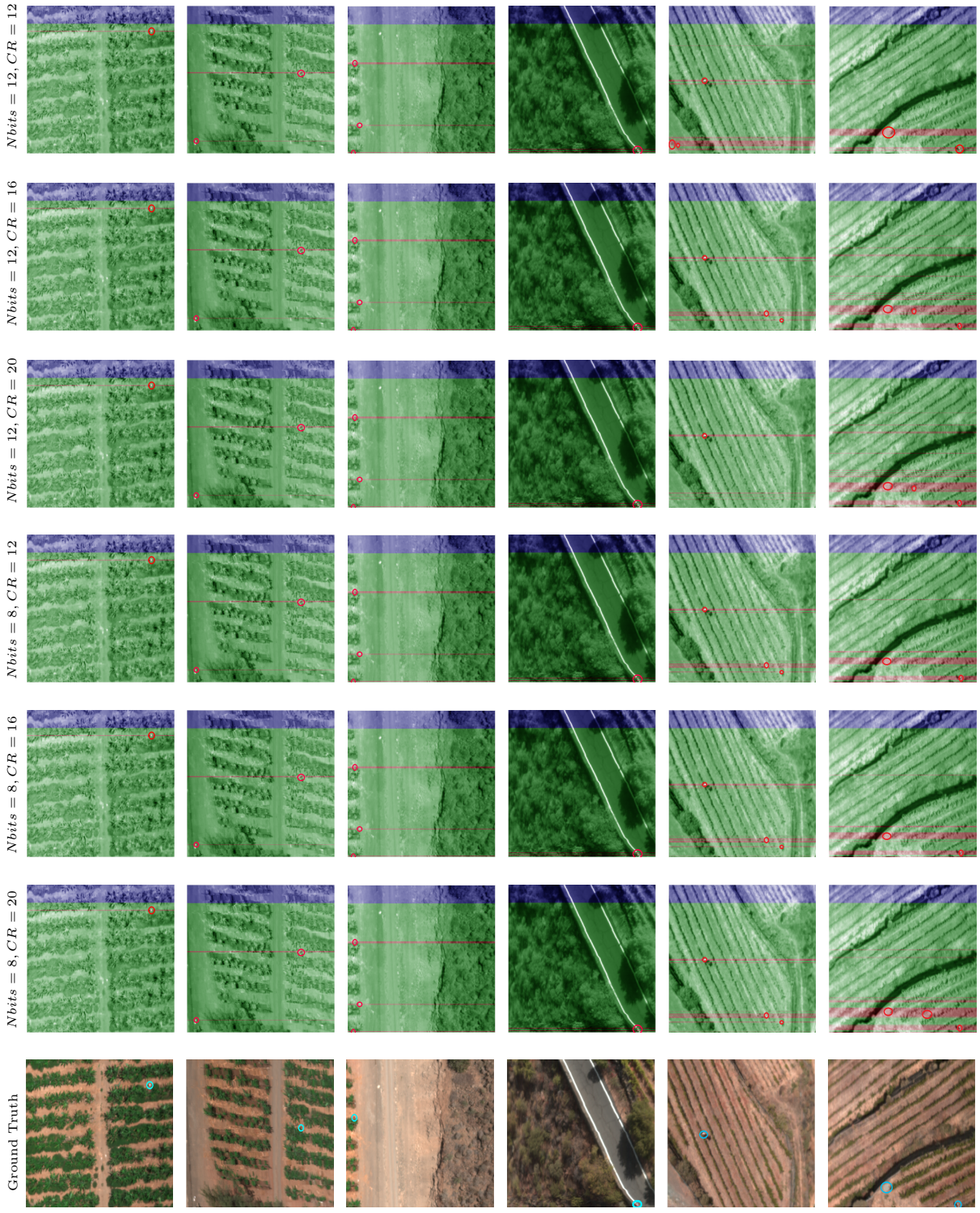


FIGURE 4.12: Anomaly detection results obtained by the LbL-FAD algorithm using the compressed images obtained by the HyperLCA algorithm for $BS = 512$.

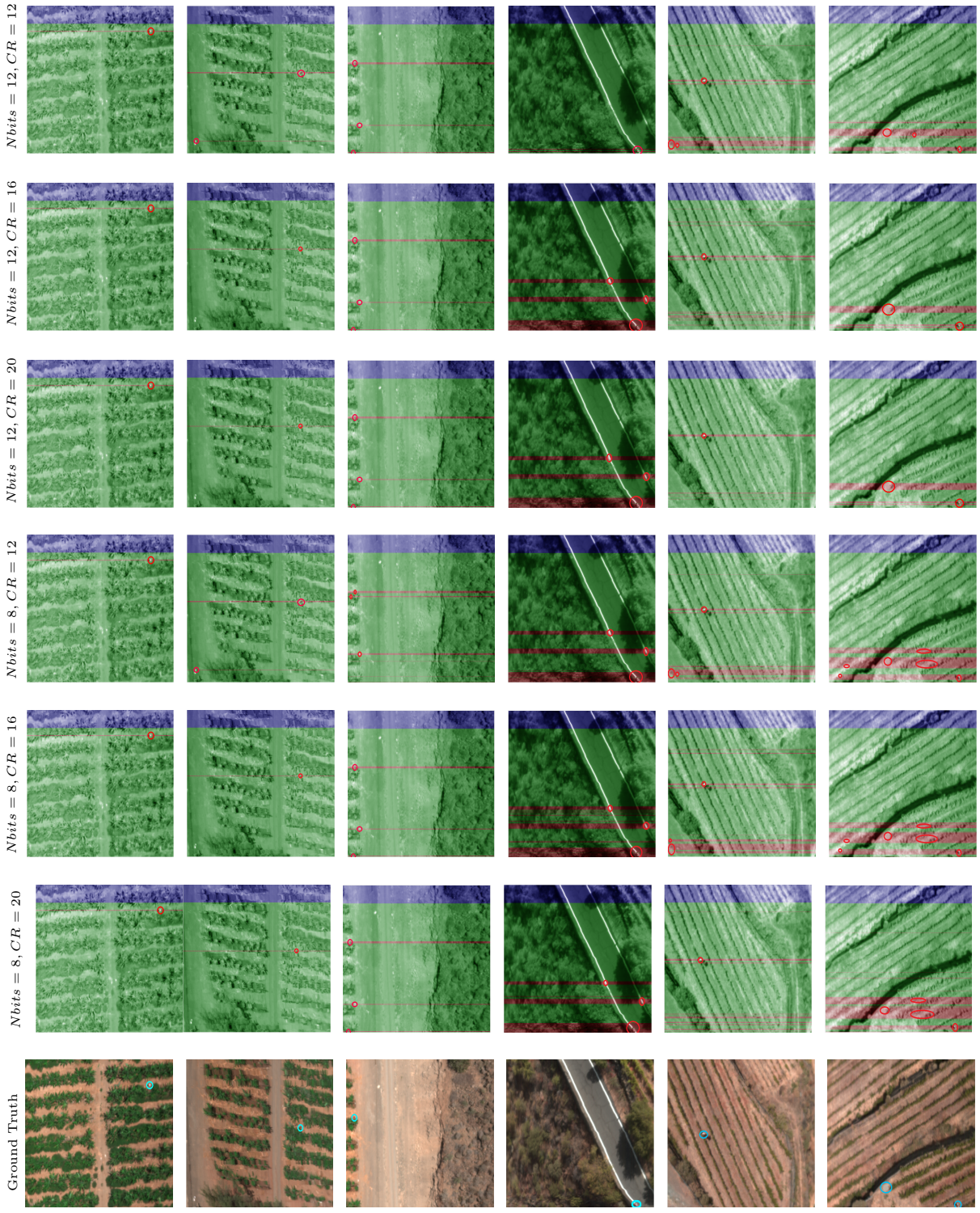


FIGURE 4.13: Anomaly detection results obtained by the LbL-FAD algorithm using the compressed images obtained by the HyperLCA algorithm for $BS = 256$.

of BS hyperspectral pixels. Since the number of OPs required by the *Initialization* and the *Preprocessing* stages are negligible, it has not been included in Table 4.5. Moreover, the possibility of performing an additional stopping condition based on quality metrics has not been retained in this analysis.

Stage	Number of operations (OPs)
HyperLCA Transform	$p_{\max} \cdot (6 \cdot N_b \cdot BS + N_b) + 2 \cdot N_b \cdot BS + N_b$
HyperLCA Entropy Coding	$5 \cdot (p_{\max}+1) \cdot N_b + 5 \cdot p_{\max} \cdot BS + 17 \cdot (2 \cdot p_{\max}+1)$

TABLE 4.5: Number of OPs required by the *HyperLCA Trasnform* and the *Entropy Coding* stages to process a single block of BS hyperspectral pixels

On the basis that p is much smaller than BS and nb, it can be easily argued that the computation of the *HyperLCA Transform* involves much more operations than the *Entropy Coding* stage. For the sake of clarity, Table 4.6 collects some numeric examples using the test bench described in Section 4.4.1 for different configurations of the HyperLCA input parameters. In this sense, Table 4.6 displays the average number of OPs required to perform both stages for a hyperspectral image block with BS elements, $\mathbf{M}_k$. From the obtained results, it is right to conclude that the *HyperLCA Transform* is the most computationally intensive part of the HyperLCA compressor, executing up to three orders of magnitude more OPs than the *HyperLCA Entropy Coding*.

Nbits	BS	CR	Number of operations	
			HyperLCA Transform	*Entropy Coding*
12	1024	12	1,11E+07	6,63E+04
		16	8,19E+06	4,84E+04
		20	6,23E+06	3,65E+04
	512	12	4,59E+06	3,14E+04
		16	3,61E+06	2,46E+04
		20	2,62E+06	1,78E+04
	256	12	1,80E+06	1,56E+04
		16	1,31E+06	1,14E+04
		20	1,07E+06	9,27E+03
8	1024	12	1,61E+07	9,61E+04
		16	1,11E+07	6,63E+04
		20	9,18E+06	5,44E+04
	512	12	6,56E+06	4,49E+04
		16	4,59E+06	3,14E+04
		20	3,61E+06	2,46E+04
	256	12	2,30E+06	1,98E+04
		16	1,80E+06	1,56E+04
		20	1,31E+06	1,14E+04

TABLE 4.6: Evaluation of the number of OPs required by the *HyperLCA Transform* and the *HyperLCA Entropy Coding* to process an image block of BS pixels, $\mathbf{M}_k$.

4.5 Conclusions

In this Chapter, the novel state-of-the-art hyperspectral imagery lossy compressor, named HyperLCA, has been described. The HyperLCA compressor is a transform-based algorithm that performs an spectral transform for the decorrelation and reduction of the hyperspectral data. To do so, the HyperLCA algorithm projects the data onto a set of orthogonal vectors, similarly to other state-of-the-art approaches that employ the well-known KLT transform. In this Thesis work, some performance-enhancing improvements have been made in the first release of the HyperLCA compressor in order to use the set of core operations extensively analysed in this manuscript during the transformation process of the input data.

Unlike other state-of-the-art transform-based compressors, the HyperLCA was introduced as a low-complexity alternative that provides a good compression performance at high compression ratios with a reasonable computational burden. Additionally, this algorithm permits compressing blocks of image pixels independently that promotes, on the one hand, the reduction of the data to be managed at once besides the hardware resources to be allocated. On the other hand, the HyperLCA algorithm becomes a very competitive solution for most applications based on pushbroom/whiskbroom scanners, paving the way a for real-time compression performance.

Most of the compression performance achieved by the HyperLCA algorithm is obtained in the *HyperLCA Transform* stage, originally designed to use floating-point precision. Nonetheless, some parallel dedicated hardware devices, such as FPGAs, are in general more efficient dealing with integer operations. For this reason, three alternative versions of the *HyperLCA Transform* described with integer arithmetic at different precision levels have been proposed in this Thesis work. Concretely, these strategies employ the fixed-point notation and they are referred to as *Int32, Int16* and *Int16-rd*, respectively. The *Int32* and *Int16* versions were developed for working with HSIs with up to 16 bits-depth while *Int16-rd* version is though for images with up to 12 bits-depth. From experiments made, it has been confirmed that the *Int32* and the *Int16-rd* versions lead to very similar results that the *Float32* version, but being more suitable for those hardware devices that are more suitable for working with integers. In the particular cases of the *Int16-rd* and *Int16* versions, they also halves the memory space required for storing the input data.

Several simulations have been carried out using real hyperspectral data sets to evaluate the HyperLCA performance for spectrally decorrelating and reducing HSIs. To this end,

different statistical assessment metrics have been used, such as the *SNR, RMSE, MAD* and *SSIM*. Regarding the computational burden, it has also been evaluated in terms of the number of operations to be executed. Finally, the effects of the missing information introduced by the lossy compression performed by the HyperLCA have been tested on anomaly detection application. For this end, anomaly detection has been conducted on the decompressed images using the LbL-FAD algorithm proposed in this Thesis work. Based on the obtained results, it can be drawn that the HyperLCA algorithm is able to preserve rare non-dominant spectra, what is crucial for many hyperspectral applications such as anomaly detection, tracking or classification.

To sum up, the methodology proposed in this Thesis work introduces the following advantages:

1. High compression ratios and competitive rate-distortion compression performance.

   HSIs resulting from the compression process performed by the HyperLCA algorithm have been analysed in terms of the *SNR, RMSE, MAD* and *SSIM* assessment metrics for different configurations of the algorithm input parameters. The average values for these quality metrics are $SNR = 38.26$, $RMSE = 14.33$, $MAD = 4\%$ of the total and $SSIM = 0.98$ for compression ratios ranging from 16.3 to 36.

2. Preservation of the most atypical spectra.

   Despite being a lossy compression approach, the HyperLCA algorithm is able to preserve the most different HSIs within a data set after the compression-decompression process, which is crucial for several applications, such as anomaly detection, classification, unmixing, or target detection [69, 185–188]. It is because the *HyperLCA Transform* extracts the projection vectors that best represent the most different pixels within the data set.

3. Line-by-Line performance.

   The HyperLCA algorithm was specially designed for independently processed blocks of image pixels with no spatial strings attached among them. On the one hand, this feature prevents the accumulation of high amount of uncompressed data onboard. On the other hand, the data volume to be managed and transferred at a time is reduced. Consequently, the proposed methodology becomes an ideal solution for applications under tight latency constraints or with limited available resources, such as memory, power and computational capabilities. Furthermore, most of nowadays

remote sensing applications employ pushbroom/whiskbroom scanners [31, 49–51], which collect the HSIs in a line-by-line fashion. In this regard, the HyperLCA compressor is also oriented to face the requirements imposed by these applications since sensed hyperspectral frames may be compressed on-the-fly.

4. Low computational complexity and speed-up due to parallelization.

   As it was analysed along this Chapter, the *HyperLCA Transform* bears most of the complexity and computational burden of the compression process. Similarly to the KLT transform-based approaches, and in particular to the PCA transform, the HyperLCA algorithm extracts a set of orthonormal vectors using the set of core operations proposed in this Thesis work, which are then used for projecting the data into a new orthogonal subspace. Unlike these state-of-the-art solutions, the HyperLCA algorithm provides a reduced computational burden and implementation complexity with a high grade of parallelism and scalability while also maintaining most of the benefits provided by the aforementioned approaches. The flexibility and the high level of parallelism intrinsic to the HyperLCA algorithm have been evaluated in earlier publications, such as [215–217], and will be further assessed in Chapter 6 of this document.

5. Reduction in the required hardware resources.

   The capacity of independently compressing blocks of image pixels develops into the advantage of keeping in memory just small portions of the data and thus, reducing the hardware resources for this purpose. Additionally, less data have to be handled at once, which do positively affect on this.

6. Different data types and precision.

   The performance analysis carried out in this Thesis work demonstrates that the *HyperLCA Transform* can be efficiently executed using floating-point and integer arithmetic. For the latter, it was used in particular the fixed-point concept in a custom way employing integer arithmetic and bits shifting. Three versions of the *HyperLCA Transform* were considered employing 32, 16 and 12 bits, respectively. Additionally, the methodology followed in this Thesis work can be used for developing any other integer version of the compressor that uses a different precision for optimizing it to the necessities of the targeted applications.

Chapter 5

Concurrent Execution of Multiple Hyperspectral Imaging Applications

The onboard processing of remotely sensed hyperspectral images for on-the-fly making-decision applications has gained momentum in recent years. Nonetheless, the adoption of this operation mode brings with it many huge challenges to be faced in the near future, mainly related with the increase of the sensor data rates and the limited amount of available onboard computational resources. Indeed, this situation becomes even more complex and cumbersome when different time-sensitive applications coexist, since different tasks must be sequentially processed onto the same computing device. As a contribution to this field of interest, we come up with a strategy based on the reuse of the set of core operations proposed in this Thesis work for the execution of several processing techniques. In particular, we analyse the potential of the suggested methodology towards the concurrent execution of multiple hyperspectral analysis processes, whilst optimizing the computational resources and the human endeavours invested during the implementation stage. In concrete, we focus on the issue behind the proposed HW-LbL-FAD algorithm and the HyperLCA method for the simultaneous detection of anomalous spectra and the lossy compression of hyperspectral images.

5.1 Rationale

The onboard processing of remotely sensed hyperspectral images (HSIs) has experienced a steady surge in popularity in recent decades. Indeed, it represents a potential solution for those applications that demand quick response in which the conventional approaches based on the off-line data handling do not meet the actual needs. Unfortunately, the data transmission from the remote sensing platforms to the Earth surface introduces important delays related to the communication of large amount of data. Consequently, this fact acts as a barrier that can seriously reduce the effectiveness of real-time applications or applications that demand prompt replies [36, 37]. In this contest, the real-time onboard processing has become a very interesting solution to this type of stringent applications [37–40]. Nonetheless, the adoption of this operation mode is not an easy matter and there are still many challenges ahead, due mainly to the high data rate of the new-generation hyperspectral sensors and the limited amount of available onboard computational resources.

In addition, the aforementioned scenario becomes even more challenging when different time-sensitive applications coexist. It is because different tasks must be sequentially processed onto the same computing device. Regrettably, the algorithms traditionally proposed for the hyperspectral analysis have been addressed as independent entities, using those mathematical methods that better maximize the results for each particular case. In addition, these approaches normally give rise to complex algorithms with many data dependencies. This is because the algorithm development phase is usually detached from the hardware implementation stage, resulting in very inefficient hardware implementations. All of this makes the onboard execution of multiple processes for analyzing the acquired hyperspectral data not fully viable, especially under real-time constraints [38, 48]. In view of the computationally intensive nature of the state-of-the-art hyperspectral imaging algorithms, new hardware-friendly solutions are required enabling real-time execution based on an appropriate trade-off among design requirements [38].

Against this backdrop, we have dealt in this Thesis work with the issue around the onboard execution of multiple hyperspectral image analysis techniques onto the same piece of hardware. To this end, we come up with a strategy based on the reuse of the proposed set of core operations, focusing the efforts in the interconnections among them for the targeted applications to be addressed. Consequently, the greatest strength of our proposal is the ability to concurrently execute multiple hyperspectral analysis processes, whilst optimizing the computational resources and the human endeavours invested during the

implementation stage. It is because the targeted imaging applications are based on the same mathematical methods, in particular, the modified version of the Gram–Schmidt orthogonalization method performed by the set of core operations proposed in this Thesis work.

In order to demonstrate the viability of the defended methodology, we have focused on the lossy compression of hyperspectral images and the detection of anomalous spectra, in particular in the Hardware-Friendly Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery (HW-LbL-FAD) and, in the Lossy Compression Algorithm for Hyperspectral Image Systems (HyperLCA), extensively analysed in previous Chapters 3 and 4. Nevertheless, we would like to mention that this methodology could be potentially extended to include other processes without a relevant increment of the required computational resources, as it will be further analysed in Appendix A.

5.2 Towards the Concurrent Execution of Multiple Hyperspectral Imaging Applications

In this Section, we discuss the suitability of the proposed set of core operations for the concurrent execution of multiple hyperspectral imaging applications, in particular, the lossy compression of HSIs and the detection of anomalous spectra. To this end, we analyse different methodological strategies ranging from the independent execution of the targeted applications to the definition of two highly optimized versions for the joint implementation of both processes using a single configuration of the hardware utilisation.

5.2.1 Using the proposed set of core operations for the detection of anomalous spectra OR for the lossy compression of HSIs

In preceding Chapters, we introduced the feasibility of the proposed set of core operations for the detection of anomalous spectra and the lossy compression of HSIs. To this end, two algorithms were extensively analysed, in particular, the HW-LbL-FAD and the HyperLCA algorithms. Nonetheless, these two issues were evaluated as two independent entities. In this Section, we go a step further in order to prove that the proposed methodology can be efficiently employed for the execution of any such above mentioned analysis techniques using a single configuration of the proposed set of core operations. On the one hand, it

means less cost, time and human endeavours during the implementation phase of each targeted hardware solutions. In this sense, designed blocks of logic or data (intellectual property (IP) cores) in making FPGA-based solutions or routines compiled for GPUs (kernels) can be shared among the processing techniques to be launched. On the other hand, the proposed methodology permits the execution of many different tasks with the advantage of sharing the most computationally costly core operations, thus reducing the overall computational cost and the required hardware resources. Although these matters will be further analysed in Chapter 6, in this Section we focus on the description of the methodology behind this statement.

For the sake of clarity, a short description about the algorithmic stages involved in the HW-LbL-FAD and the HyperLCA methods is also presented in the following lines. This overview is supported by the schematic diagram displayed in Figure 5.1, where the proposed methodology for the execution of the set of core operations applied to anomaly detection or lossy compression of HSIs is depicted. A closer view of the aforementioned graphical representation is collected in Figure 5.2 for each targeted application.

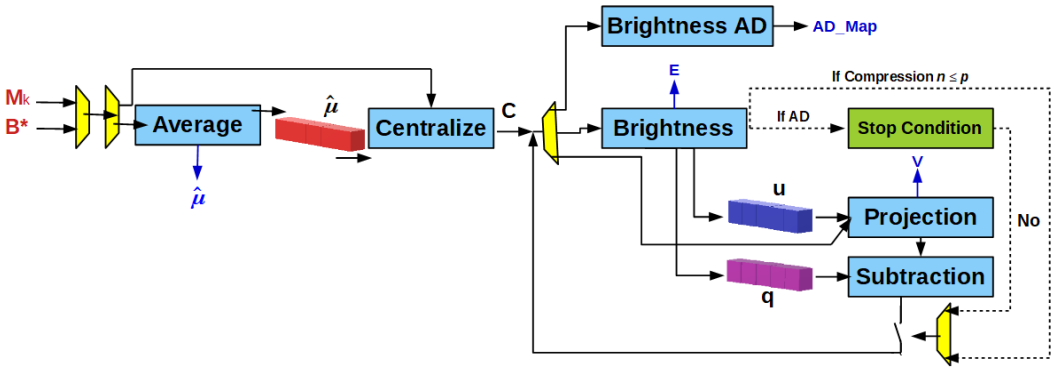


FIGURE 5.1: Proposed scheme for the execution of the set of core operations applied to anomaly detection or lossy compression of HSIs.

In general terms, the HyperLCA algorithm is mainly structured in four stages: *Initialization*, *HyperLCA Transform*, *Preprocessing* and *Entropy Coding*. Nevertheless, the *HyperLCA Transform* involves the most computationally demanding operations, besides, it actually executes the proposed set of core operations. For this reason, we focus on this stage of the HyperLCA algorithm through the rest of the analysis made in this Chapter. Overall, the flow of operations carried out by the *HyperLCA Transform* is highlighted in pink color in Figure 5.2a and listed below:

1. Each time that a new block of image pixels, $\mathbf{M}_k$, is available, it feeds the set of core operations in order to extract the p most characteristic pixels, $\mathbf{E}$, and their

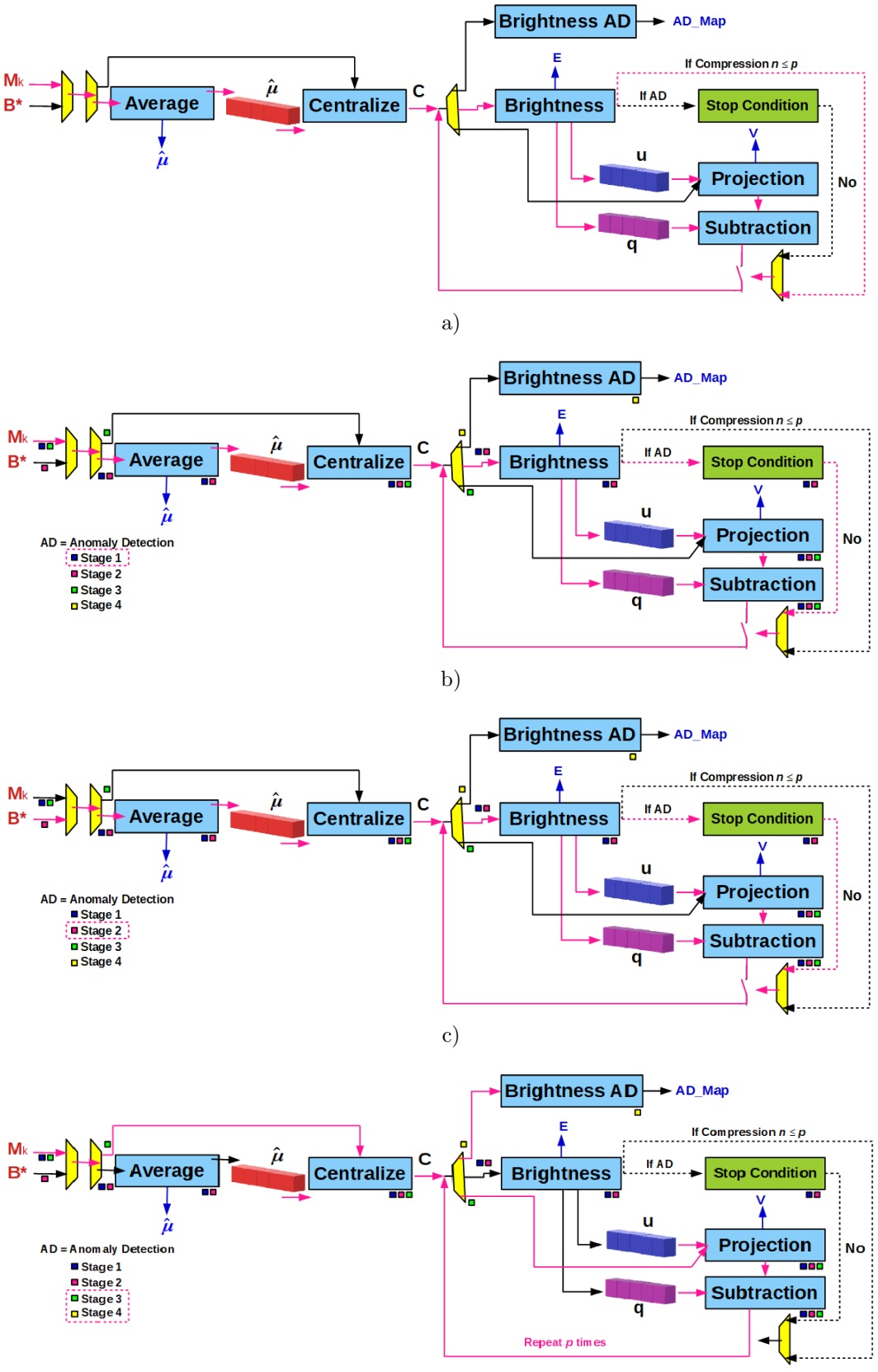


FIGURE 5.2: Detailed view of the schematic diagram displayed in Figure 5.1. (a) Lossy Compression. (b) Anomaly Detection, Stage 1. (c) Anomaly Detection, Stage 2. (d) Anomaly Detection, Stages 3 and 4.

corresponding projection vectors, $\mathbf{V}$. Parameter p is set at the very beginning in the previous *Initialization* stage according to some input parameters.

2. Then, the average pixel of spectra within $\mathbf{M}_k$, $\hat{\boldsymbol{\mu}}$, is estimated (*Average* block in Figure 5.2a) and used to centralize the input image in the *Centralization* step, resulting in the centralized version of the image, $\mathbf{C}$.

   The following operations are repeated p times:

3. The pixels selected as the most characteristic within each image block, $\mathbf{E}$, are those with the highest l^2-*norm*, also referred to as brightness in this Thesis work. Derived from it, orthogonal vectors, $\mathbf{q}_n$ and $\mathbf{u}_n$, are also estimated in order to be used in the following algorithm stages, as it can be seen from rows getting out from *Brightness* block in Figure 5.2a.

4. Subsequently, the projection vector, $\mathbf{v}_n$, is computed as the spectral information of each pixel within $\mathbf{C}$ spanned by $\mathbf{u}_n$ (*Projection* block in Figure 5.2a).

5. Finally, the data from $\mathbf{C}$ that is orthogonal to the already selected $\mathbf{e}_n$ pixels are retained in the *Subtraction* step for the next iteration, n. In this sense, $\mathbf{C}$ matrix is updated if $n \leq p$, as it can be seen from the multiplexor located in the bottom right side of Figure 5.2a, which activates the corresponding switch for this condition.

The HW-LbL-FAD algorithm entirely employs the proposed set of core operations but their proper execution order depends on the four different defined algorithm stages. It means that the major complexity of this process lies in the interconnections among operations in relation to the algorithm stage to be launched. Overall, the flow of operations carried out by the HW-LbL-FAD algorithm for its different computing stages is displayed from Figure 5.2b to Figure 5.2d and listed below:

1. *Stage 1*: In this stage, the first n_f image blocks, $\mathbf{M}_k$, are independently analysed in search of the p most different pixels, $\mathbf{E}$, within them. In this sense, the set of core operations is executed in the same way as by the *HyperLCA Transform*, as it can be noticed from the comparison between Figures 5.2a and 5.2b. The major difference is the estimation of the number of iterations, n, to be carried out for extracting the p reference vectors. Regarding the HyperLCA compressor, p is known beforehand since it is computed in the *Initialization* stage as a function of some input parameters. On the contrary, a stop condition is checked in every

iteration in the case of the HW-LbL-FAD algorithm. For this reason, an extra green block is included in the schematic diagram shown in Figures 5.1 and 5.2, named *Stop Condition*, for representing this additional stopping condition. Therefore, the multiplexor placed in the bottom right side of Figures 5.1 and 5.2 is in charge of activating the corresponding switch that updates the $\mathbf{C}$ values if $n \leq p$, when the compression process is enabled, or if the stopping condition is not fulfilled in the case of anomaly detection.

2. *Stage 2*: In this stage, the subspace of orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$, that model the background distribution is estimated using the set of core operations organized in the same way as it was defined in *Stage 1* (see Figure 5.2c). Nonetheless, the set of vectors, $\mathbf{E}$, above selected is employed as input pixel block, $\mathbf{B^*}$, instead of $\mathbf{M}_k$. For this reason, it is included the first multiplexor placed in the left side of Figures 5.1 and 5.2 that selects the data to be used according to the algorithm stage.

3. *Stages 3 and 4*: Once the background pattern is modelled, the next HW-LbL-FAD stages deal with the detection of the anomalous pixels within the following sensed HSI blocks, $\mathbf{M}_k$. These stages slightly differ from the previous two, as it can be seen from Figure 5.2d. For doing so, the average pixel, $\hat{\boldsymbol{\mu}}$, estimated in the *Stage 2* is used for centralizing the input image block, $\mathbf{M}_k$. Consequently, the *Average* blue block must be skipped in this third Stage and thus, it requires the use of the second multiplexor located in the left side of Figures 5.1 and 5.2. Subsequently, the information spanned by the $\mathbf{Q}$ and $\mathbf{U}$ vectors computed in *Stage 2* is subtracted from pixels within $\mathbf{M}_k$. To do this, the *Projection* and *Subtraction* operation blocks are repeated p times, that is, the number of vectors contained in $\mathbf{Q}$ or $\mathbf{U}$. As it can be noticed, the *Brightness* operation block is not required in the actual stage and therefore, it is defined the multiplexor located in the middle of Figures 5.1 and 5.2. Finally, the brightness of the remaining spectral information in each image pixel is used for identifying the anomalous entities in the *Stage 4*. For this reason, it is defined an additional operation block, named *Brightness AD*, though it basically performs the l^2-*norm* of each image pixel as the *Brightness* operation block.

To sum up, it can be drawn that the proposed set of core operations can be potentially used to perform the different algorithm stages involved by the HW-LbL-FAD detector or the spectral transform carried out by the HyperLCA algorithm using a single configuration of the aforementioned operations for both processing techniques. Consequently, this will

result in a marked reduction in the hardware resources and human endeavours if it is pretended to launch both processes independently in the same piece of hardware.

5.2.2 Using the proposed set of core operations for the concurrent execution of the anomaly detection issue AND the lossy compression of HSIs

As discussed in preceding Section, the lossy compression of HSIs as well as the detection of anomalous pixels can be addressed using the same mathematical method and, in particular, by means of the set of core operations proposed in this Thesis work. In this Section, we go a step further and explore the potential of the suggested methodology for the joint implementation of both targeted applications.

5.2.2.1 First approximation towards the simultaneous detection of anomalous pixels and the lossy compression of HSIs

In the field of hyperspectral lossy compression, the most different pixels, $\mathbf{E}$, and their corresponding projection vectors, $\mathbf{V}$, within an image block, $\mathbf{M}_k$, can be potentially used to decorrelate the HSIs and thereby, to compress them. In the field of anomaly detection, the most characteristic pixels, $\mathbf{E}$, and their orthogonal counterparts, $\mathbf{Q}$ and $\mathbf{U}$, can be efficiently employed for the estimation of the orthogonal subspace spanned by the background pattern in which anomalous spectra are easily detectable. On this basis, the first step to be followed consists in extracting the most representative pixels within each image block, $\mathbf{M}_k$.

To this end, the set of core operations is executed in the same way as the HyperLCA and the HW-LbL-FAD methods in its *Stage 1*, as it was further analysed in Section 5.2.1. The main difference lies in the estimation of the number of iterations, n, to be carried out for extracting the p reference vectors needed by each targeted application. In the following, p_c and p_{AD} represent the number of p pixels selected by the HyperLCA algorithm and the HW-LbL-FAD detector, respectively. Regarding the HyperLCA compressor, p_c is known beforehand since it is computed in the *Initialization* stage as a function of some input parameters. On the contrary, a stop condition is checked in every iteration in the case of the HW-LbL-FAD algorithm. On this basis, the set of core operations might be executed just once and its outputs could be reused by both targeted applications. In this sense,

the number of p iterations to be carried out by the set of core operations is determined by the requirements imposed by both p_c and p_{AD}, in such a way that if $p_{AD} < p_c$, then $\mathbf{E_{AD}}$ is a subset of $\mathbf{E_c}$. Otherwise, $\mathbf{E_c}$ is a subset of $\mathbf{E_{AD}}$.

The process described in the above lines is done on the first n_f images blocks, $\mathbf{M}_k$. Nonetheless, in order to estimate the subspace spanned by the background distribution required for the detection of anomalous pixels, the *Stage 2* of the HW-LbL-FAD algorithm has to be also computed. For doing so, the set of core operations may be applied once again in background, playing $\mathbf{B}^* = \mathbf{E}_k$ $(k \leq n_f)$ as the input matrix $\mathbf{M}_k$. As a result, they are obtained the average pixel of the background distribution, $\hat{\boldsymbol{\mu}}$, and the orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$, which will be later employed to identify the anomalous spectra.

So far, the outputs of the proposed set of core operations could be reused for both targeted applications enabling their simultaneous performance. Nonetheless, the issue is further complicated from this point on. Regarding the anomaly detection issue, once the background pattern is modelled in the *Stage 2* of the HW-LbL-FAD algorithm, the detection of the anomalous spectra is addressed on the new sensed HSI blocks, $\mathbf{M}_k$, in the *Stage 3* of the algorithm. For doing so, the average pixel, $\hat{\boldsymbol{\mu}}$, estimated in the *Stage 2* is used for centralizing the input image block, $\mathbf{M}_k$, and hence, the operations involved in the average pixel calculation are indeed discarded. Subsequently, the information spanned by the $\mathbf{Q}$ and $\mathbf{U}$ vectors computed in the *Stage 2* is subtracted from pixels within $\mathbf{M}_k$. To do this, the projection and subtraction operations are repeated p_{AD} times, that is, the number of vectors contained in $\mathbf{Q}$ or $\mathbf{U}$. As it can be noticed, the brightness computation is not required in the actual stage since characteristic pixels were selected in the previous *Stage 2*. For the sake of clarity, we encourage the reader to see Figure 5.2d. Nonetheless, the methodology followed by the HyperLCA algorithm for compressing the new sensed HSI blocks, $\mathbf{M}_k$, remains the same as it was defined at the beginning. Therefore, it implies the centralization of the data according to the average pixel of the current image block, $\hat{\boldsymbol{\mu}}$, the computation of the pixel brightness in search of the most characteristic pixels within $\mathbf{M}_k$, $\mathbf{E}$, and the execution of the projection and subtraction operations employing the already selected pixels, $\mathbf{e}_n$, from the image block in question.

Having regard to the above, it can be concluded that, though the computing operations performed by the HyperLCA and the HW-LbL-FAD methods are those defined within the proposed set of core operations, different data are processed by them for each case. Consequently, if both targeted applications are desired to be simultaneously executed in

the same piece of hardware, the *Centralization*, the *Projection* and the *Subtraction* operation blocks displayed in Figures 5.1 and 5.2 must be replicated. Naturally, it negatively affects the resource utilisation because they have to be duplicated for indeed performing the same operations. An alternative solution for power-constrained scenarios with limited hardware resources would focus on the serial execution of the targeted hyperspectral analysis techniques for preventing the replication of the aforementioned core operations. However, it would clearly concern the execution times.

Against this backdrop, the versatility inherent to the proposed set of core operations leads to two optimized proposals for the simultaneous detection of anomalous pixels and the lossy compression of HSIs. The first one, referred to as ADeLoC, searches for the highest accuracy in the detection and compression results, whereas the other, namely HADeLoC, prioritizes the optimization of the hardware resources and the minimization of the execution times.

5.2.2.2 Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs (ADeLoC)

As it was previously mentioned, the ADeLoC version focuses on the preservation of the accuracy of the results for the two targeted applications, that is, anomaly detection and lossy compression. For this reason, this alternative ensures the same results as the original HW-LbL-FAD and the HyperLCA methods [218] in exchange of some minor changes in the way as *Stages 3* and *4* of the HW-LbL-FAD method are performed. In return, this avoids the replication of the *Centralization*, the *Projection* and the *Subtraction* operation blocks and reduces the number of operations to be performed compared with the context of the issue raised in previous Section 5.2.2.1.

In this regard, although the p_{AD} background reference vectors were extracted from the previous n_f image blocks, $\mathbf{M}_k$, in the *Stage 2* of the HW-LbL-FAD algorithm, the whole package of the proposed set of core operations has to be run on the new received hyperspectral frames to extract the p_c most characteristic pixels, $\mathbf{E_c}$, needed for the compression process. Nonetheless, these $\mathbf{E_c}$ vectors are actually the most different pixels within $\mathbf{M}_k$, and hence, they also collect the rarest signatures too. For this reason, if any anomalous pixel is present in $\mathbf{M}_k$, it must be collected in $\mathbf{E_c}$. Therefore, the ADeLoC approach only projects these $\mathbf{E_c}$ vectors onto the orthogonal subspace to the one spanned by the $\mathbf{Q}$ and $\mathbf{U}$ vectors estimated in the *Stage 2* of the HW-LbL-FAD algorithm. In case of presence of

any anomalous pixel within $\mathbf{E_c}$, the entire image block, $\mathbf{M}_k$, is processed in order to also detect mixed anomalous pixels. Otherwise, only $\mathbf{E_c}$ vectors are checked, thus reducing the number of operations to be performed.

5.2.2.3 Hardware-friendly proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs (HADeLoC)

As already concluded in the previous Section 5.2.2.1, the proposed set of core operations may be perfectly applicable for the simultaneous execution of both anomaly detection and lossy compression of HSIs. Nonetheless, in the way that the methodology was defined in previous Section 5.2.2.1, some operations must be replicated due to the different data handled by the two targeted applications, which obviously negatively affects the hardware utilisation. On this basis, we present a more resource-optimized solution in this Section that, although is not a faithful image of the HW-LbL-FAD and the HyperLCA algorithms, but it is based on the same rationale behind the methodology followed by them. It will be referred to as HADeLoC along the remainder of this Section. Figure 5.3 displays a schema of the operations carried out by this new proposal for the parallel execution of anomaly detection and lossy compression. Additionally, a closer view of the aforementioned graphical representation is collected in Figure 5.4 for each algorithmic stage, which are further explained in the following lines.

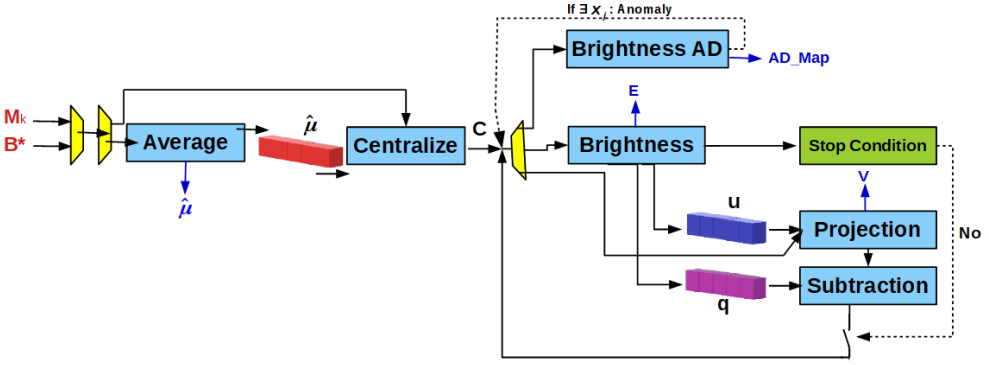


FIGURE 5.3: Proposed methodology for the simultaneous performance of the anomaly detection process and the lossy compression of HSIs.

1. *Stage 1*: This stage corresponds with the *Stage 1* of the HW-LbL-FAD algorithm. As a reminder, the set of core operations is independently performed on each of the first n_f image blocks, $\mathbf{M}_k$, for extracting the set of the most characteristic pixels, $\mathbf{E}$, within them. Indeed, this stage also corresponds with the spectral transformation,

named *HyperLCA Transform*, conducted by the HyperLCA compressor. The main difference between both algorithms lies in how the number of pixels to be selected, p, is determined. In the case of the HyperLCA algorithm, it is previously estimated according to same input parameters, in particular, BS, N_{bits} and CR, in the *Initialization* stage. For the HW-LbL-FAD algorithm, an stopping condition based on the ratio between the brightness of the selected pixel in $\mathbf{M}_k$ and $\mathbf{C}$ is checked in every iteration. In the actual proposal, the stopping condition inherent to the HW-LbL-FAD algorithm is maintained. The selected $\mathbf{E}$ pixels are saved for the subsequent algorithm stages and, jointly with their corresponding projection vectors, $\mathbf{V}$, and the average pixel of the image block, $\hat{\boldsymbol{\mu}}$, are packaged to be send to the ground segment as the compressed data. For the sake of clarity, the work-flow of the operations involved in this first stage of the current approach is shown in Figure 5.4a.

2. *Stage 2*: This stage has also a correspondence with the *Stage 2* of the HW-LbL-FAD algorithm. Although in this point no image block, $\mathbf{M}_k$, is processed, this stage is indeed a key point since the background distribution is modelled through the definition of its average pixel, $\hat{\boldsymbol{\mu}}$, and a set of orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$. These vectors will be later used in subsequent stages for the detection of anomalous spectra and the compression of the hyperspectral frames. In general terms, it executes the same operations as above *Stage 1* but the input image block is in turn the set of pixels, $\mathbf{B}^* = \mathbf{E}_k$ $(k \leq n_f)$, selected in this previous stage. Since the data currently handled is not part of the image to be compressed and reconstructed on the Earth surface, only the selected pixels, $\mathbf{E}$, and the average pixel, $\hat{\boldsymbol{\mu}}$, will be added to the bitstream. These pixels are essential for decompressing the subsequent image blocks, $\mathbf{M}_k$, as it will be more clear in the description of the next algorithm stages. A closer view of the flow of operations conducted in this second stage is displayed in Figure 5.4b.

3. *Stages 3 and 4*: The detection of the anomalous spectra takes place in these two stages. This process is conducted in the new received image blocks, $\mathbf{M}_k$, in a similar way to what it is done in the HW-LbL-FAD algorithm. On the contrary, the compression process changes considerably compared with the original HyperLCA compressor in the interest of preventing the replication of the *Centralization*, the *Projection* and the *Subtraction* operations, as it was analysed in Section 5.2.2.1. In the original version of the HyperLCA compressor, it would be proceeded to extract the most characteristic pixels within the image block under analysis, $\mathbf{M}_k$, as those with the highest brightness within matrix $\mathbf{C}$ in each iteration. On the basis that

vectors **Q** and **U**, which were estimated in previous *Stage 2*, model the background pattern, they should be representative enough of the majority of the spectral information contained in each pixel within $\mathbf{M}_k$. In addition, the aforementioned vectors are employed by the HW-LbL-FAD algorithm as a vector space basis for projecting the image pixels onto them in order to just retain the orthogonal information. In this sense, the orthogonal subspace to the one spanned by the background distribution modelled by **Q** and **U** vectors represents better the anomalous spectra. For this reason, an anomalous pixel would be identified by a notably high value of the l^2-*norm*, also named brightness, after the subtraction of the spectral information spanned by the background pattern. For doing so, the *Projection* and the *Subtraction* operations are repeated as many times as the number of pixels selected in the *Stage 2*, p. Regarding the compression process, a projection vector, $\mathbf{v}_n$, is obtained in each iteration, n, and it may be used for off-line reconstructing the data. In this sense, **Q** and **U** vectors are also needed for this purpose but they can be estimated from **E** vectors transmitted in *Stage 2*. For this reason, these two set of vectors are not required to be part of the bitstream in this algorithm stage. For the sake of clarity, the work-flow of the operations involved in this algorithm stage of the current approach are shown in Figure 5.4c.

4. *Stage 5*: Unlike the HW-LbL-FAD algorithm, this approach involves an additional computing stage. It is because the anomalous pixels detected in the above stage could not be well reconstructed using the transmitted **V** vectors since they just retain the spectral information representative of the background pattern. For this reason, when anomalous spectra are identified, the group of operations *Brightness - Projection - Subtraction - Brightness AD* is repeated until *Brightness AD* does not extract any more anomalous pixels. Therefore, apart from **V** vectors estimated in *Stage 3*, the new selected pixels with the highest brightness, $\mathbf{e}_n$, and its corresponding projection vectors, $\mathbf{v}_n$, are also included in the bitstream to be transmitted for each image block, $\mathbf{M}_k$. It is clear that this stage is not conducted if no anomalous pixels are detected at the very beginning. A closer view of the flow of operations involved in this last stage is displayed in Figure 5.4d.

It is important to mention that the Golomb-Rice encoding conducted by the HyperLCA compressor is performed as well for each single outcome, $\mathbf{e}_n$, $\mathbf{v}_n$ or $\hat{\boldsymbol{\mu}}$, resulting from each of the aforementioned stages. Therefore, the *Error Mapping* step within the *HyperLCA*

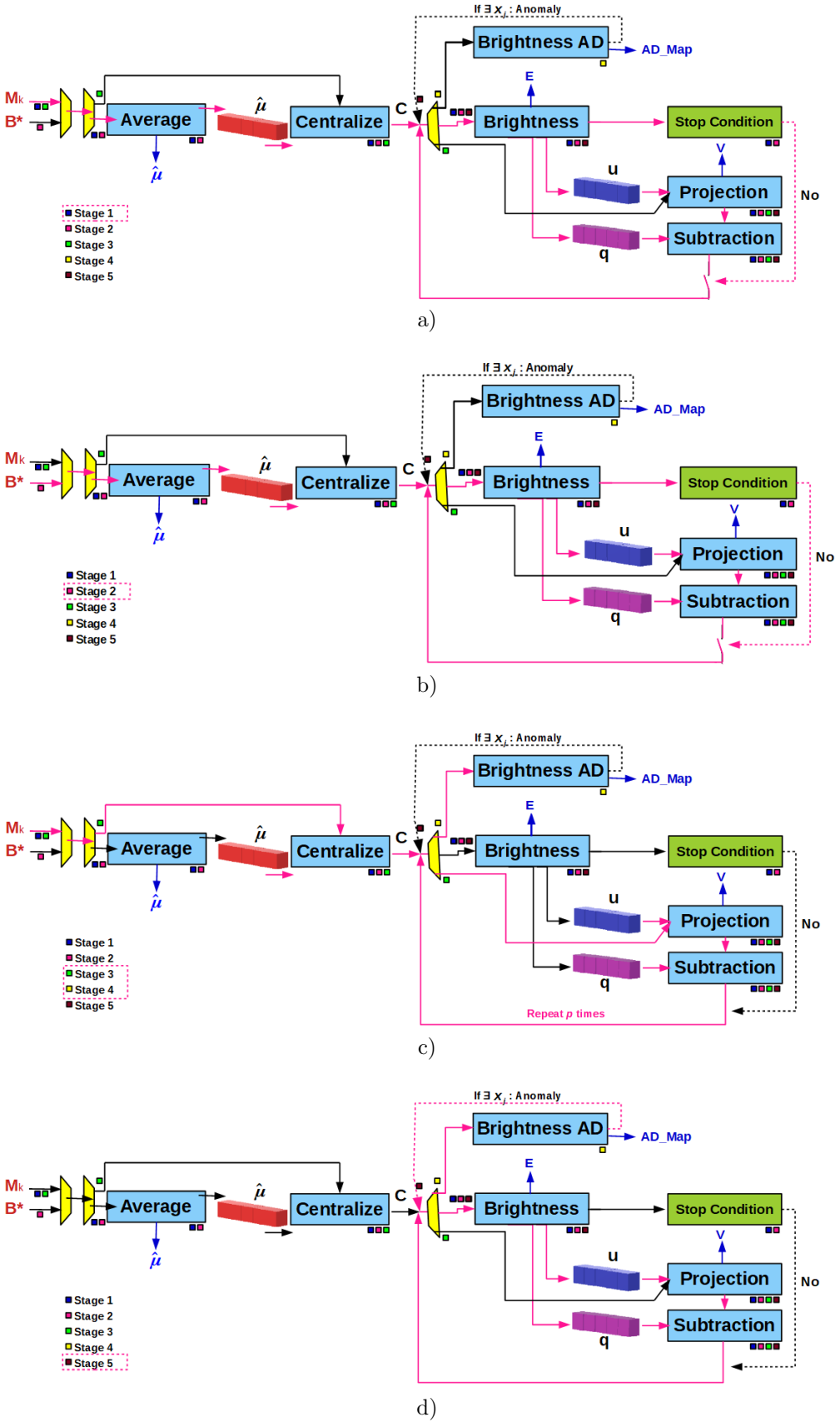


FIGURE 5.4: Detailed view of the schematic diagram displayed in Figure 5.3. (a) Stage 1. (b) Stage 2. (c) Stages 3 and 4. (d) Stage 5.

Preprocessing stage is also launched. Additionally, $\mathbf{V}$ vectors must be also scaled exploiting the dynamic range offered by the input parameter N_{bits}. As it was mentioned in Section 4.3.2.3 of Chapter 4, values of each element within $\mathbf{v}_n$ are typically in the range of $(-1, 1]$. It is because $\mathbf{E}$ pixels in the HyperLCA compressor are extracted independently for all sensed image blocks and hence, each $\mathbf{v}_n$ is a function of each $\mathbf{e}_n$ selected from the same image block, $\mathbf{M}_k$. Nonetheless, in *Stage 3* of the current approach, pixels within the image block in question, $\mathbf{M}_k$, are projected onto the $\mathbf{Q}$ and $\mathbf{U}$ vectors that were estimated from previous frames in *Stage 2*. In this context, light conditions could fluctuate among new sensed frames during the flight mission. Therefore, it could derive in higher values of $\mathbf{v}_n$ elements when illumination conditions change during the acquisition of new hyperspectral frames whilst keeping the $\mathbf{Q}$ and $\mathbf{U}$ vectors, which may be obtained under other environmental conditions. Consequently, values of each element within $\mathbf{v}_n$ could be higher in such unlikely situations. Therefore, we have considered to set the limits of the value range in $[-2, 2]$, which would prevent overflowing even with an increment of the pixel brightness up to a factor of 2. A wider range could be defined as well, though it derives in the decrease of the precision for representing elements of $\mathbf{V}$ vectors after the scaling, since a greater variability in the data has to be represented with N_{bits} bits. Hence, $\mathbf{V}$ vectors are ultimately scaled according to the function displayed in Equation 5.1. To sum up, though these stages of the HyperLCA compressor are not displayed in Figures 5.3 and 5.4, they are addressed as well.

$$v_{j_{\text{scaled}}} = (v_j + 2) \cdot (2^{N_{\text{bits}}-2} - 1) \tag{5.1}$$

In general, the work-flow inherent to the anomaly detection process largely follows the same scheme as the HW-LbL-FAD method. Nonetheless, the compression process has been subjected to the methodological changes introduced by the anomaly detection one. For this reason, the generated bitstream representative of the compressed data slightly differs from the one defined by the original HyperLCA algorithm. In this current version, the generated bitstream consists of two parts: the header, which contains the global information about the HSI that is needed to later decompress it, and $k+1$ data packages with the compressed data necessary to reconstruct each image block, $\mathbf{M}_k$.

Regarding the header, it collects the following data:

1. Size of the HSI (nc and nb): they represent the number of columns and spectral bands, respectively and, are coded as plain binary using 16 bits each.

2. The number of pixels within an image block $\mathbf{M}_k$, BS, coded as plain binary using 16 bits.

3. The number of image blocks used to estimate the background distribution, n_f, coded as plain binary using 16 bits.

4. The number of bits per pixel per band used to save the sensed HSI, DR, coded as plain binary using 8 bits.

5. The number of bits per pixel per band used to save values of $\mathbf{V}$, N_{bits}, coded as plain binary using 8 bits.

The second part of the bitstream that contains the information of each individual block of image pixels, $\mathbf{M}_k$, may be packed in four different ways according to the algorithm stage, as shown in Figure 5.5.

1. *Stage 1*: The bitstream generated in this stage is very similar to the one created by the HyperLCA compressor since the average pixel, $\hat{\boldsymbol{\mu}}$, the selected pixels, $\mathbf{E}$, and the projection vectors, $\mathbf{V}$, are codified and packaged. The main difference lies in that the number of selected pixels, p, is not known beforehand as in the HyperLCA algorithm, since it depends on the fulfilment of the defined stopping condition. In this regard, the process that will later decompress the data receives an array of bits and it must identify somehow the vectors that correspond with each iteration, n. For this reason, an additional bit is introduced among the couple of vectors, $\mathbf{e}_n$ and $\mathbf{v}_n$, estimated in each iteration: 0 means that a new iteration will be addressed and new $\mathbf{e}_n$ and $\mathbf{v}_n$ vectors will be incorporated to the bitstream, while 1 means that the processing of the actual image block, $\mathbf{M}_k$, is over and thus, a new $\mathbf{M}_{k+1}$ will be then managed. This synchronization barrier will be also employed in the other strategies considered for the data packing. Figure 5.5a displays a graphical representation of the data structure within the bitstream for this first stage, which only affects the first n_f hyperspectral frames.

2. *Stage 2*: In this point, only the average pixel, $\hat{\boldsymbol{\mu}}$, and the selected pixels, $\mathbf{E}$, are codified and packaged, as it can be seen from Figure 5.5b. It is because they are employed solely for modelling the background information that is essential for the reconstruction of the spectral information contained in the subsequent image blocks.

3. *Stages 3 and 4*: When only *Stages 3* and *4* are serially performed means that the image block in question, $\mathbf{M}_k$, does not contain any anomalous spectra. Therefore, the data can be perfectly representative by pixels selected in previous *Stage 2*. For this reason, projection vectors, $\mathbf{V}$, are solely part of the bitstream, as displayed in Figure 5.5c. In this case, the aforementioned method to distinguish among vectors is not included since the number of p $\mathbf{Q}$ and $\mathbf{U}$ vectors used for projecting the data is actually known since the *Stage 2*.

4. *Stages 3, 4 and 5*: The execution of the *Stage 5* implies that the image block in question, $\mathbf{M}_k$, is corrupted by anomalous spectra. For this reason, the bistream framework slightly differs from the above mentioned situations free of anomalies. In this sense, the bitstream also comprises the projection vectors, $\mathbf{V}$, of the image pixels onto the $\mathbf{Q}$ and $\mathbf{U}$ vectors representative of the background pattern and the additional selected $\mathbf{e}_n$ vectors and their corresponding $\mathbf{v}_n$ as well. Figure 5.5d displays the structure of the data within this type of data packages.

Additionally, an extra bit is also included in the bitstream once each package has been codified. The rationale behind this is to indicate if more image blocks, $\mathbf{M}_k$, will be codified and transmitted to the ground segment or if the flight mission has been completed. The former situation is represented by zero value of the mentioned bit while the latter by one.

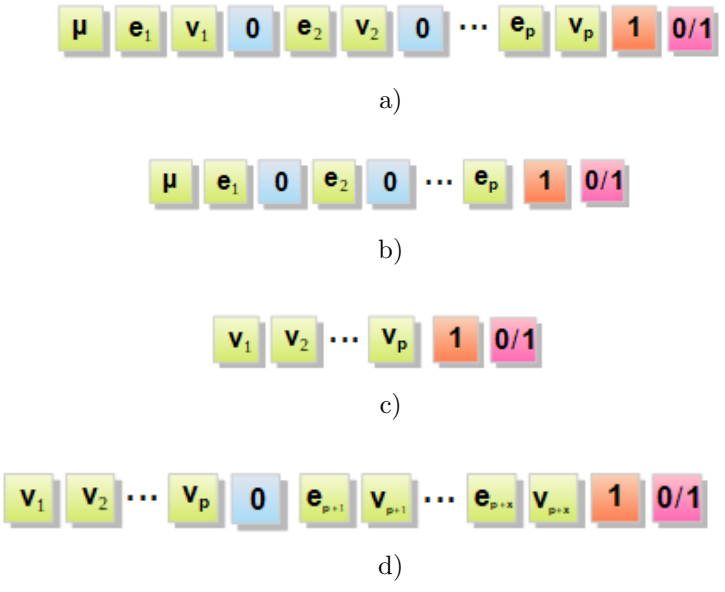


FIGURE 5.5: Data package structure for the different stages of the HADeLoC proposal for the simultaneous execution of the anomaly detection process and the lossy compression of HSIs. (a) Stage 1. (b) Stage 2. (c) Stages 3 and 4 (no anomalies in the image block, $\mathbf{M}_k$). (d) Stages 3, 4 and 5 (existence of anomalies in the image block, $\mathbf{M}_k$).

Given the above, it is reasonably safe to conclude that although the minimum desired compression ratio cannot be known in advance unlike the original HyperLCA compressor, very high compression ratios will be reached. The reasoning behind this is that the average pixel, $\hat{\boldsymbol{\mu}}$, the selected pixels, $\mathbf{E}$, and the projection vectors, $\mathbf{V}$, are always transmitted in the HyperLCA compressor but this only eventuates in the first n_f hyperspectral frames in the current methodology for the estimation of the background distribution. In this sense, n_f represents a negligible part of the total number of image blocks to be streamed. In fact, $\mathbf{Q}$ and $\mathbf{U}$ vectors could be uploaded in advance from previous flight missions. For the other image blocks, $\mathbf{M}_k$ ($k > n_f$), just the projection vectors, $\mathbf{V}$, are packaged when non anomalies are detected. In the opposite scenario, it might be packaged more bits in some cases but the likelihood of the existence of anomalous spectra is very low, which could be set to less than 10% of the image blocks.

5.3 Experimental Results

The performance of the HW-LbL-FAD algorithm for the detection of anomalous spectra and the HyperLCA algorithm for the lossy compression of HSIs were carefully evaluated in their corresponding Chapters 3 and 4. On the basis that the results obtained by the ADeLoC approach are identical to those obtained by the aforementioned methods, this Section looks more closely to the evaluation of the HADeLoC performance for the simultaneous detection of anomalous pixels and the lossy compression of HSIs. The strengths and limitations of the proposed method have been extensively discussed and the quality of the results for both targeted applications has been compared with those obtained by the original methods, the HW-LbL-FAD and the HyperLCA. Additionally, a first approximation towards the estimation of the savings in terms of the number of operations to be computed for the concurrent implementation of both targeted analysis techniques is also set out, though a through evaluation is made in Chapter 6.

5.3.1 Reference Hyperspectral Data

For evaluating the performance and effectiveness of the proposed methodology and in the interest of the comparison with the original HW-LbL-FAD and HyperLCA methods, the bunch of real hyperspectral data collected by the acquisition platform described in [89] and employed in preceding Chapters 3 and 4, is also used in this Section. For the sake of

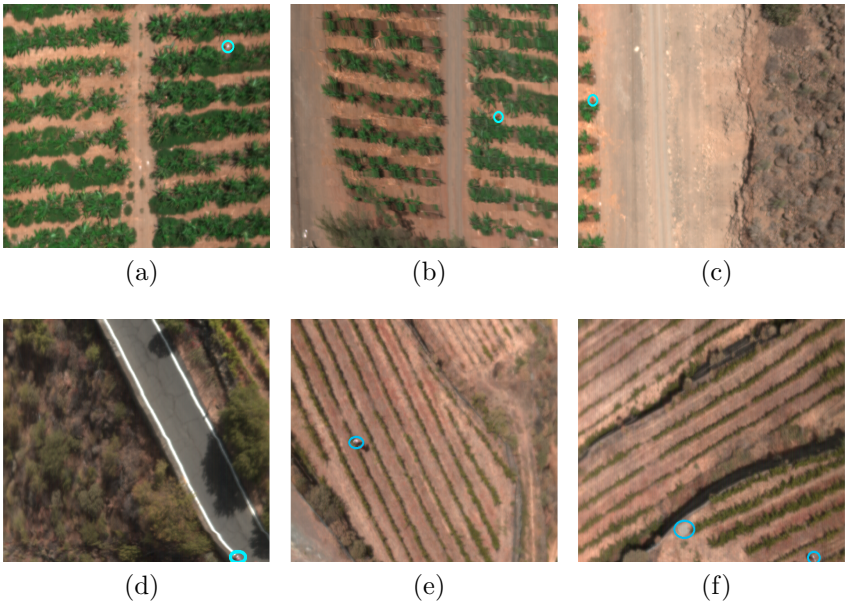
clarity, a brief summary about these image descriptions is provided below, although we encourage the reader to see Chapter 3 to expand the details about the flight campaigns in which images were taken.

This data set was collected over multiple farming areas on the island of Gran Canaria (Spain) by a pushbroom sensor mounted on a UAV. In particular, the reference images are selected portions of some swaths within three different flight campaigns. These data cover the spectral information from 400 to 1000 nm using 160 spectral bands and consist of 825 lines height, each line comprising 1024 hyperspectral pixels with 12-bits depth. A RGB representation of these hyperspectral image portions are displayed in Figure 5.6. Images displayed in Figures 5.6 a–c were taken at a height of 72 m over the ground at a speed-rate of 6 m/s with a camera frame-rate of 125 frames per second (FPS), resulting in a ground sampling distance in line and across line of approximately 5 cm. Data shown in Figure 5.6 d was sensed in a second flight campaign performed at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm. Finally, frames exhibited in Figures 5.6 e–f were scanned at a flight height of 45 m over the ground and at a speed of 6 m/s with the hyperspectral camera capturing at 200 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm.

The aforementioned images were calibrated using a white and dark calibration to obtain reflectance values. Nonetheless, either orthorectification or georeferencing processes were not carried out for the acquired raw data. In this sense, images were built up just by placing the subsequent captured hyperspectral frames one next to the other [156]. This does not degrade the quality of the experiments carried out in this work since the tested algorithms do not use any kind of spatial information. A notable aspect of these images is the existence of some anomalous artefacts, such as some humans and concrete construction, which have been circled in blue in Figure 5.6.

5.3.2 Assessment Metrics

The goodness of the proposed method has been evaluated from two different perspectives, the quality of the compression performance and the efficiency in the detection of anomalous pixels. Firstly, the compression performance has been twofold evaluated by means of the achieved compression ratio, measured as the ratio between the data volume



(a) (b) (c)

(d) (e) (f)

FIGURE 5.6: RGB representation of the employed test bench. Pixels enclosed in blue circles represent some anomalous spectra. (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Drone Image 4. (e) Drone Image 5. (f) Drone Image 6.

before and after the compression, and the average number of bits per pixel per band, *bpppb*, used for representing the compressed images. Secondly, the missing data after the compression-decompression process have been also assessed using five different quality metrics: the *Signal-to-Noise Ratio (SNR)*, the *Root Mean Squared Error (RMSE)*, the *Maximum Absolute Difference (MAD)*, the *Structural Similarity index (SSIM)* and the *Peak Signal-to-Noise Ratio (PSNR)*. Regarding the anomaly detection process, the evaluation of the detection performance is visually made at object-level through the description of the resulting binary maps where anomalies and background elements are segmented. This is because the test images have been sensed at high altitudes and the exact position of the anomalies in the field was not measured. As a consequence, anomalous entities cover a very small number of image pixels. Additionally, the pixels at the object borders are mixed with the background. For this reason, it is very difficult to establish precise boundaries and hence, generate accurate pixel-level ground-truths.

5.3.3 Compression performance of the proposed HADeLoC approach

This section discloses the results obtained in all addressed experiments with the purpose of evaluating the goodness of the proposed methodology for the lossy compression of HSIs.

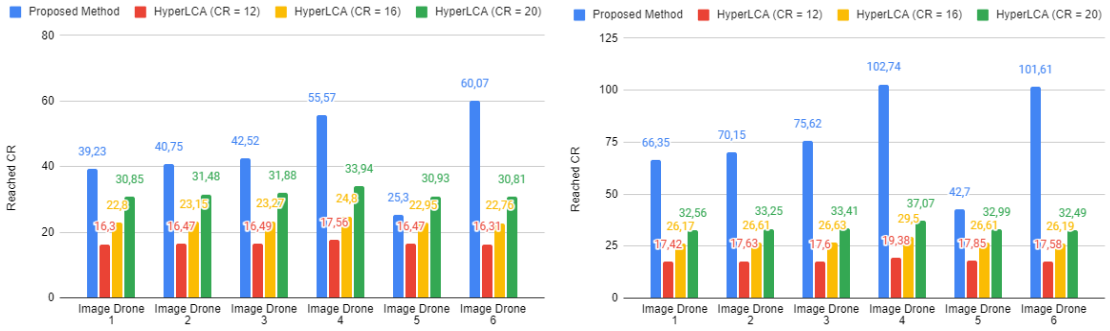
On the basis that in preceding Chapters it was demonstrated the feasibility of the proposed set of core operations for their implementation using floating-point notation and integer arithmetic based on fixed-point notation, only the results obtained by the *Float32* version are analysed in this Section, in pursuit of reducing the complexity of the evaluation and comparison with the performance of the original HyperLCA compressor.

Image	N_{bits}	BS	CR	*bpppb*	SNR	MAD	$RMSE$	$PSNR$	$SSIM$
Image Drone 1	14	1024	39.23	0.31	39.45	287.00	15.55	48.41	0.98
	10	1024	66.35	0.18	39.32	286.00	15.79	48.28	0.98
Image Drone 2	14	1024	40.75	0.29	38.03	317.00	16.38	47.96	0.96
	10	1024	70.15	0.17	37.90	320.00	16.62	47.83	0.96
Image Drone 3	14	1024	42.52	0.28	40.09	203.00	11.78	50.82	0.96
	10	1024	75.62	0.16	39.76	205.00	12.24	50.49	0.96
Image Drone 4	14	1024	55.57	0.22	21.49	470.00	49.68	38.32	0.62
	10	1024	102.74	0.12	21.47	471.00	49.79	38.30	0.61
Image Drone 5	14	1024	25.30	0.47	33.66	568.00	23.97	44.65	0.90
	10	1024	42.70	0.28	33.58	567.00	24.18	44.58	0.90
Image Drone 6	14	1024	60.07	0.20	30.19	371.00	33.48	41.75	0.85
	10	1024	101.61	0.12	30.18	372.00	33.52	41.74	0.85

TABLE 5.1: Compression Results. Achieved *CR, bpppb, SNR, MAD, RMSE, PSNR* and *SSIM* for the six data sets.

Unlike the HyperLCA algorithm, the minimum compression ratio to be desirable, CR, is not required as input parameter since the number of p pixels to be selected in *Stages 1* and *2* of the current solution is estimated according to the outcome of a quality stopping condition. Additionally, only settings of $BS = 1024$ have been evaluated since the image acquisition system captures 1024 spatial pixels per scanned cross-track line, as well as, the HyperLCA algorithm gets the best possible results with this configuration. Therefore, experiments carried out for evaluating the behaviour of the proposed methodology for the lossy compression of HSIs have been done according to $BS = 1024$ and $N_{bits} = [14, 10]$. As can be seen, N_{bits} values have been increased in two bits compared to when the HyperLCA algorithm was evaluated in Chapter 4 ($N_{bits} = [12, 8]$). It is due to the scaling process of $\mathbf{V}$ vectors presented by Equation 5.1. Unlike the HyperLCA algorithm where values of $\mathbf{V}$ vectors after scaling are between [0,2], they are ranging from [0,4] here or even more for increments of pixel brightness higher than 2. Table 5.1 collects the obtained results

for the assessment metrics defined in Section 5.3.2, that is, the reached CR, the *bpppb*, the SNR, the MAD, the $RMSE$, the $SSIM$ and, new in this Chapter 5, the $PSNR$. In addition, Figures 5.7 and 5.8 compare the obtained CR, SNR, MAD, $RMSE$ and $SSIM$ with those results analysed in Chapter 4 for the HyperLCA algorithm. Since the HyperLCA performance was evaluated in terms of three different settings of the minimum desirable $CR = [12, 16, 20]$ in this preceding Chapter 4, the obtained results for all these tested configurations have been included in the graphics displayed in Figures 5.7 and 5.8. In addition, a correspondence in the comparison has been established between N_{bits} values equal to 14 and 12 bits and, 10 and 8 bits, respectively.



(a) (b)

FIGURE 5.7: Comparison among the compression ratios obtained by the HyperLCA algorithm for different settings of the input parameters $CR = [12, 16, 20]$ and $N_{bits} = [12, 8]$ and, by the HADeLoC proposal with $N_{bits} = [14, 10]$. (a) $N_{bits} = 14$ and 12. (b) $N_{bits} = 10$ and 8.

Several observations may be drawn from the analysis of these results:

1. The CR obtained by the proposed methodology is automatically calculated as a function of the spectral variability present in the HSIs to be independently analysed. From results displayed in Figure 5.7, it is inferred that the obtained CRs are in general much higher than those obtained by the HyperLCA compressor for an input minimum $CR = 20$. These differences are more pronounced for N_{bits} equal to 10. In fact, CRs obtained by configurations of $N_{bits} = 14$ are roughly 40-46% lower than using $N_{bits} = 10$. It makes totally sense since the same p pixels are selected in *Stages 1* and *2* and the same number of anomalous spectra are detected. Indeed, N_{bits} is solely used for the scaling and encoding of each element within $\mathbf{V}$ vectors. On this basis, compressed data packaged using $N_{bits} = 14$ employ 40% more bits than using $N_{bits} = 10$. In those cases where the proportion in the compression ratios is higher ($> 40\%$), it is due to the additional data compression introduced by the coder.

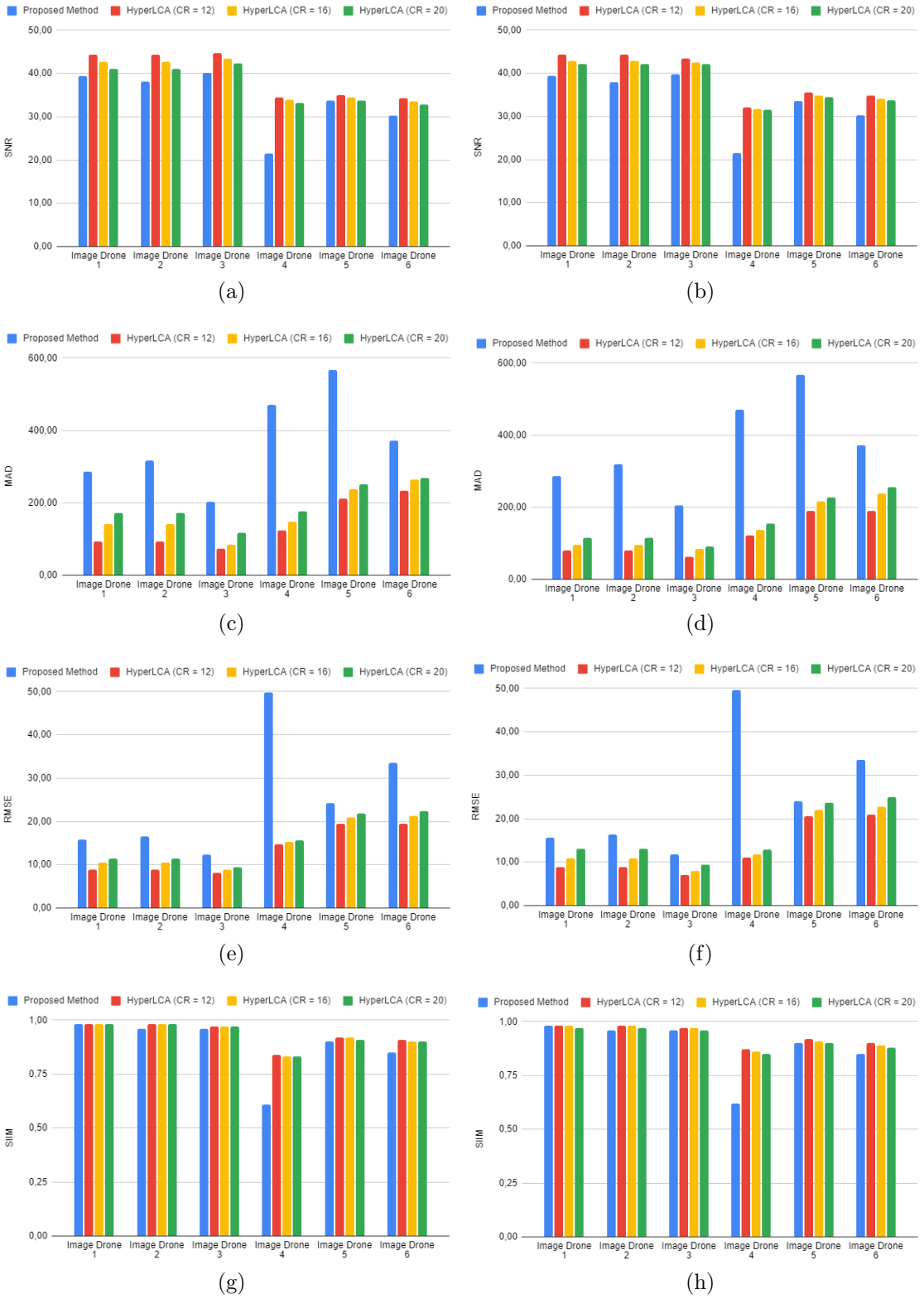


FIGURE 5.8: Comparison among the quality of the compression results obtained by the HyperLCA algorithm for different settings of the input parameters $CR = [12, 16, 20]$ and $N_{bits} = [12, 8]$ and, by the HADeLoC proposal with $N_{bits} = [14, 10]$. (a) *SNR* for $N_{bits} = 14$ and 12. (b) *SNR* for $N_{bits} = 10$ and 8. (c) *MAD* for $N_{bits} = 14$ and 12. (d) *MAD* for $N_{bits} = 10$ and 8. (e) *RMSE* for $N_{bits} = 14$ and 12. (f) *RMSE* for $N_{bits} = 10$ and 8. (g) *SIIM* for $N_{bits} = 14$ and 12. (h) *SIIM* for $N_{bits} = 10$ and 8.

2. Regarding the results of the quality metrics collected in Table 5.1, the data losses derived from the use of less number of bits for representing $\mathbf{V}$ vectors are negligible between $N_{bits} = 14$ and 10 settings. Therefore, it is worthwhile algorithm configurations using $N_{bits} = 10$ since roughly 60% more compression ratio is obtained for almost the same spectral distortions introduced by the lossy compression process.

3. In general terms, the quality of the compression results are worse than those obtained by the HyperLCA algorithm for all assessment metrics. Nonetheless, it should keep in mind that the obtained CRs are much higher than those shown by the HyperLCA algorithm. In fact, CRs bigger than 100 are reached for $N_{bits} = 10$ settings. For instance, the CR obtained by the proposed methodology and by the HyperLCA algorithm for a minimum desirable $CR = 20$ is almost the same for *Drone Image 5* and hence, the SNR values reach similar outcomes for both targeted methods. Nonetheless, the test bench is in general composed of dark images that do not exploit the full dynamic range available by the sensor. It can be seen from the $PSNR$ values collected in Table 5.1. $PSNR$ represents the ratio between the maximum possible power of a signal ($2^{12} - 1$ for the data set employed in the experiments) and the power of corrupting noise. On the contrary, the SNR represents the ratio between the power of the signal and the power of the noise. In this sense, the obtained $PSNR$ values are much higher than the SNR values and it is indicative that images were obtained under dark light conditions. For this reason, the proportion of the inherent image noise is comparable to the signal power and hence, the SNR values get affected.

4. The presence of undesirable distortions derived from sparkles, scattering or from the sweeping motion of the data acquisition platform might lead to a misrepresentation of the background model estimated in *Stages 1* and *2* and hence, this fact impacts the quality of the compression in the subsequent image blocks. This phenomena can be observed in *Image Drone 4* that is actually the worst reconstructed image after the compression/decompression process. This data set is very challenging since pixels located on the white lines of the road are close to be very saturated and hence, their are characterized by a high l^2-*norm* value or brightness. Consequently, they will be selected as pixels of interest for the background modelling performed in *Stages 1* and *2*, though they are not really representative of the spectra contained in the subsequent hyperspectral frames. In addition, the stopping condition defined for the pixel selection process is actually based on the ratio between pixel brightnesses. Consequently, the existence of outliers in the frames employed for the estimation of

the background pattern could lead to not trustworthy **Q** and **U** vectors. Given the above, enough n_f hyperspectral frames must be taken to ensure that the variability of the background has been fully covered to generate accurate background models.

5. Finally, concerning the MAD assessment metric, it is the one that shows the greatest differences compared with the results obtained by the HyperLCA compressor. As a reminder, the MAD reveals the biggest reconstruction error among all image pixels after the compression/decompression process. In this sense, pixels placed on the rounded edges of the anomalous entities are composed in its majority by background spectra mixed with the anomalous signature. Consequently, they cannot be perfectly reconstructed by a linear combination of the background spectra and therefore, the biggest MAD values are reached by these pixels in the proposed methodology. To illustrate this point, Figure 5.9 displays a portion of the MAD value map near the location of the anomalous entity in *Drone Image 5*.

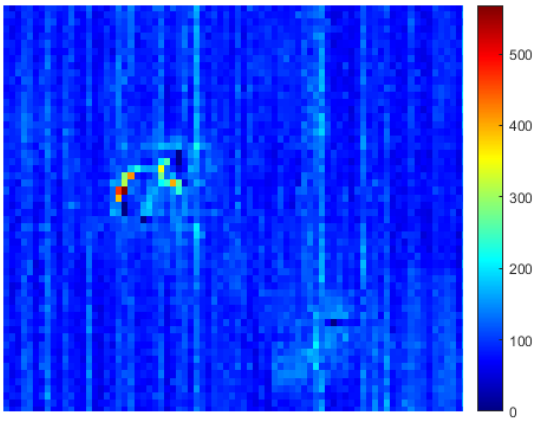


FIGURE 5.9: Portion of the MAD value map around the anomalous entity located in *Image Drone 5*.

Apart from the analysis made in above lines, we have also compared the HyperLCA algorithm with the proposed methodology under more similar conditions. In this sense, we have forced the HyperLCA algorithm to reach similar CRs as those obtained by the proposed method. Under this scenario, Figure 5.10 displays graphical representations of the achieved CRs and the quality of the compression results in terms of the SNR, the MAD, the $RMSE$ and the $SIIM$ obtained by both targeted algorithms. Four main observations may be drawn from the analysis of these results:

1. Concerning the MAD and the $RMSE$ assessment metrics, the HyperLCA performance results in less losses of information after the compression/decompression process. The main methodological differences between both targeted algorithms start from *Stage 3* of the HADeLoC proposal. In this sense, the HyperLCA algorithm centralizes each image block, $\mathbf{M}_k$, using the average pixel, $\hat{\boldsymbol{\mu}}$, obtained from pixels within $\mathbf{M}_k$ and, codifies and packages selected pixels, $\mathbf{E}$, within the image block in question. On the contrary, the proposed methodology uses the average pixel, $\hat{\boldsymbol{\mu}}$, and the orthogonal vectors $\mathbf{Q}$ and $\mathbf{U}$ estimated from pixels selected from the first sensed n_f image blocks in the *Stage 2* of the algorithm. This makes us conclude that the quality of the compression performance of the proposed methodology is influenced by the environmental conditions under the data were sensed.

2. On the basis of the above stated hypothesis, the data set employed in the experiments carried out in this Chapter 5 were calibrated using white reference and dark images obtained prior to the beginning of the flight campaigns. Therefore, they could have been sensed under different light conditions to those shown when the first n_f hyperspectral frames were captured; besides, the subsequent frames analysed in the *Stage 3* using data obtained from *Stage 2*.

3. On the basis that operations involved in *Stage 1* of the proposed methodology are identical to those performed by the *HyperLCA Transform* and the reached CRs are very close among each other, almost the same pixels are selected from the first n_f frames by both targeted methods. Therefore, pixels placed on the white lines of the road contained in *Drone Image 4* are selected as pixels of interest by both methods. Nonetheless, *Drone Image 4* is once again far from being equally reconstructed using the data packaged by the proposed method than the HyperLCA algorithm. This fact reinforces the importance of the calibration and the radiometric calibration of the data.

4. Finally, the worst MAD values are also obtained by the proposed methodology due to the same reasons analysed in the preceding analysis, that is, due to the pixels located at the rounded edges of the anomalous entities.

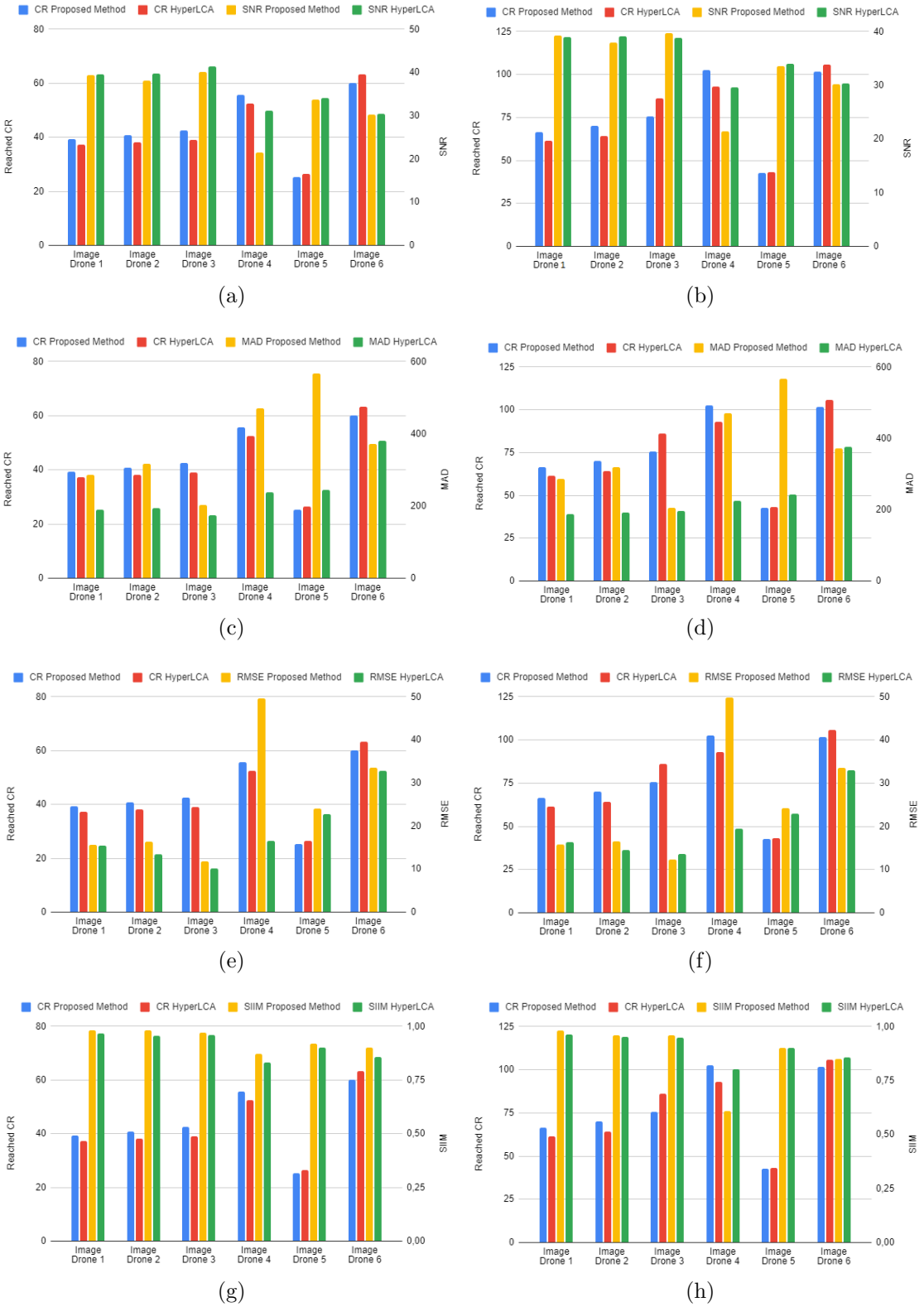


FIGURE 5.10: Comparison among the quality of the compression results obtained by the HyperLCA algorithm with N_{bits} = [12,8] and by the HADeLoC proposal with N_{bits} = [14,10] for similar reached CRs. (a) SNR for N_{bits} = 14 and 12. (b) SNR for N_{bits} = 10 and 8. (c) MAD for N_{bits} = 14 and 12. (d) MAD for N_{bits} = 10 and 8. (e) $RMSE$ for N_{bits} = 14 and 12. (f) $RMSE$ for N_{bits} = 10 and 8. (g) $SIIM$ for N_{bits} = 14 and 12. (h) $SIIM$ for N_{bits} = 10 and 8.

5.3.4 Anomaly Detection performance of the proposed HADeLoC approach

In general terms, the proposed methodology for the concurrent implementation of both targeted applications was structured in such a way as to maintain the same schedule based on the processing stages defined by the HW-LbL-FAD algorithm. Indeed, the compression process is actually the application that has undergone several changes, as shown the results presented in Section 5.3.3. For this reason, the anomaly detection results given as output by the current proposal are identical to those obtained by the original HW-LbL-FAD algorithm. In this regard, two-dimensional (2D) binary maps displayed in Figure 5.11 clearly demonstrated this. As it was done in preceding Chapters, the aforementioned anomaly detection maps have been superimposed on a panchromatic representation of the scenes to be analysed in order to make easier the result interpretation. Pixels and spatial lines corrupted by anomalous signatures have been also highlighted in red color. As can be seen, detected anomalous pixels are the same, even regardless the configured values of N_{bits} parameter. It makes totally sense since this parameter is solely employed for preprocessing and encoding the $\mathbf{V}$ vectors that are part of the bitstream, which actually concerns to the compression process.

5.3.5 Discussions about the HADeLoC performance

From the findings of the foregoing study, some discussions emerge about the HADeLoC proposal:

1. In the solution analysed in this Section, the background distribution is estimated from several of the first sensed HSI blocks, n_f, under the assumption that they are fully representative of the background pattern. On this basis, enough n_f hyperspectral frames must be taken to ensure that all the spectral variability is covered and hence, to estimate a trustworthy background model. Nonetheless, this methodology could be not a feasible solution in very heterogeneous scenarios or beneath the presence of undesirable distortions derived from sparkles, scattering or from the sweeping motion of the data acquisition platform, which may be one limitation of our proposal. In this regard, a potential solution might be to already have a set of reliable orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$, resulting from the study and the analysis of the spectral signatures taken from previous flights over the same areas.

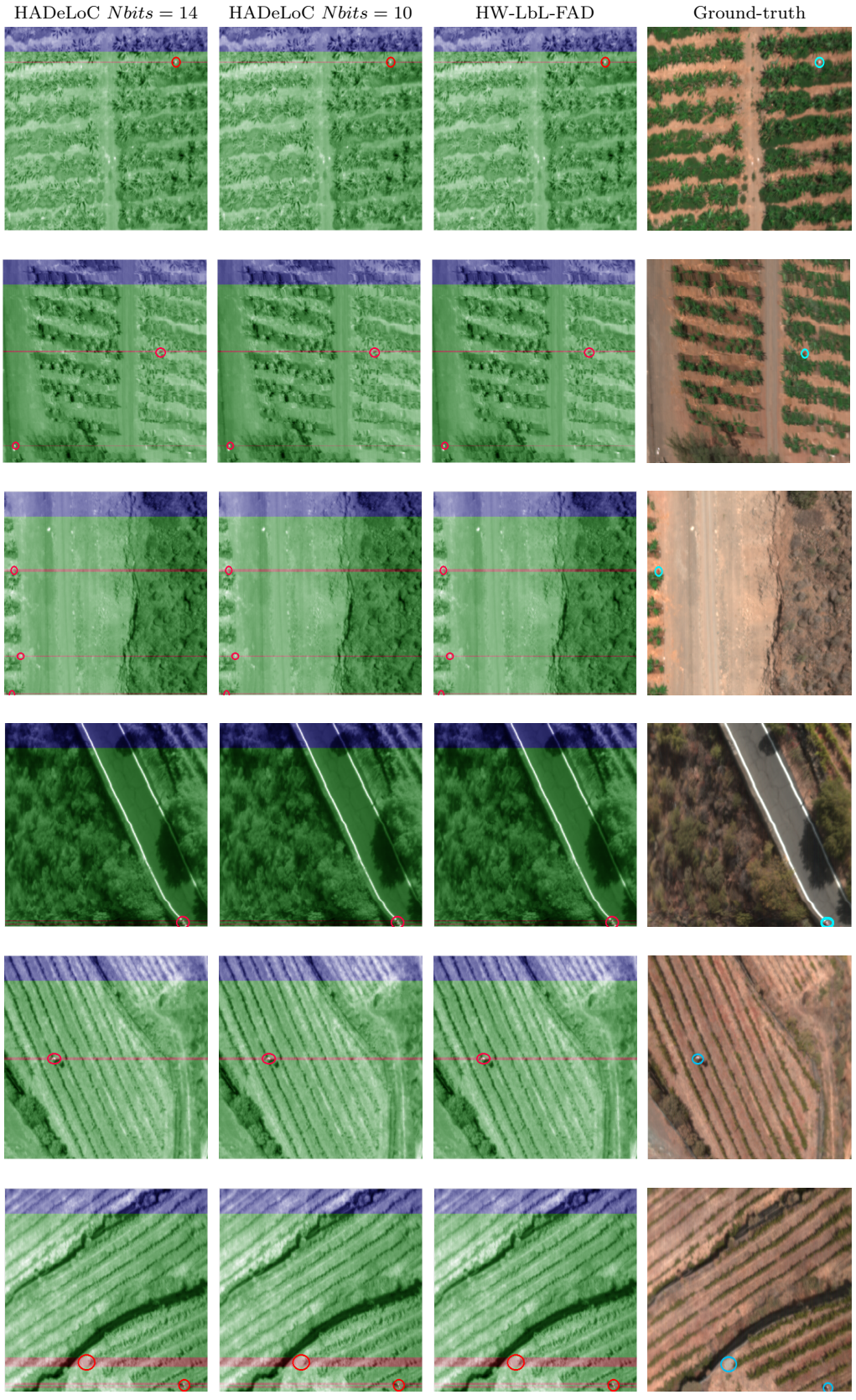


FIGURE 5.11: Anomaly detection results obtained by the HW-LbL-FAD algorithm and by the HADeLoC proposal.

2. It should also be mentioned the importance of the calibration and the radiometric correction of the HSI to obtain accurate reflectance values. It is a determining factor since subsequent frames are centralized and projected over a subset of vectors obtained from previous frames that may have been captured under other lighting conditions. For instance, the data set employed in the experiments carried out in this Chapter 5 were calibrated using white reference and dark images obtained prior to the beginning of the flight campaigns. Therefore, luminance conditions over the course of these data acquisition campaigns are likely to bear the brunt of the environmental shifts. That is why there is an increasing scientific motivation towards the definition of more accurate radiometric correction methods under operational conditions that take into account the incident radiance at each instant [31, 219–222].

3. It is also important to mention the adequacy of exploiting the full dynamic range of the hyperspectral sensors. In this sense, proper settings of the camera exposure time according to the capturing frame rate are required in order to prevent from taken dark images where a small range of the full sensor dynamic range is covered. In this situations, as those presented in Section 5.3, the ratio between the signal and the noise intrinsic to the image capture, in other words the *SNR*, gets affected since the power of noise is comparable to the power of the signal.

4. One of the main conclusions drawn from the analysis of the compression results conducted in preceding Section 5.3.3 is that the highest and worst *MAD* values were attained by the pixels located in the rounded edges of the anomalous entities. It is because they are mixed pixels composed of both anomalous signatures and background spectra. In some cases they are not identified as anomalous and hence, they are reconstructed as solely a linear combination of the vector space that defines the background distribution. Against this backdrop, a workable solution consists in always selecting a couple of additional pixels, $\mathbf{e}_n$, in the same way as it done by frames containing anomalous spectra.

5.3.6 Computational Complexity Analysis

The greatest strength of the proposed ADeLoC and HADeLoC approaches is the ability to reuse operations when both the anomaly detection and the lossy compression processes are simultaneously launched in the same piece of hardware. Consequently, the number of operations to be executed considerably decreases compared to other approaches where

no algorithmically related processes are implemented. On this basis, the computational complexity concomitant with the proposed methods are evaluated in this Section in terms of the number of basic operations (OPs) involved in each algorithm stage. For clarity, OPs are simple calculations such as additions, subtractions, multiplications, and divisions. In addition, it is also compared with the number of OPs to be performed by classical approaches in which the lossy compression and the anomaly detection processes are executed as two independent algorithms. For that purpose, the HW-LbL-FAD algorithm and the HyperLCA compressor are used in the comparative analysis.

Table 5.2 collects the number of OPs required to process one image block, $\mathbf{M}_k$, for each method. Since the number of OPs computed by the proposed methodologies and the HW-LbL-FAD algorithm is different according to the algorithm stages, it is specified for each case. In addition, Table 5.2 just retains the number of OPs involved by the proposed set of core operations, discarding those operations performed by the *Preprocessing* and the *Entropy Coding* stages belonging to the compression process. Regarding the HyperLCA and the HW-LbL-FAD algorithms, the number of pixels extracted per image block is different and hence, there are referred to as p_c and p_{AD}, respectively. With regard to the *Stage 1* of the ADeLoC method, the number of p iterations to be carried out by the set of core operations is determined by the requirements imposed by both p_c and p_{AD} and therefore, it is represented by p. Finally, the HADeLoC maintains the stopping condition inherent to the HW-LbL-FAD algorithm and hence, the number of pixels selected per image block is equal to p_{AD}.

For analysing the reduction in the number of OPs to be carried out for the approaches proposed in this Chapter, let us set an example using an hypothetical HSI with the same dimensions as those images employed in this manuscript ($nr = 1024$, $nc = 1024$, $nb = 160$). For the sake of simplicity, we have considered the number of selected pixels p and p_{AD} equal to p_c for every targeted methods, though it is not applied in real scenarios. In this scene, p_c is a function of the minimum desirable CR and the BS in the case of the HyperLCA algorithm and for the others, it depends on the variability of the spectra present in the image block, $\mathbf{M}_k$, in question. Moreover, experience has demonstrated that normally $p_{AD} < pc$. On this basis, the comparison among methods could be done as a function of different values of BS and the minimum desired CR. Additionally, the number of hyperspectral frames used for estimating the background model for addressing anomaly detection, n_f, has been set to 100 frames of 1024 hyperspectral pixels, no matter BS setting. It means that for $BS = 512$, 200 frames of 512 pixels are used, whereas 400 frames of 256 hyperspectral pixels are used for $BS = 256$ settings. Finally, the

number of OPs carried out by the ADeLoC and the HADeLoC methods also depends on the probability of anomalous $\mathbf{M}_k$, x, within the HSI. Since the presence of anomalous pixels is very unlikely, the analysis has also contemplated three different settings of $x = [0.01, 0.05, 0.1]$.

Algorithm	Stage	Complexity
HyperLCA	∇k	$p_c \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$
HW-LbL-FAD	*Stage 1*: $k \leq n_f$	$p_{AD} \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$
	Stage 2:	$p_{AD} \cdot (6 \cdot nb \cdot p_{AD} \cdot nf + nb) + 2 \cdot nb \cdot p_{AD} \cdot nf + nb$
	Stages 3 and 4: $k > n_f$	$BS \cdot (4 \cdot p \cdot nb + 3 \cdot nb)$
ADeLoC	*Stage 1*: $k \leq n_f$	$p \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$
	Stage 2:	$p_{AD} \cdot (6 \cdot nb \cdot p \cdot nf + nb) + 2 \cdot nb \cdot p \cdot nf + nb$
	Stages 3 and 4: $k > n_f$ *(No anomalies)*	$p_c \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$ $+4 \cdot p_{AD} \cdot p_c \cdot nb + 3 \cdot p_c \cdot nb$
	Stages 3 and 5: $k > n_f$ *(Anomalies)*	$p_c \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$ $+4 \cdot p_{AD} \cdot nb \cdot (p_c + BS) + 3 \cdot nb \cdot (p_c + BS)$
HADeLoC	*Stage 1*: $k \leq n_f$	$p_{AD} \cdot (6 \cdot nb \cdot BS + nb) + 2 \cdot nb \cdot BS + nb$
	Stage 2:	$p_{AD} \cdot (6 \cdot nb \cdot p_{AD} \cdot nf + nb) + 2 \cdot nb \cdot p_{AD} \cdot nf + nb$
	Stages 3 and 4: $k > n_f$ *(No anomalies)*	$BS \cdot (4 \cdot p_{AD} \cdot nb + 3 \cdot nb)$
	Stages 3, 4 and 5: $k > n_f$ *(Anomalies)*	$BS \cdot (4 \cdot p_{AD} \cdot nb + 3 \cdot nb)$ $+6 \cdot p_{extra} \cdot BS \cdot nb + p_{extra} \cdot nb$

TABLE 5.2: Number of OPs performed for the processing of one image block, $\mathbf{M}_k$, by the HyperLCA compressor, the HW-LbL-FAD algorithm and the proposed methodologies for the simultaneous execution of the anomaly detection process and the lossy compression of HSIs, that is, the ADeLoC and the HADeLoC approaches.

Figure 5.12 graphically displays the proportion, in percentage (%), of fewer number of operations performed by the ADeLoC method (Figure 5.12a) and by the HADeLoC approach (Figure 5.12b) compared with the serial execution of the HW-LbL-FAD and the HyperLCA algorithms for different settings of BS, CR and x. For this purpose, p_{extra} pixels selected by the HADeLoC approach for anomalous hyperspectral frames has been set to 2, which means that up to two different anomalous materials may be present in the $\mathbf{M}_k$ in question.

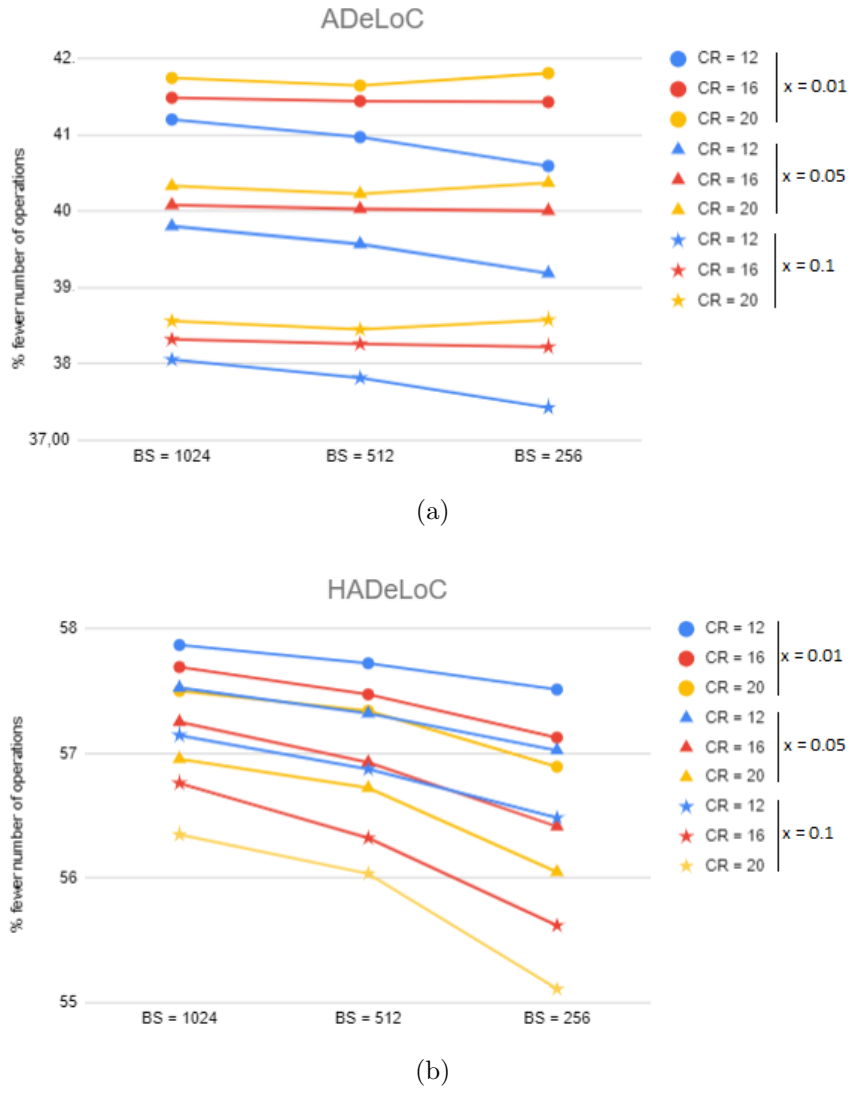


FIGURE 5.12: Reduction in the number of OPs (%) performed by the ADeLoC and the HADeLoC algorithms compared with the serial execution of both the lossy compression and the anomaly detection processes as two independent algorithms.

From results shown in Figure 5.12a, it can be concluded that the ADeLoC approach carries out between approximately 41.21% - 38.58% fewer operations than independently implementing the HW-LbL-FAD algorithm and the HyperLCA methods. Naturally, this reduction in the number of OPs decreases for higher x, since it implies the computation of $4 \cdot p_{AD} \cdot nb \cdot (p_c + BS) + 3 \cdot nb \cdot (p_c + BS)$ extra operations per anomalous block (see Line 8 of Table 5.2). In terms of BS, the ADeLoC makes slightly more operations for smaller BS ($BS = 256$). It is because, $4 \cdot p_{AD} \cdot p_c \cdot nb + 3 \cdot p_c \cdot nb$ operations are used to project the p_c pixels selected from each $\mathbf{M}_k$ over the set of $\mathbf{Q}$ and $\mathbf{U}$ orthogonal vectors in order to see if there are anomalous entities in the current image block (see Line 8 of Table 5.2). In this regard, the smaller BS, the higher the number of total image blocks to

process and hence, more times the aforementioned number of OPs have to be performed. With respect to CR, more number of iterations of the set of core operations is executed for smaller CR since more p pixels are selected. Therefore, the reduction in the number of OPs decreases with lower CR.

From results shown in Figure 5.12b, it can be concluded that the HADeLoC approach carries out between approximately 57.87% - 55.11% fewer operations than independently implementing the HW-LbL-FAD algorithm and the HyperLCA methods. Moreover, the HADeLoC approach implies the execution of 27-30% fewer OPs than the ADeLoC version. Nonetheless, the HADeLoC behaviour according to the CR is opposite to the ADeLoC approach. As it was mentioned in Section 5.2.2.3, the work-flow inherent to the HADeLoC version largely follows the same scheme as the HW-LbL-FAD method. Therefore, the reduction in the number of launched OPs is mainly due to the additional operations carried out by the HyperLCA compressor. Therefore, lower desired CR involves more iterations of the set of core operations for extracting p_c. Since these additional operations are not carried out by the HADeLoC, the reduction in the number of OPs is notoriously higher for smaller CR. Another point to mention is that the reduction in the number of computed operations for settings of $CR = 20$ and x= 0.01 are similar to those obtained by $CR = 12$ and $x = 0.05$ for $BS = 512$. The same happens between $CR = 16$, $x = 0.05$ and $CR = 12$, $x = 0.1$. It is because the proportion of fewer operations inherent to the extraction of less number of p pixels for higher CR is compensated by the higher number of times that p_{extra} pixels are extracted from anomalous image blocks.

In conclusion, it can be observed that the use of a common mathematical method for addressing different targeted applications implies a considerably reduction in the number of computing operations to be carried out. This is expected to be translated into a reduction in the execution times, as well as in the required computational resources, as it will be further analysed in Chapter 6.

5.3.7 General discussions

From the findings of the foregoing analysis, some general discusses emerge:

1. There is a trade-off between the causality in line-by-line approaches and how to model the background distribution. Actually, very few publications can be found within this field in the existing literature [129, 149, 157, 199, 203, 223–225]. In the

solutions analysed in this Section, the background distribution is estimated from several of the first sensed HSI blocks, n_f, under the assumption that they are fully representative of the background pattern. On this basis, enough n_f hyperspectral frames must be taken to ensure that all the spectral variability is covered and hence, to estimate a trustworthy background model. Nonetheless, this methodology could be not a feasible solution in very heterogeneous scenarios or beneath the presence of undesirable distortions derived from sparkles, scattering or from the sweeping motion of the data acquisition platform, which may be one limitation of our proposal. In this regard, a potential solution might be to already have a set of reliable pixels representative of the background resulting from the study and the analysis of the spectral signatures taken from previous flights over the same areas.

2. The strength of the proposed methodologies lies in the definition of a set of common core operations for different hyperspectral analysis imaging applications. It results in many benefits in view of hardware acceleration for real-time or near real-time performance in terms of a reduction in the execution times, hardware resources and above all, in human endeavours. Concerning this latter, it implies the studio and analysis of only a single mathematical approach, which consequently permits to focus the efforts from a methodological and productivity points of view, which consequently results in a reduction in the time to market.

3. Regarding the ADeLoC approach, it outperforms the HADeLoC proposal in terms of the quality of the results, which are the same as those obtained by the original HyperLCA and the HW-LbL-FAD methods but carrying out approximately 41-39% less number of operations. Unlike the HADeLoC, this approach permits to know in advance the minimum compression ratio to be reached by the application since the ADeLoC approach faithfully follows the same algorithmic stages performed by the HyperLCA algorithm. On the contrary, this approach searches for the maximum accuracy in the detection and compression results, relegating to a second plain the issues around the computational complexity. Although it is also a hardware-friendly solution, the ADeLoC approach executes roughly a 37-41% more operations than the HADeLoC version.

4. The HADeLoC approach arises under the search of a more competitive solution in terms of computational resources and execution times at the cost of a slight loss of accuracy in the compression results. In this sense, the HADeLoC version performs

roughly 27-30% less number of operations than the ADeLoC and 58-55% beside executing serially the HyperLCA and the HW-LbL-FAD methods.

5. Finally, the overall conclusion that can be drawn from the foregoing is that there is always a trade-off among the quality of the results, the computational resources and the execution times. These statement has been supported by the methodological sequence presented in this Chapter.

5.4 Conclusions

In recent decades, much effort has been made for the onboard processing of HSIs in applications under real-time constraints. The mainstream solution has been focused on the acceleration of the hyperspectral imaging calculations using high-performance computing infrastructures and parallel techniques. It has been possible due to the latest advances in parallel programmable hardware devices that, on the one hand, have been extended in computing resources and, on the other hand, have drastically reduced their energy consumption. It has permitted to bridge the gap towards the onboard execution of hyperspectral analysis techniques. However, while the computational capability of the next-generation commercial hardware devices is expected to continue growing, so is the resolution of hyperspectral sensors in the spatial, spectral and temporal domains. Therefore, the onboard real-time hyperspectral image processing is far from being an effective solution nowadays, specially when multiple applications must be simultaneously executed in the same piece of hardware. Consequently, it is mandatory the development of new algorithmic solutions in the field of hyperspectral remote sensing on which to confront these potential challenges.

In this Chapter, we have approached the issue around the real-time processing of HSIs from a new algorithmic perspective. Concretely, we have analysed the adequacy of the set of core operations proposed in this Thesis work for the simultaneous execution of multiple hyperspectral analysis techniques. This provides several benefits, above all in the field of onboard hyperspectral imaging processing when some time-sensitive applications must be executed in the same computing hardware device. Firstly, it implies less time and effort during the stage of hardware acceleration since the same product can be reused for several algorithms targeting different applications. Secondly, it permits the execution of several tasks at the same time with the advantage of sharing the most computationally

costly operations, thus reducing the overall computational cost and the required hardware resources.

In particular, we have verified the suitability of the proposed methodology for the concurrent execution of the lossy compression of HSIs jointly with the detection of anomalous signatures. In this sense, three different methodological approaches based on the proposed set of core operations have been studied for performing both targeted applications, seeing them as an evolution towards a highly optimized version that permits the concurrent implementation of both processes. Firstly, it was discussed the feasibility of the proposed methodology for the execution of any such above mentioned analysis techniques using a single configuration of the proposed set of core operations. For doing so, the different computing stages performed by the HW-LbL-FAD algorithm and the HyperLCA compressor, fully described in preceding Chapters 3 and 4, were accurately reproduced. In the same vein, the potential of the suggested methodology for the joint implementation of both targeted applications was secondly analysed. Nonetheless, it was concluded that, though the computing operations performed by the HyperLCA and the HW-LbL-FAD methods are those defined within the proposed set of core operations, different data are processed by them for each case. Therefore, the simultaneous execution of both targeted applications implies whether the replication of some operation blocks or the serial execution of the hyperspectral analysis techniques in question. Naturally, these approaches concern either the hardware resource utilization or the execution times.

To meet this issue, two optimized versions for the simultaneous detection of anomalous pixels and the lossy compression of HSIs were eventually proposed. The first one, referred to as ADeLoC, searches for the highest accuracy in the detection and compression results, whereas the other, namely HADeLoC, prioritizes the optimization of the hardware resources and the minimization of the execution times. The ADeLoC approach ensures the same detection and compression results as the original HW-LbL-FAD and the HyperLCA methods but, launching 41-39% less number of operations. On the contrary, the HADeLoC follows the methodology behind the HW-LbL-FAD algorithm and the HyperLCA compressor but introduces some minor changes in the search of a more competitive solution from a hardware implementation point of view. The experiments carried out prove the benefits of employing this methodology in terms of the number of operations to be performed. Concretely, it was verified that roughly 59-55% fewer operations are executed than if both processes were independently implemented and 30-27% less than the ADeLoC version. Nonetheless, the quality of the results are not as competitive as those obtained by the original HyperLCA compressor, though identical with the HW-LbL-FAD

algorithm. For the foregoing, the overall conclusion that can be drawn is that there is always a trade-off among the quality of the results, the computational resources and the execution times.

Chapter 6

Hyperspectral imaging acceleration through the utilization of embedded systems

Different algorithms for the processing of hyperspectral imagery have been proposed along the previous Chapters of this Thesis. All these algorithmic solutions share a set of common core operations based on projection techniques and more specifically, in the well-known Gram-Schmidt orthogonalization method. This Chapter verifies the benefits of developing algorithmic approaches based on the same mathematical method in terms of reducing the execution-times, the hardware resources and the human endeavours. For doing this, the HW-LbL-FAD, the HyperLCA and the HADeLoC algorithms have been implemented into different kinds of parallel hardware devices, namely graphical processing units (GPUs) and field-programmable gate array (FPGAs). In this sense, the HW-LbL-FAD, the HyperLCA and the HADeLoC methods have been implemented on a Xilinx System on Chip (SoC) FPGA device, while the HyperLCA has been also accelerated in embedded computing boards from NVIDIA.

The proposed set of core operations takes into consideration the hardware-design characteristics of the most commonly used computing platforms. Accordingly, it is evaluated in this Chapter their adaptability to the requirements imposed by the targeted devices. Therefore, the aforementioned algorithms have been implemented in FPGA devices using integer arithmetic and the concept of fixed-point notation, while floating-point notation is exploited for GPUs-based systems.

6.1 Rationale

Along the preceding Chapters, it has been demonstrated that multiple hyperspectral analysis techniques can be efficiently addressed by the same mathematical method and, in particular, by the modified version of the Gram–Schmidt orthogonalization method performed by the set of core operations proposed in this Thesis work. In addition, it has also proved the feasibility of the simultaneous execution of different hyperspectral image processing tasks, whilst optimizing the computational resources, the execution times and the human endeavours invested during the implementation stage. Nonetheless, these assertions have been established from a methodological point of view. In this Chapter, instead, the algorithms developed in this Thesis have been implemented on parallel computing devices, such as graphical processing units (GPUs) and field-programmable gate array (FPGAs), in order to determine the benefits derived from the definition of algorithmic solutions based on the same mathematical method.

In general terms, this Chapter is oriented to achieve the following objectives:

1. Verifying the suitability of the developed algorithms for real-time applications by implementing them on different parallel devices, namely GPUs and FPGAs. In this sense, the HW-LbL-FAD, the HyperLCA and the HADeLoC methods have been implemented on Xilinx system on chip (SoC) FPGA devices, while the HyperLCA has been also accelerated in embedded computing boards from NVIDIA. Since all algorithms developed in this Thesis use similar operations, it is assumed that the results and conclusions obtained by the implementations of the HyperLCA algorithm on embedded low-power GPUs (LPGPUs) could be extrapolated to the rest of the developed algorithms within this Thesis.

2. To evaluate the accuracy of the obtained results using fixed-point notation in FPGAs and floating-point notation in GPUs. The proposed set of core operations takes in consideration the hardware-design characteristics of the most commonly used computing platforms, such as FPGAs and GPUs. Accordingly, they can be easily adapted to the requirements imposed by the targeted devices and thereby, be seamlessly implemented using both fixed-point and floating-point notation. In this context, FPGA devices are in general more efficient dealing with integer operations with a close-to-hardware programming approach, while GPUs are optimised for parallel processing of floating-point operations using thousands of small cores.

3. To confirm the benefits of developing algorithmic solutions based on the same mathematical method in terms of a reduction in the execution-times, the hardware resources and the human endeavours. In this sense, FPGA-based modules that implement each of the proposed core operations have been defined using High-Level Synthesis (HLS) tools. These defined modules have been reused in the hardware implementation of the HW-LbL-FAD, the HyperLCA and the HADeLoC methods. Consequently, efforts have been focused on the interconnections among them for each target algorithm to be accelerated. In this sense, only memory buffers and custom logic that integrates and orchestrates all the components in the design have been instantiated and implemented using customized VHDL language (Very High Density Language). Therefore, it implies less time and effort during the stage of hardware acceleration since the same product can be reused for several algorithms targeting different applications.

In order to define the targeted real-time constraints, we focus on remote sensing applications where the available computational resources are limited, due to power, weight or space limitations. Concretely, we present a smart farming application where a visible-near-infrared (VNIR) hyperspectral pushbroom scanner is mounted onto an unmanned aerial vehicle (UAV), which results in a huge amount of data that needs to be managed, processed and analysed. The employed camera is able to collect information from 400 to 1000 nm using 224 spectral bands and 1024 spatial pixels per frame. It provides a maximum frame rate of 330 frames per second (FPS), what results in 144.375 MB/s (more than 8 GB per minute). In addition, this UAV carries a processing board that manages many tasks at the same time, such as, the data acquisition, the data calibration, the data storing and/or their transfers, the camera controlling and also the drone flight control. Some of the limitations of this application are related to the limited power available as well as the restrictions in the available space and weight that may be efficiently carried by the drone.

Due to these reasons, we have paid special attention to the Xilinx Zynq-7000 programmable System on Chip (SoC). We have selected this SoC because it can be found in low-cost, low-weight and compact-size development boards, such as the MicroZed^{TM},the ZedBoard^{TM} and the PYNQ boards. In addition, we have also analysed the performance of LPGPUs embedded in several NVIDIA Jetson development boards, which provide a reasonable computational power at a relatively low power consumption. Although UAVs have been consolidated as trending aerial observation platforms, their acquisition costs are

still not accessible for many end customers, not only those who want to purchase them but also those who lease their services. For this reason, we must also aim to solve the economic implications that comes along with these devices. On this basis, we have also focused on the search of less expensive computing platform alternatives that, in exchange, cannot offer the same level of both performance and functionality than other costly commercial products.

It is important to note that while experiments carried out in this work are oriented to the current necessities imposed by an application based on drones, all drawn conclusions can be extrapolated to other fields in which remotely sensed hyperspectral images (HSIs) have to be processed in real time, such as spaceborne missions that employ next-generation space-grade FPGAs.

6.2 Materials

This Section collects the data description and the most relevant characteristics of the target hardware devices used for the evaluation of the developed implementations addressed in this Chapter.

6.2.1 Reference Hyperspectral Data

The hardware implementations developed in this Thesis work for the different devices and architectures have been evaluated and compared in this Chapter using the bunch of real hyperspectral data collected by the acquisition platform described in [89] and employed in preceding Chapters 3-5. For the sake of clarity, a brief summary about these image descriptions is provided below, although we encourage the reader to see Chapter 3 to expand the details about the flight campaigns in which images were taken.

This data set was collected over multiple farming areas on the island of Gran Canaria (Spain) by a pushbroom sensor mounted on a UAV. In particular, the reference images are selected portions of some swaths within three different flight campaigns. These data cover the spectral information from 400 to 1000 nm using 160 spectral bands and consist of 825 lines height, each line comprising 1024 hyperspectral pixels with 12-bits depth. A RGB representation of these hyperspectral image portions are displayed in Figure 6.1. Images displayed in Figures 6.1 a–c were taken at a height of 72 m over the ground at a

speed-rate of 6 m/s with a camera frame-rate of 125 FPS, resulting in a ground sampling distance in line and across line of approximately 5 cm. Data shown in Figure 6.1 d was sensed in a second flight campaign performed at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm. Finally, frames exhibited in Figures 6.1 e–f were scanned at a flight height of 45 m over the ground and at a speed of 6 m/s with the hyperspectral camera capturing at 200 FPS, resulting in a ground sampling distance in line and across line of approximately 3 cm.

The aforementioned images were calibrated using a white and dark calibration to obtain reflectance values. Nonetheless, either orthorectification or georeferencing processes were not carried out for the acquired raw data. In this sense, images were built up just by placing the subsequent captured hyperspectral frames one next to the other [156]. This does not degrade the quality of the experiments carried out in this work since the tested algorithms do not use any kind of spatial information. A notable aspect of these images is the existence of some anomalous artefacts, such as some humans and concrete construction, which have been circled in blue in Figure 6.1.

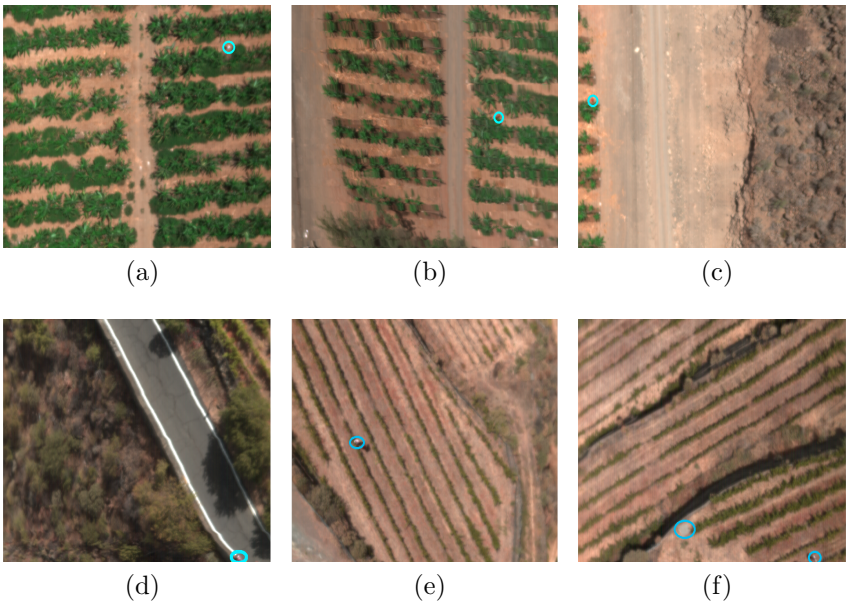


FIGURE 6.1: RGB representation of the employed test bench. Pixels enclosed in blue circles represent some anomalous spectra. (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Drone Image 4. (e) Drone Image 5. (f) Drone Image 6.

6.2.2 Targeted parallel computing devices

For the implementations described in this Chapter, different parallel computing platforms have been used. First of all, the developed FPGA-based solutions have been implemented onto a heterogeneous mid-range Zynq-7000 SoC chip (XC7Z020-clg484) from Xilinx that combines a Processor System (PS), based on a dual core ARM processor, and a Programmable Logic (PL) based on a Artix-7 FPGA architecture. This SoC can be found in the ZedBoard^{TM} development board, which has been selected because of its low-cost, low-weight and high flexibility, features that make it an interesting device to be integrated in aerial platforms, such as drones. Secondly, different NVIDIA computing boards were selected for the GPU-based implementation of the HyperLCA compressor, in particular, the Jetson TK1, the Jetson Nano, the Jetson TX2 and the Jetson Xavier-NX. These embedded system-on-modules include a LPGPU that allows parallel programming for speeding up the executed processes. These boards embed different NVIDIA architectures, going from the oldest Kepler architecture to the latest-generation Volta architecture. Due to the importance of the characteristics of the selected developer kits for the research work covered in this Chapter 6, their most relevant characteristics are described in Table 6.1.

Table 6.1: Most relevant characteristics of the NVIDIA modules Jetson TK1, Jetson Nano, Jetson TX2 and Jetson Xavier-NX, as well as the ZedBoard^{TM} development kit.

Jetson TK1	
LPGPU	GPU NVIDIA Kepler architecture with 192 CUDA cores
CPU	ARM Cortex A15 quad-core NVIDIA 4-Plus-1
Memory	2GB DDR3L, 933MHz, 64 bits bandwidth, 14.93 GB/s
Jetson Nano	
LPGPU	GPU NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores
CPU	Quad-core ARM Cortex-A57 MP Core processor
Memory	4 GB LPDDR4, 1600 MHz, 64 bits bandwidth, 25.6 GB/s
Jetson TX2	
LPGPU	GPU NVIDIA Pascal architecture with 256 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM A57/2 MB L2
Memory	8 GB LPDDR4, 128 bits bandwidth, 59.7 GB/s
Jetson Xavier-NX	
LPGPU	GPU NVIDIA Volta architecture with 384 NVIDIA CUDA cores and 48 Tensor cores
CPU	6-core NVIDIA Carmel ARM v8.2 64-bit CPU, 6 MB L2 + 4 MB L3
Memory	8 GB LPDDR4, 128 bits bandwidth, 51.2GB/s
ZedBoard^{TM} (Xilinx Zynq-7020 SoC)	
Programmable Logic (PL)	Artix-7, 85K logic cells, 53,2000 LUTs, 106,400 Flip-Flops, 612 KB BRAM (140 36Kb blocks)
Processing System (PS)	Dual-core ARM Cortex-A9 MPCore, up to 866 MHz, 256 KB on-chip memory
Memory	512 MB DDR3, 533 MHz, 1066 Mb/s

6.3 Real-time FPGA implementation of the algorithms proposed in this Thesis

An FPGA can be seen as a whiteboard for designing specialized hardware accelerators (HWaccs) by a composition of predefined memory and logic blocks that are available in the platform. Therefore, a HWacc is a set of architectural FPGA resources, connected and configured to carry out a specific task. Each of these HWaccs can be composed of smaller modules that work in parallel due to the high parallelism inherent to the FPGAs, as well as the flexibility provided by FPGA-based solutions, which allows designers to build customized solutions instead of adapting them to the hardware resources.

The definition of the HWaccs that implement the several algorithms proposed in this Thesis has been carried out by using a combination of generated HLS modules and custom glue logic in VHDL. HLS technology [226] has been used to synthesize the RTL (Register Transfer Language) code corresponding to the components that instantiate the functionality of the proposed set of core operations. RTL models are the entry points to the implementation tools that are in charge of the generation of the *bitstream*, that is, the programming file that configures the FPGA fabric to behave as it is described by the RTL. However, RTL models are low-level, time-consuming to write and verify, which leads to error-prone and lengthy development cycles. Thus, the HLS tools are key to rise productivity of such kind of developments since they are able to automatically generate the RTL models out of a specification of the functionality by means of high-level programming languages, such as C or C++ [226, 227]. This methodology focuses the effort on the design and the verification of the HWacc, as well as the exploration of the solution space that helps to speed up the search for value-added solutions.

Productivity is the strongest point of the HLS technology and one of the main reasons why hardware architects and engineers have been recently attracted to it. Nonetheless, the HLS tools that implement the synthesis process have some weaknesses. For example, despite the fact that the designer can describe a modular and hierarchical implementation of a HWacc, all sub-modules are orchestrated by a global clock due to the way the translation to the RTL from the C code is done. Another example is the rigid semantic when specifying dataflow architectures, allowing a reduced number of alternatives. This prevents the designer from obtaining optimal solutions for certain algorithms and problems [228, 229], as it was the case of the algorithmic solutions proposed in this Thesis. Therefore, to overcome the limitations of current HLS tools, hybrid solutions that combine modules

developed using VHDL and HLS-synthesized C or C++ blocks have been selected. On top of that, this approach makes it also possible to optimize the necessary resources because of the use of custom producer-consumer data exchange patterns that are not supported by the HLS tools.

In the following Sections, these aforementioned hybrid solutions are analysed and described in detail. Firstly, the HLS modules developed for the execution and implementation of the proposed set of core operations are introduced. Secondly, these HLS modules are combined with some tailor-made VHDL glue logics for the implementation of each particular targeted method, that is, the HW-LbL-FAD, the HyperLCA and the HADeLoC algorithms.

6.3.1 Descriptions of the HLS modules that implement the proposed set of core operations

As commented before, HLS technology has been used to synthesize the RTL codes corresponding to the components that instantiate the functionality of the set of core operations proposed in this Thesis. For this purpose, three HLS modules, referred to as *Avg_Cent*, *Brightness* and *Proj_Sub*, have been modelled and implemented using the aforementioned HLS tools. Instead, memory buffers and custom logic that integrate and orchestrate all the components in the design have been instantiated and implemented using VHDL language. For the sake of clarity, Figure 6.2 shows a diagram of the modules that implement each core operation and whose functionality have been described using HLS tools (light blue and white boxes) and the main glue logic and memory elements designed and instantiated using VHDL language (light red boxes, FIFOs [First in, First out] and memory elements). In this image, it has been distinguished the case of considering the stopping condition inherent to the HW-LbL-FAD and the HADeLoC methods for the extraction of the p most characteristic pixels (see Figure 6.2b).

It is important to emphasize that the tailor-made VHDL modules (light red boxes) implement an optimized dataflow. In this regard, the VHDL logic is responsible for connecting the inputs and outputs of the HLS-synthesized blocks by means of a network of selectors and buffers (i.e. FIFO and BRAM components that are generated using vendor-specific tools) that is governed by a scheduler. The scheduler is implemented as a synchronous FSM (Finite State Machine) that selectively activates/deactivates the HLS blocks and the data paths depending on the processing stage in which the algorithm is. For this

reason, diagrams displayed by Figure 6.2 will be extended for each targeted algorithm in the following Sections 6.3.2-6.3.4.

Among the main VHDL instantiated logics, Figure 6.2 highlights the importance of *SBuffer* with capacity to store a complete hyperspectral block, $\mathbf{M}_k$. The size of the *SBuffer* depends on the *BS* hyperspectral pixels within the hyperspectral block, $\mathbf{M}_k$, to be independently processed and its depth is determined at design time. The role of this *SBuffer* is to avoid the costly access to external memory, such it is the case of the double data rate (DDR) memory in the Zynq-7000 SoCs. The implementation of the *SBuffer* is based on a FIFO memory that is written and read by different producers. Since there are more than one producer and consumer for the *SBuffer*, a dedicated synchronization and control access logic has been developed in VHDL. The use of a FIFO contributes to reduce the on-chip memory resources in the FPGA fabric, being its use feasible because of the linear pattern access of the producers and consumers. However, this type of solutions would not have been possible with HLS tools because the semantic of stream-based communication between stages in a dataflow limits the number of producers and consumer to one. Also, it is not possible to exploit inter-loop parallelism as it is done in the proposed solution. Additionally, it is also important to underline the importance of the bridge buffer (*BBuffer*) that connects the *Avg_Cent* module with the others. It is a very small FIFO element whose depth is only 32 words.

Notwithstanding the foregoing, this Section focuses on the description of the HLS modules that implement the functionality of each operation within the set of cores proposed in this Thesis. Their understanding is of central importance since all algorithms targeted in the following Sections are based on these HLS modules to implement their workflow.

6.3.1.1 *Avg_Cent* HLS module: average pixel calculation and image centralization

The *Avg_Cent* HLS module implements the average pixel calculation, $\hat{\boldsymbol{\mu}}$, and the centralization of the input image blocks, $\mathbf{C}$, respectively. To this end, it is composed of two sub-modules, namely *Avg* and *Cent*, which are described below.

1. *Avg*: This sub-module computes the average pixel, $\hat{\boldsymbol{\mu}}$, of the original hyperspectral block, $\mathbf{M}_k$, and stores it in *CBuffer*, an array buffer that shares with *Cent* sub-module. During this operation, *Avg* also forwards a copy of the centroid via a

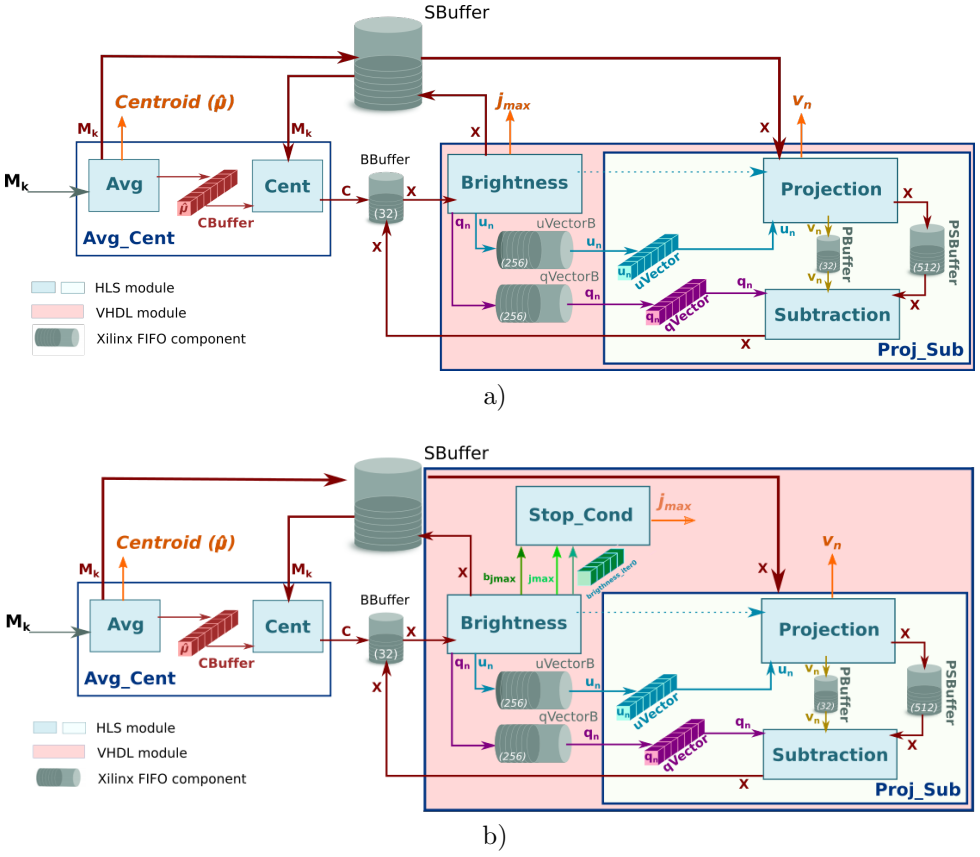


FIGURE 6.2: Overview of the benchmark HWacc that performs the proposed set of core operations. (a) Not considering stopping condition (HyperLCA algorithm). (b) Considering stopping condition (HW-LbL-FAD and HADeLoC methods). Light blue and white boxes represent modules implemented using HLS. Light red boxes, arrays and FIFOs represent the glue logic and memory elements designed and instantiated using VHDL language.

dedicated port (orange array $\hat{\boldsymbol{\mu}}$ in Figure 6.2), which will be used in particular by the *Entropy Coding* stage of the HyperLCA and the HADeLoC algorithms, as it will later further explained in Sections 6.3.2 and 6.3.4. At the same time, the *Avg* sub-module writes all the pixels within $\mathbf{M}_k$ into the *SBuffer*. A copy of the original hyperspectral block, $\mathbf{M}_k$, will be available once *Avg* finishes, ready to be consumed as a stream by *Cent*, which reduces the latency.

Figure 6.3 shows in detail the functioning of the *Avg* stage. The main problem of this stage is the way in which the hyperspectral data is stored. In our case, the hyperspectral block, $\mathbf{M}_k$, is ordered by the bands that make up a hyperspectral pixel. However, to obtain the centroid, $\hat{\boldsymbol{\mu}}$, the hyperspectral block must be read by bands (in-width reading) instead of by pixels (in-depth reading). In this sense, we introduce an optimization that handles the data as it is received (in-depth), avoiding

the reordering of the data to maintain a stream-like processing. This optimization consists of an accumulate vector, whose depth is equal to the number of bands that stores partial results of the summation for each band, i.e., the first position of this vector contains the partial results of the first band, the second position the partial results of the second band and so on.

2. *Cent*: This sub-module reads the original hyperspectral block, $\mathbf{M}_k$, from the *SBuffer* to centralize it, obtaining the centralized version of the input data, $\mathbf{C}$. This operation consists of subtracting the average pixel, $\hat{\boldsymbol{\mu}}$, calculated in the previous stage, from each hyperspectral pixel of the block. Figure 6.4 shows this process, highlighting the elements that are involved in the centralization of the first hyperspectral pixel. Thus, the *Cent* block reads the centroid, $\hat{\boldsymbol{\mu}}$, which is stored in the *CBuffer*, as many times as hyperspectral pixels have the original block (i.e., *BS* times in the example illustrated in Figure 6.4). Therefore, *CBuffer* is an addressable buffer that permanently stores the centroid of the current hyperspectral block that is being processed. The result of this stage is written into the *BBuffer* FIFO, which makes unnecessary an additional copy of the centralized image, $\mathbf{C}$. As soon as the centralized components of the hyperspectral pixels are computed, the data is ready at the input of the *Brightness* module and, therefore, it can start to perform its operations without waiting for the block to be completely centralized.

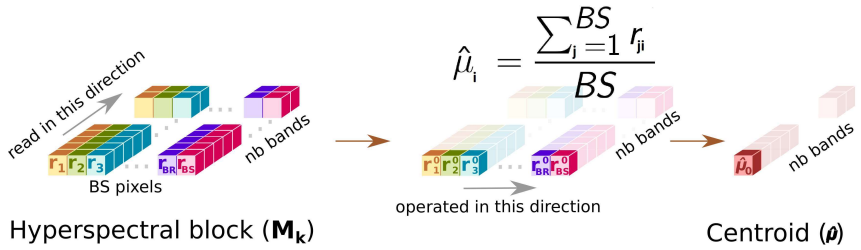


FIGURE 6.3: Overview of the *Avg* HLS sub-module that implement the average pixel, $\hat{\boldsymbol{\mu}}$, calculation.

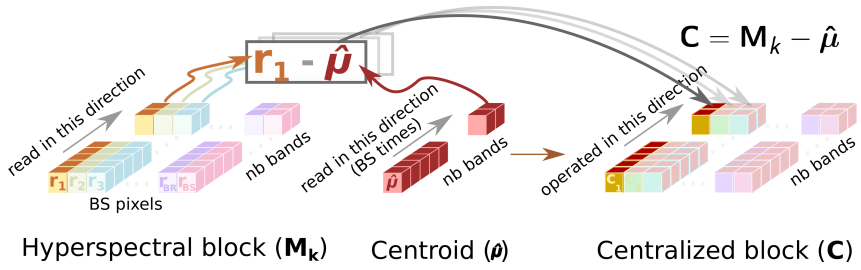


FIGURE 6.4: Overview of the *Cent* HLS sub-module that manages the centralization of the image, $\mathbf{C}$.

6.3.1.2 *Brightness* HLS module: brightness pixel calculation

The *Brightness* module calculates the brightness of each image pixel, b_j, and looks for the index of the image pixel with the maximum brightness, j_{max}. In this sense, *Brightness* module has been optimized to achieve a dataflow behaviour that takes the same time regardless of the location of the brightest hyperspectral pixel. The *Brightness* module starts working as soon as there are data in the *BBuffer*. In this sense, this module works in parallel with the rest of the system; the input of the *Brightness* module is the output of the *Cent* module in the first iteration, that is, the centralized image, $\mathbf{C}$, while the input for all other iterations is the output of the *Subtraction* sub-module, depicted as $\mathbf{X}$ for the sake of clarity. In addition, this module also estimates the $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors in each iteration, which are later used by the *Proj_Sub* module. For this reason, both modules are connected by two FIFO components (*uVectorB* and *qVectorB* in Figure 6.2) using customized VHDL code to link them. Additionally, Figure 6.5 shows in the detail the functioning of the *Brightness* module.

The *Brightness* HLS module is designed to read in order the hyperspectral pixels of the image block from the *BBuffer* ($\mathbf{C}$ or $\mathbf{X}$ depending on the loop iteration) for calculating its brightness, b_j, in the *Brightness_calc* sub-module employing a loop unrolling strategy. The *Brightness_calc* sub-module also makes a copy of the hyperspectral pixel in an internal ping-pong buffer, *h_pixel*, and in *SBuffer*. A ping-pong buffer is a double buffer that is used here to contain the current hyperspectral pixel whose brightness is being calculated in one of the available buffers. If the actual brightness is greater than the previous calculated ones, the *Brightness_calc* sub-module saves the subsequent hyperspectral pixel to be processed in the other buffer. In this way, one of the available buffer within the *h_pixel* ping-pong buffer always saves the pixel with the maximum brightness, $\mathbf{r}_{j_{max}}$. This module also returns the index of the brightest pixel, j_{max}, the maximum brightness value, $b_{j_{max}}$, and an array with BS components, **brightness_iter0**, that contains the original pixel brightnesses within $\mathbf{C}$. These outputs are later used by the module that performs the stopping condition inherent to the HW-LbL-FAD and the HADeLoC algorithms (see Figure 6.5b). Nonetheless, this stopping condition is discarded by the HyperLCA compressor. For this reason, another approach is only to forward a copy of the j_{max} via a dedicated port (orange array j_{max} in Figures 6.5a and 6.2a), which will be used in particular by the *Entropy Coding* stage of the HyperLCA compressor, as it will later further explained in Section 6.3.2.

Once that the brightness of all image pixels has been calculated, the orthogonal projection vectors $\mathbf{q}_n$ and $\mathbf{u}_n$ are accordingly obtained from the brightest pixel. For doing so, the *Build_quVectors* sub-module reads $\mathbf{r}_{j_{max}}$ from the corresponding buffer within the *h_pixel* ping-pong buffer. Then, the *Build_quVectors* sub-module writes both $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors in two single FIFOs, that is *qVectorB* and *uVectorB*, respectively. Furthermore, the contents of these FIFOs are copied in *qVector* and *uVector* arrays in order to get a double space memory that does not deadlock the system and allows the *Proj_Sub* module to read the orthogonal projection vectors $\mathbf{q}_n$ and $\mathbf{u}_n$ BS times to obtain the projection vector, $\mathbf{v}_n$.

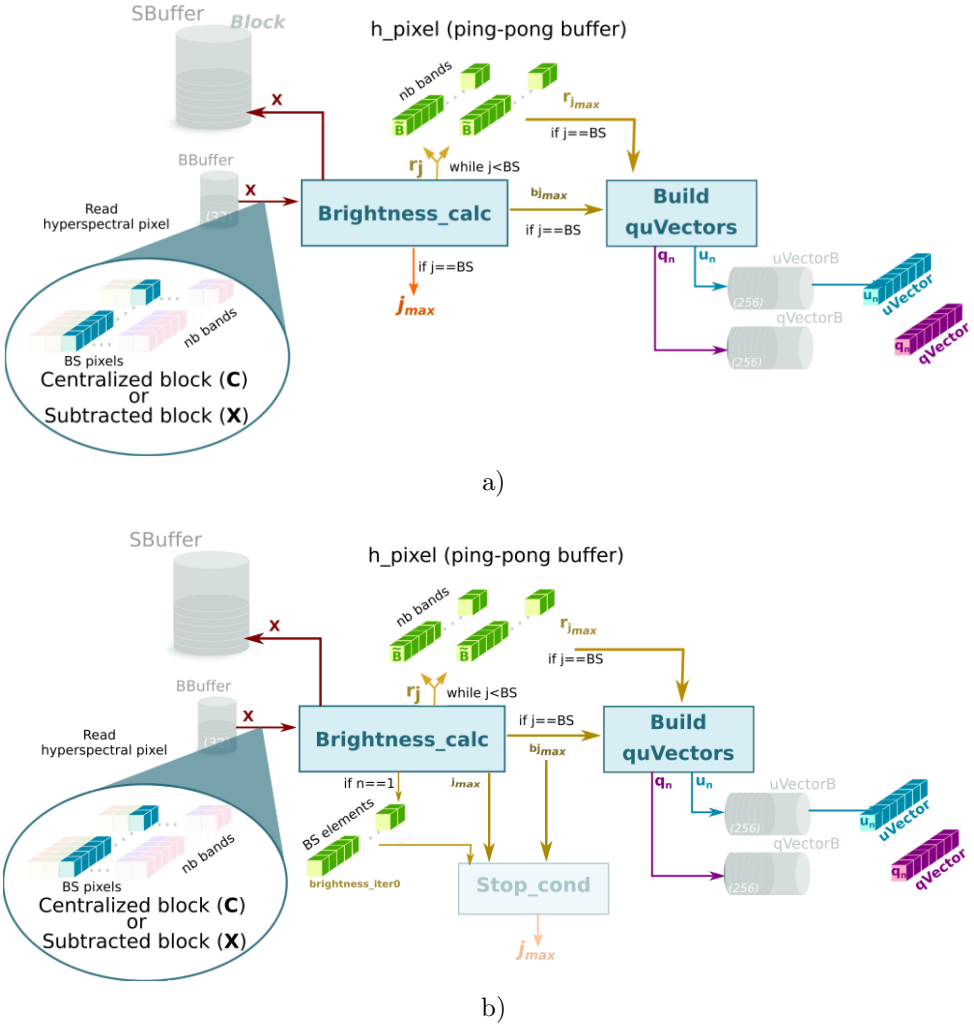


FIGURE 6.5: Overview of the *Brightness* HLS module. (a) Not considering stopping condition (HyperLCA algorithm). (b) Considering stopping condition (HW-LbL-FAD and HADeLoC methods).

6.3.1.3 *Proj_Sub* HLS module: projection vector calculation and spectral information subtraction

The *Proj_Sub* module implements the projection vector calculation, $\mathbf{v}_n$, and the subtraction of the spectral information spanned by the selected pixel, $\mathbf{q}_n$, from $\mathbf{X}$. To this end, this module includes two sub-modules, namely *Projection* and *Subtraction*. Although these sub-modules are represented by separate boxes in Figure 6.2, it must be mentioned that both perform their computations in parallel. Figure 6.6 shows a detailed representation of the functioning of these aforementioned sub-modules. Firstly, each hyperspectral pixel of the block is read by the *Projection* sub-module from *SBuffer* to obtain the projection image vector, $\mathbf{v}_n$. At the same time, the hyperspectral pixel is written in *PSBuffer*, which can store two hyperspectral pixels. It is because the *Subtraction* stage begins right after the projection of the first hyperspectral pixel is ready, i.e. the execution of both the *Projection* and *Subtraction* sub-modules are shifted by one pixel, as indeed Figure 6.6 shows. While pixel $\mathbf{r}_1$ is being consumed by the *Subtraction* sub-module, pixel $\mathbf{r}_2$ is being written in *PSBuffer*. During the projection of the second hyperspectral pixel, $\mathbf{r}_2$, the subtraction of the first one, $\mathbf{r}_1$, can be performed since all the input operands, including the projection $\mathbf{v}_n$, are available. The output of the *Projection* sub-module is the projection image vector, $\mathbf{v}_n$, which is forwarded via a dedicated port (orange array $\mathbf{v}_n$ in Figure 6.2), which will be used in particular by the *Entropy Coding* stage of the HyperLCA and the HADeLOC methods, as it will be further explained in Sections 6.3.2 and 6.3.4. Hence, the *Projection* module also addresses the *Scaling of V vectors* step inherent to these algorithms.

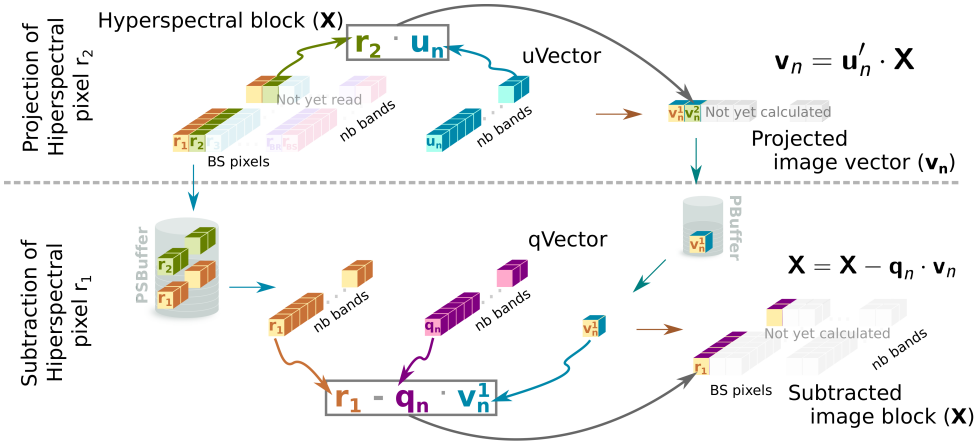


FIGURE 6.6: Example of the *Projection* and *Subtraction* sub-modules that implement the projection vector calculation, $\mathbf{v}_n$, and the spectral information retained for the next iterations, respectively.

6.3.1.4 *Stop_cond* HLS module: stopping condition inherent to the HW-LbL-FAD and the HADeLoC algorithms

Apart from the aforementioned modules, one additional module is also considered related with the anomaly detection process carried out by the HW-LbL-FAD and the HADeLoC methods. This module is the *Stop_cond* shown in Figure 6.2b. The *Stop_cond* module is in charge of the computation of the ratio between the actual maximum brightness, $b_{j_{max}}$, which is calculated in the *Brightness* module, and the original brightness of the selected pixel, $\mathbf{r}_{j_{max}}$, in $\mathbf{C}$. This is indeed calculated in the first iteration of the set of core operations for the extraction of the first characteristic pixel, $\mathbf{e}_1$. For this reason, the *Brightness* module contemplates the **brightness_iter0** array, which eases the calculation of the stopping condition addressed in the *Stop_cond* module in question. In addition, when this module is executed, it is also responsible to forward a copy of the j_{max} via a dedicated port (orange array j_{max} in Figures 6.2b and 6.5b) instead of the *Brightness* module. This j_{max} index will be used in particular by the *Entropy Coding* stage of the HADeLoC solution, as it will be further explained in Section 6.3.4.

6.3.1.5 Other considerations about the FPGA-based implementation of the proposed set of core operations

So far, it has been described the inner architecture of a HWacc that only performs a computational operation over a single band component of a hyperspectral pixel. Nonetheless, it is not sufficient to reach the minimum targeted requirements imposed by the specific application. For this reason, the described HWacc has been modified to increase the number of bands that are processed in parallel, which are referred to as processing element (PE) along the remainder of this Chapter. Thus, the described HWacc turns from a single PE to multiple PEs. This fact opens two new challenges. The first challenge is to increase the width of the input and output ports of the modules, in accordance with the number of bands that would be processed in parallel. It must be mentioned that it is technologically possible because the HLS-based solutions allow designers to build their own data types. For example, if a band component of a hyperspectral pixel is represented by an unsigned integer of 16-bits, we could define our own data types consisting of an unsigned integer of 160-bits packing ten bands of a hyperspectral pixel (see Figure 6.7). The second challenge has to do with the strategy to process the data in parallel. In this regard, a solution based on the map-reduce programming model has been followed [230].

Figure 6.7 shows an example of the improvements applied to the *Cent* stage following the above-mentioned optimizations. The input of this stage is the hyperspectral block, $\mathbf{M}_k$, and the average pixel, $\hat{\boldsymbol{\mu}}$, which are read in blocks of N bands. The example assumes that the block is composed of ten bands and uses an user-defined data type, specifically an unsigned integer of 160-bits (10 bands by 16-bits to represent each band). Then, both blocks are broken down into the individual components that feed the PEs in an orderly fashion. This process is also known as map phase in the map-reduce programming model [230]. It must be mentioned that the design needs as many PEs as number of divisions made in the block. Once the PEs have performed the assigned computational operation, the reduce phase of the map-reduce model is executed. For *Cent* sub-module, this stage consists of gathering in a block the partial results produced by each one of the PEs. Thus, a new block of N-bands is built, which in turn is part of the centralized block, $\mathbf{C}$, which is also the output of the *Cent* sub-module.

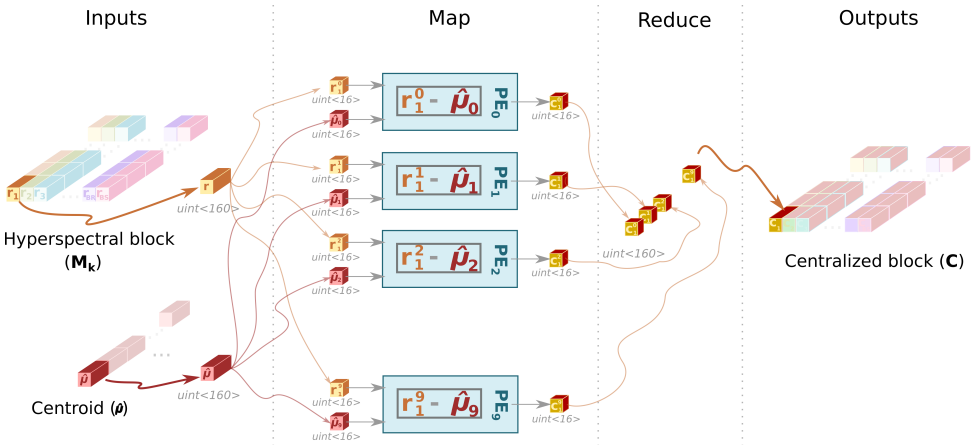


FIGURE 6.7: Map-reduce programming model and data packaging on the *Cent* module.

6.3.2 FPGA-based implementation of the HyperLCA lossy compressor

As it was further described in Chapter 4, the HyperLCA algorithm is mainly structured in four computing stages: the *Initialization*, the *HyperLCA Transform*, the *Preprocessing* and the *Entropy Coding*. As a remainder, these algorithmic stages address the following operations:

1. *Initialization*: this stage is just in charge of the estimation of the number of selected pixels, p, according to some input parameters known at design time (the minimum

desired compression ratio (CR), the block size (BS) and the number of bits (N_{bits}) used for scaling the projection vectors, $\mathbf{V}$.

2. *HyperLCA Transform*: the *HyperLCA Transform* performs the spectral transformation that results in the uncorrelated and compressed image. Indeed, it involves the most computationally demanding operations, besides, it actually executes the proposed set of core operations to meet the aforementioned purpose.

3. *Preprocessing*: the *HyperLCA Transform* outputs ($\hat{\boldsymbol{\mu}}$, $\mathbf{E}$ and $\mathbf{V}$) are adapted in this stage for being efficiently entropy coded in the subsequent *Entropy Coding* stage. To this end, the projection vectors within $\mathbf{V}$ are firstly scaled to be represented using integers in the *Scaling of V vectors* step. Secondly, the aforementioned listed outputs are individually lossless preprocessed and transformed in the *HyperLCA Error Mapping* stage to be exclusively composed of positive integer values closer to zero than the original ones.

4. *Entropy Coding*: this stage is in charge of the entropy encoding of the *HyperLCA Transform* outputs. For this purpose, it follows a lossless entropy-coding strategy based on the Golomb–Rice algorithm [212].

Nonetheless, the aforementioned computing stages were defined from an algorithmic point of view. Therefore, they have been redefined in this Chapter in order to be better adapted to devices with a high degree of fine-grain parallelism, such as FPGAs. For instance, the number of selected pixels, p, estimated in the *Initialization* stage depends on the configuration of the input parameters and hence, it can be fixed at design time. Indeed, several hardware components, such as internal memories or FIFOs, must be configured with the appropriate size, which is also dependant on this parameter. For these reasons, the HyperLCA algorithm has been structured in two main stages from an implementation point of view: the *HyperLCA Transform* and the *HyperLCA Entropy Coder*. Therefore, the FPGA-based implementation of the HyperLCA algorithm involves the definition of two HWacc modules that actually address these two algorithm stages and also the two steps within the *Preprocessing* stage. Their execution is managed in parallel, which improves the performance of the designed implementation. These HWacc modules are further analysed in below Sections 6.3.2.1 and 6.3.2.2.

6.3.2.1 *HyperLCA Transform* HWacc

The *HyperLCA Transform* module involves the most computationally demanding operations, besides, it actually executes the proposed set of core operations. For this reason, the implementation of this HWacc module has been tackled exactly as it was described in preceding Section 6.3.1 and, in particular, as shows Figure 6.2a. Therefore, the *HyperLCA Transform* module also performs the computation of the *Scaling of V vectors* step within the *Preprocessing* stage. However, the description made in above Section 6.3.1 only highlights how the operations are performed in parallel to make the most of such technology. Apart from this, it has to be taken into account other considerations, as those discussed in the following paragraph.

In this regard, the adopted hybrid solution based on HLS and hand-written VHDL codes has also spotted specific synchronization scenarios that bring to life an efficient dataflow. On this basis, it is a key point the use of optimal sized FIFOs to interconnect the HLS modules that execute each of the proposed core operations. For example, while the *Brightness* module is filling the *SBuffer*, the *Projection* sub-module is draining it and, at the same time, it supplies to the *Subtraction* sub-module with the same data read from *SBuffer*. Finally, the *Subtraction* sub-module feeds back the *Brightness* sub-module through the *BBuffer* FIFO. The *Brightness* sub-module fills, in turn, the *SBuffer* with the same data, closing the circle, which is indeed repeated p times.

6.3.2.2 *HyperLCA Entropy Coder* HWacc

The *HyperLCA Entropy Coder* is the second HWacc module developed for the implementation of the HyperLCA algorithm in FPGA-based systems. The coder module teams up with the *HyperLCA Transform* HWacc to perform in parallel the CCSDS prediction error mapping [211] and the Golomb–Rice [212] entropy-coding algorithms as the different vectors are received from the *HyperLCA Transform* block. It means that the operation of the transform and the coder blocks overlaps in time. In this regard, the *HyperLCA transform* module generates the centroid, $\hat{\boldsymbol{\mu}}$, the indexes of the selected pixels, j_{max}, and the projection vectors, $\mathbf{v}_n$, for an input hyperspectral block, $\mathbf{M}_k$, which feeds the inputs of the *HyperLCA Entropy Coder*. These arrays are consumed as they are received, reducing the need for large intermediate buffers. Furthermore, the coder takes approximately half of the time than the *HyperLCA Transform* needs to generate each vector for the maximum number of PEs. Therefore, a contention situation is not taking place, reducing the

pressure over the FIFOs that connect both blocks and, therefore, requiring less space for these communication channels.

Figure 6.8 sketches the internal structure of the *HyperLCA Entropy Coder* that has been modelled entirely using Vivado HLS. It is a dataflow architecture comprising three steps. During the first step, the *Error Mapping* step and the *Entropy coding* stage are performed on all input vectors by the *Coding Command generator* module. The output of this step is a sequence of commands that are subsequently interpreted by the *Bitstream Generator* module. The generation of the bitstream was extracted from the entropy-coding original functionality, which enabled a more efficient implementation of the latter since could be re-written as a perfect loop. Therefore, Vivado HLS was able to generate a pipelined datapath with the minimum initiation internal (II = 1). To do this, the *Bitstream Generator* module is continuously reading the *cmd_queue* FIFO for a new command to be processed. A command contains the operation (unary or binary coding) as well as the word (quotient or reminder) to be coded, and the number of bits to generate. Unary and binary coding functions simply iterate over the word to be coded and produces a sequence of bits that corresponds to the compressed hyperspectral data. Finally, the third step packs the compressed bitstream in words and writes them to memory. For this implementation, the width of the memory word is 64 bits, the maximum allowed by the *AXI Master* interface port for the Zynq-7020 SoC. The *Bitstream Packer* module instantiates a small buffer (64 words) that is flushed to DDR memory once it has been filled. This way, the average memory access cycles per word is optimized by means of the use of burst requests.

As mentioned above, the *HyperLCA Transform* block feeds the coder with the indexes of the selected pixels, j_{max}. Nonetheless, the *HyperLCA Entropy Coder* has indeed to codify the selected hyperspectral pixels, $\mathbf{e}_n$, not their indexes, j_{max}. Hence, the *HyperLCA Coder* is the responsible to obtain each hyperspectral pixel, $\mathbf{e}_n$, from the external memory in which is stored the HSI to build the vector of most different hyperspectral pixels, $\mathbf{E}$. This is the role played by the *Pixel Reader* module. As in the case of the *Bitstream Packer* step, the *Pixel Reader* uses a local buffer and issue burst requests to read the bands in the minimum number of cycles.

Nonetheless, the real challenge of the *Entropy Coder* module implementation is to write a C++ HLS model that is consistent through the whole design, verification and implementation processes. For this purpose, side-channel information is embedded in the *cmd_queue* and *compressed_stream* FIFOs that connect the different stages of the coder. This information is used by the different modules to reset their internal states or stop and resume

their operations (i.e., special commands to be interpreted by the *Bitstream Generator* or a flush signal as input to the *Bitstream Packer* module). This way, it is possible to integrate under a single HLS design all the functionality of the coder, which simplifies and speeds up the design process.

As mentioned before, decoupling the generation of the compressed bitstream from the entropy-coding logic, leads to a more efficient implementation of the latter by the HLS synthesis tool. In the *Coding Command Generator*, a simple logic that controls the encoding of each input vector plus a header is implemented. It is an iterative process that performs the error mapping and the entropy coding over the centroid, and p times over the extracted pixels and projections vectors. The bulk of this process is, thus, the encoding algorithm. The encoding is delegated in another module that implements an internal dataflow itself. In this way, it is possible to reduce the interval between two encoding operations. As can be seen in Figure 6.8, the *Error Mapping* and the *Entropy-Coding* sub-modules communicates through a ping-pong buffer for the *Mapped Vector*.

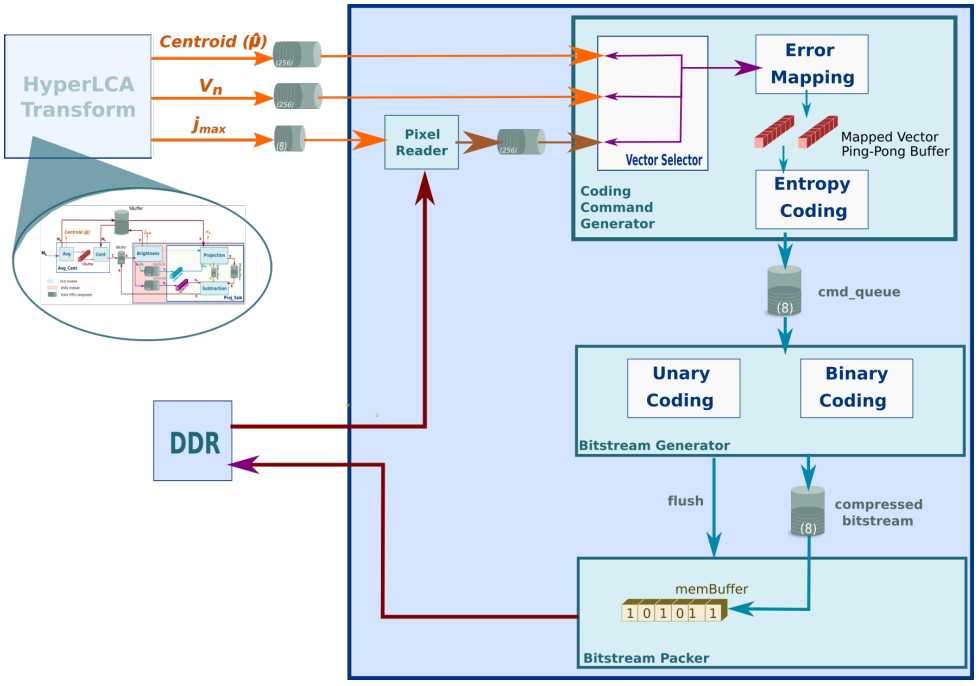


FIGURE 6.8: Overview of the *HyperLCA Entropy Coder* HWacc.

6.3.3 FPGA-based implementation of the HW-LbL-FAD algorithm for the detection of anomalous spectra

The HW-LbL-FAD algorithm entirely employs the proposed set of core operations for the detection of anomalous spectra and hence, the HLS modules described in preceding Section 6.3.1. Nonetheless, the main challenge of the HW-LbL-FAD implementation on FPGA-based devices lies in the scheduler that governs the different computing stages inherent to this algorithm. To overcome this issue, tailor-made VHDL logic has been defined to implement an optimized dataflow. The VHDL logic is responsible for connecting the inputs and outputs of the HLS-synthesized blocks by means of a network of selectors and buffers (i.e. FIFO and BRAM components that are generated using vendor-specific tools) that is governed by the scheduler. The scheduler is implemented as a synchronous FSM that selectively activates/deactivates the HLS blocks and the data paths depending on the processing stage in which the algorithm is.

Figure 6.9 shows a diagram of the modules implemented using the HLS tools (light blue and white boxes) and the main glue logics and memory elements designed and instantiated using VHDL language (light red boxes, FIFOs and memory elements). As it can be seen, few additional resources are included in this diagram compared with those displayed in Figure 6.2b. Small coloured squares indicate the algorithm stage or stages in which these resources are operative. The same color code is used to tag the data sources in the selectors (trapeziums in Figure 6.9) for the different steps of the algorithm. This strategy not only promotes the optimization of the available hardware resources for the targeted application but it also permits that one memory buffer can be shared by more than one producer and consumer, such as in the case of the *SBuffer*. For the sake of clarity, further details are provided about the designed scheduler according to the different algorithm stages:

1. *Stage 1*: In this stage, the first n_f image blocks, $\mathbf{M}_k$, are independently analysed in search of the p most different pixels, $\mathbf{E}$, within them. In this sense, the set of core operations is executed in the same way as by the *HyperLCA Transform* described in previous Section 6.3.2. The major difference is the estimation of the number of iterations, n, to be carried out for extracting the p reference vectors. Regarding the HyperLCA compressor, p is known beforehand since it is computed in the *Initialization* stage as a function of some input parameters. On the contrary, a stop condition is checked in every iteration in the case of the HW-LbL-FAD algorithm. For this reason, the *Stop_cond* HLS module is also included in the schematic diagram

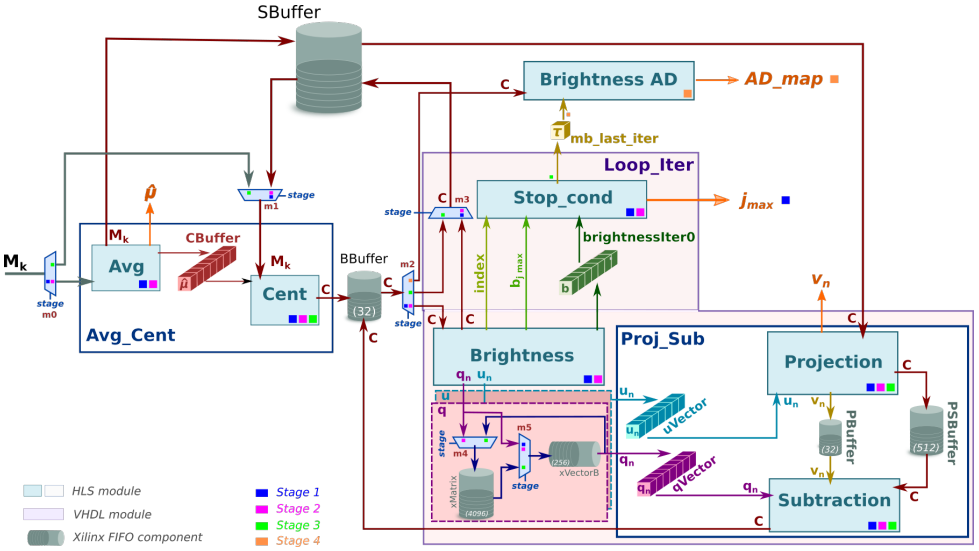


FIGURE 6.9: Overview of the HW-LbL-FAD HWacc.

shown in Figures 6.2b and 6.9. In the designed FSM, this HLS module does not act as a blocking module that alters the implemented dataflow. In this regard, the *Proj_sub* module stars once the pixel with the maximum brightness, $\mathbf{r}_{jmax}$, is found by the *Brightness* module. At the same time, the *Stop_cond* HLS module computes the HW-LbL-FAD stopping condition in order to determine if other iteration has to be conducted. Otherwise, *BBuffer* and *SBuffer* are drained to be ready for the allocation of a new received frame.

2. *Stage 2*: In this stage, the subspace of orthogonal vectors, $\mathbf{Q}$ and $\mathbf{U}$, that model the background distribution is estimated using the set of core operations organized in the same way as it was defined in *Stage 1*. Nonetheless, the set of vectors, $\mathbf{E}$, above selected is employed as input pixel block, $\mathbf{B}^*$, instead of $\mathbf{M}_k$. For this reason, it is included the first multiplexor (*m0*) placed in the left side of Figure 6.9. Nonetheless, *SBuffer* depth is configured according to the number of *BS* elements within each $\mathbf{M}_k$. In this sense, the number of pixels within $\mathbf{B}^*$ could be smaller than *BS*. For dealing with this issue, elements within $\mathbf{B}^*$ are repeated until fulfilling the required *BS* pixels to fill the *SBuffer*. It is demonstrated that neither the average pixel estimation, $\hat{\boldsymbol{\mu}}$, nor the selected pixels, $\mathbf{r}_{jmax}$, are affected. Another aspect to be highlighted is the estimation and storage of the set of orthogonal $\mathbf{Q}$ and $\mathbf{U}$ vectors since they will be later used by the subsequent *Stage 3*. For this purpose, the circular buffers *qMatrix* and *uMatrix* are used for storing these vectors respectively (*xMatrix* in Figure 6.9).

3. *Stage 3*: Once the background pattern is modelled, the next HW-LbL-FAD stages deal with the detection of the anomalous pixels within the following sensed HSI blocks, $\mathbf{M}_k$. These stages slightly differ from the previous two. In this regard, the average pixel, $\hat{\boldsymbol{\mu}}$, estimated in the *Stage 2* is used for centralizing the input image block, $\mathbf{M}_k$. Consequently, the *Avg* sub-module is skipped using the multiplexor *m1* and the input image block, $\mathbf{M}_k$, is directly centralized in the *Cent* sub-module using the average pixel computed in the *Stage 2* and stored in the *CBuffer* array. Therefore, the *SBuffer* is filled in this case trough the *BBuffer*. Additionally, the *Brightness* operation block is not required in the actual stage and hence, the definition of the selector *m3*.

   Subsequently, the information spanned by the $\mathbf{Q}$ and $\mathbf{U}$ vectors computed in *Stage 2* is subtracted from pixels within $\mathbf{M}_k$. To do this, the *Projection* and *Subtraction* operation blocks are repeated p times, that is, the number of vectors contained in $\mathbf{Q}$ or $\mathbf{U}$, which are also stored in *qMatrix* and *uMatrix* buffers, respectively. Therefore, the *Projection* and *Subtraction* sub-modules read in each iteration one pixel $\mathbf{q}_n$ and $\mathbf{u}_n$ from *qVectorB* and *uVectorB* buffers (*xVectorB* in Figure 6.9). Nonetheless, these vectors have to be still retained for next input images, $\mathbf{M}_k$. For this reason, *xMatrix* buffers are filled at the same time with the vector stored in *xVectorB* buffers, obtaining a circular buffer behaviour. This is achieved by the definition of the *m4* and *m5* selectors.

4. *Stage 4:* Finally, the brightness of the remaining spectral information in each image pixel is used for identifying the anomalous entities in the *Stage 4*. For this reason, it is defined an additional HLS module, named *Brightness AD*. The *Brightness AD* calculates the brightness of each pixel within $\mathbf{X}$ in a similar way as the *Brightness_calc* sub-module. In this sense, this HLS module is also designed to read in order the hyperspectral pixels of the image block from the *BBuffer* (see *m2* multiplexor) for calculating their brightness, b_j, employing also a loop unrolling strategy. Nonetheless, since the brightness calculation is just addressed at this point, non the search of $\mathbf{r}_{j_{max}}$ as well, the internal ping-pong buffer is not included in the definition of the *Brightness AD* module. The output of the *Brightness AD* module is a binary map where anomalies are marked as class 1 and the background pixels as class 0 (orange array $\mathbf{AD_{map}}$ in Figure 6.9). For doing so, the brightness of the last pixel selected in the *Stage 2*, τ, is used as a threshold (see *mb_last_iter* register in Figure 6.9). This parameter is also an output of the *Stop_cond* HLS module.

Finally, it is also important to mention that dedicated ports $\hat{\boldsymbol{\mu}}$ and $\mathbf{v}_n$ displayed in Figure 6.9 are not actually required by the correct implementation of the HW-LbL-FAD algorithm. Nonetheless, they have been kept in order to reuse the defined HLS modules for the other algorithms addressed in this Thesis.

6.3.4 FPGA-based implementation of the HADeLoC solution for the simultaneous execution of the anomaly detection process and the lossy compression of HSIs

The FPGA-based implementation of the HADeLoC proposal for the simultaneous execution of the anomaly detection process and the lossy compression of HSIs was straightforward building on the above described HWaccs that implement the HW-LbL-FAD detector and the *Entropy Coding* stage of the HyperLCA compressor. As it was analysed in Chapter 5, the workflow inherent to the anomaly detection process largely follows the same scheme as the HW-LbL-FAD method. Therefore, the strategy follows for the implementation of the HW-LbL-FAD in FPGA-based systems could be reused in its totality, in addition to some minor changes introduced in the FSM. Regarding the compression process, it has been subjected to the methodological changes introduced by the anomaly detection one. For this reason, the generated bitstream representative of the compressed data slightly differs from the one defined by the original HyperLCA algorithm. Nonetheless, irrespective of the vectors to to be codified, the HADeLoC algorithm also uses the Golomb-Rice encoding conducted by the HyperLCA compressor for the individual coding of the respective vectors. Consequently, the workflow follows by the *HyperLCA Entropy Coder* HWacc described in Section 6.3.2.2 can be also replicated for the HADeLoC issue.

In general terms, the HADeLoC method behaves similarly as the HyperLCA compressor in the way that there are two HWaccs that work in conjunction. The first one indeed executes the set of core operations proposed in this Thesis in a similar way as described in Section 6.3.3. This module manages the computation of the data vectors that make up the bitstream that will be later handled by the codification stage. Therefore, the *HyperLCA Entropy Coder* HWacc module teams up with the aforementioned module to perform in parallel the CCSDS prediction error mapping and the Golomb–Rice entropy-coding algorithms on the different vectors received as outputs of the preceding HWacc. It means that the operations involved in both modules overlap in time.

Although the HADeLoC method is almost a faithful mirror of the HW-LbL-FAD algorithm, it includes an additional computing stage (*Stage 5*) that requires to extend the FSM described in Section 6.3.3 for the HW-LbL-FAD algorithm to include this fifth stage. Figure 6.10 displays a diagram of the hybrid solution for the FPGA implementation of the HADeLoC solution. As it can be seen, few additional glue logics are instantiated using VHDL language compared with the module displayed in Figure 6.9. In this sense, the search of anomalous spectra addressed by the HW-LbL-FAD method in each image block, $\mathbf{M}_k$, finishes once the binary map output is obtained in the *BrightnessAD* module in the *Stage 4*. In this point, it was not necessary to fill the *SBuffer* with the data, $\mathbf{X}$, obtained in the last iteration of the *Proj_Sub* module, since they are processed in the *Brightness AD* module and then discarded pending the reception of a new image block, $\mathbf{M}_k$, to be analysed. Nonetheless, these data have to be retained in the case of the HADeLoC algorithm for the computation of this additional *Stage 5*.

As a reminder, the anomalous pixels detected in the *Stage 4* could not be well reconstructed using the transmitted $\mathbf{V}$ vectors since they just retain the spectral information representative of the background pattern. For this reason, when anomalous spectra are identified, the group of operations *Brightness - Projection - Subtraction - Brightness AD* is repeated until *Brightness AD* does not extract any more anomalous pixels. Therefore, apart from $\mathbf{V}$ vectors estimated in *Stage 3*, the new selected pixels with the highest brightness, $\mathbf{e}_n$, and its corresponding projection vectors, $\mathbf{v}_n$, are also included in the bitstream to be transmitted for each image block, $\mathbf{M}_k$. For this reason, unlike Figure 6.9, the *Brightness AD* module must transmit the image block, $\mathbf{X}$, to *SBuffer* in case that anomalous pixels are detected and the *Stage 5* has to be launched. If so, then the *BBuffer* reads the data from *SBuffer* to bridge them with the *Brightness* module.

It is also important to mention that the *Brightness AD* module is in charge of indicating the number of times that the group of operations *Brightness - Projection - Subtraction - Brightness AD* is repeated in the *Stage 5* instead of the *Stop_cond* module. Nevertheless, the latter is actually which forwards a copy of the indexes of the selected pixels via a dedicated port (orange array j_{max}) in Figure 6.10, which is latter read by the *HyperLCA Entropy Coder* HWacc. Consequently, the operations involved in the *Stop_cond* module are also executed in background although just for the forwarding of the j_{max} output. This was done in this way merely for reusing the initial HLS definition of this aforementioned module.

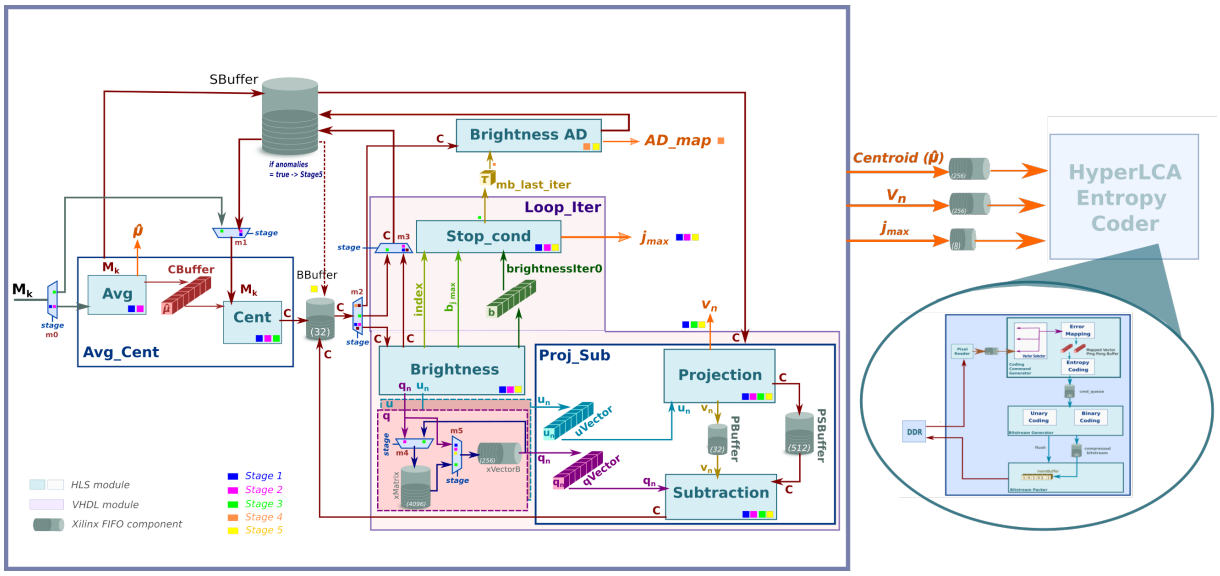


FIGURE 6.10: Overview of the HADeLoC HWacc.

6.3.5 Experimental results

Several experiments have been carried out in order to evaluate the performance of the HWaccs described in preceding Sections for the FPGA-based implementations of the HyperLCA compressor, the HW-LbL-FAD detector and the HADeLoC joint solution. In this regard, the conducted analysis has been done in terms of two critical factors in this kind of implementations, that is, the resource utilization and the maximum frame rates reached by the targeted processes, which is defined as the number of image blocks composed of BS hyperspectral pixels that can be processed in one second.

Concerning the data width of the designed architectures, it depends on two parameters: the number of hyperspectral bands that can be processed in parallel (PEs) and the size of the image block to be compressed (BS and the number of bits used to save each image element). Consequently, the benchmarking made in following Sections have been addressed for different PEs settings. It is worth mentioning that the number of instantiated PEs must be a divisor of the number of hyperspectral bands in order to simplify the design of the datapath logic. Additionally, only settings of $BS = 1024$ have been evaluated since the image acquisition system captures 1024 spatial pixels per scanned cross-track line, as well as, the HyperLCA compressor and the HW-LbL-FAD algorithm get the best results with this configuration.

Finally, the feasibility of the proposed set of core operations for its implementation using integer arithmetic and fixed-point notation with different levels of precision was demonstrated in preceding Chapters. For this reason, the *Int16-rd* version has been selected for the targeted HWaccs under evaluation, due to its good behaviour in terms on the quality and the precision of the obtained results and the resource savings it brings. For the *Int16-rd* version, a band is represented with an *unsigned short int* which turns into 16-bit words in memory. On the contrary, the *Int32* version, for instance, demands almost the double of memory resources and internal buffers (such as the *SBuffer*), since *unsigned int* data type is used in the model definition.

6.3.5.1 Evaluation of the HyperLCA Hardware Accelerator

This section discloses the discussion made about the developed HyperLCA HWacc with the purpose of evaluating the goodness of the proposed design for the execution of this algorithmic solution in FPGA-based systems. The architecture of the HWacc is divided into two blocks, the *HyperLCA Transform* and the *HyperLCA Entropy Coder* that run in parallel following a producer-consumer approach. Consequently, the slowest block is the one determining the productivity of the proposed architecture. In this sense, the *HyperLCA Transform* block bears most of the complexity and computational burden of the compression process. For this reason, several optimizations have been applied during its design in order to achieve a high degree of parallelism and, thus, to reduce the latency. One of the most important improvements is the realization of the map-reduce programming model to enable an architecture with multiple PEs working concurrently on several bands. The experiments have been made using different configurations of the algorithm input parameters, that is, N_{bits} = (12, 8), BS = (1024) and CR = (12, 16, 20) and, for PEs = (1,2,4,8,10,16,20). The number of p pixels to be extracted from each image block, $\mathbf{M}_k$, to ensure the minimum CR desirable by the user is obtained at design time from the input parameters following Equation4.1 of Chapter 4.

To evaluate the HyperLCA HWacc, the proposed architecture has been implemented using the Vivado Design suite. This toolchain is provided by Xilinx and features a HLS tool (Vivado HLS) devoted to optimize the developing process of an IP component FPGA-based solutions for their own devices. The implemented prototype targets the XC7Z020-CLG484 version of the Xilinx Zynq-7000 SoC. This FPGA has been selected because of its low-cost, low-weight and high flexibility, features that make it an interesting device to be integrated in aerial platforms, such as drones. The aim is to evaluate the capability of

a mid-range reconfigurable FPGAs, such as the XC7Z020 chip, for a specific application such as the HyperLCA compression algorithm. Hence, and due to the amount of resources available on the target device, the maximum possible number of PEs for the proposed hardware prototype is 20.

Table 6.2 summarizes the average resource utilization required by the *HyperLCA Transform* block for the processing of the test-bench described in Section 6.2.1. In this sense, the demanded hardware resources only depend on the BS and the PEs parameters. Table 6.3 shows the post-synthesis results for the *HyperLCA Entropy Coder* block. Unlike Table 6.2, the resources demanded by the coder does not depend on the BS parameter or the number of PEs. It is important to mention that the majority of the employed BRAM, flipflops (FFs) and look-up-tables (LUTs) are assigned to the two AXI-Memory interfaces that the HLS tool generates for the *Pixel Reader* and the *Bitstream Packer* modules (see Figure 6.8).

BS	PEs	BRAM18K	DSP48E	FlipFlops	LUTs
1024	1	96 (68.57%)	9 (4.09%)	7121 (6.69%)	5434 (10.21%)
	2	94.5 (67.50%)	16 (7.2%)	6655 (6.25%)	6257 (11.76%)
	4	98 (70%)	30 (13.64%)	7725 (7.26%)	7416 (13.94%)
	8	96.5 (68.93%)	58 (26.36%)	9910(9.31%)	9553(17.96%)
	10	104.5 (74.64%)	72 (32.73%)	11415 (10.73%)	11630 (21.86%)
	16	95.5 (68.21%)	114 (51.82%)	14891 (14%)	14721 (27.67%)
	20	95.5 (68.21%)	142 (64.55%)	17916 (16.87%)	18851 (35.43%)

TABLE 6.2: Post-Synthesis results for the different versions of the *HyperLCA Transform* HWacc for a Xilinx Zynq-7020 programmable SoC and image block up to 160 bands.

BS	BRAM18K	DSP48E	FFs	LUTs
1024	7 (2.5%)	1 (0.45%)	3464 (3.25%)	4106 (7.71%)

TABLE 6.3: Post-Synthesis results for the *HyperLCA Entropy Coder* HWacc for a Xilinx Zynq-7020 programmable SoC and pixel size up to 160 bands.

Additionally, Table 6.4 shows the average number of hyperspectral blocks, $\mathbf{M}_k$, that the HWacc is able to compress per second (FPS) for a specific configuration of the HWacc using a clock frequency of 143 MHz. It is worth noting that these results include only the computation of the *HyperLCA Transform* block since the *Entropy Coding* stage is executed in parallel and takes roughly 50% less time on average. Since the relation between both hardware components is a dataflow architecture, the latency of the whole

process corresponds to the maximum one, that is, the delay of the *HyperLCA Transform* stage.

N_{bits}	BS	CR	Maximum frame rate (FPS)						
			PE=1	PE=2	PE=4	PE=8	PE=10	PE=16	PE=20
12	1024	12	67.91	140.30	275.81	503.45	603.09	857.47	997.80
		16	88.38	182.37	357.02	636.44	754.67	1045.57	1200.30
		20	110.61	202.63	444.14	772.54	906.49	1225.44	1387.95
8	1024	12	48.99	101.32	199.98	373.50	451.81	659.67	778.90
		16	67.91	140.29	275.82	503.52	603.13	857.44	997.75
		20	80.31	165.79	325.12	584.98	696.25	974.35	1123.68

TABLE 6.4: Maximum frame rates (FPS) obtained by the *HyperLCA Transform* HWacc on a Xilinx ZynQ-7020 programmable SoC at a clock frequency of 143 MHz for hyperspectral images with 160 bands. Evaluation is made according to different PE settings.

Several conclusions can be derived from the aforementioned results. One key factor is the minimum frame rate that must be supported for the targeted application. Ideally, such threshold would correspond to the maximum frame rate provided by the employed hyperspectral sensor, that is 330 FPS. However, the experimental validation of the camera set-up in the targeted application tells us that frame rates between 150 and 200 FPS are enough to obtain hyperspectral images with the desired quality, given the speed and altitude of the flights. Therefore, a threshold value of 200 FPS is established as the minimum throughput to validate the viability of the HyperLCA hardware core. As it can be seen from Table 6.4, settings of $PE \geq 4$ fulfil this constraint, even for the most demanding scenario ($CR = 12$ and $N_{bits} = 8$). Furthermore, $PE \geq 8$ settings also complies with the maximum frame rate established in 330 FPS. Indeed, the range of maximum compression frame rates obtained by $PE = 20$ are between 778.90 and 1387.95 FPS.

From the above analysis, we can also conclude the importance of the followed map-reduce programming model for the attainment of the requirements imposed by the targeted application. Nonetheless, the increase in the number of PEs implies the utilisation of more hardware resources. In this sense, it is important to also corroborate that the usage of the necessary logic falls into the available resources within the targeted device. As it can be seen from results shown in Table 6.2, the amount of digital signal processors (DSPs), FFs and LUTs increases with the number of PEs. Nonetheless, the percentages of these hardware resources are within the limits imposed by the resources available on

the targeted device for all considered PE settings. In fact, the employed DSPs ascend to the 64.55% of the total, the FFPs to 16.87% and the LUTs to the 35.43% for $PE = 20$. The limiting factor is indeed the BRAM, which are employed in roughly 67.50%-74.64% of the totality. Nonetheless, the overall consumption of hardware resources expended by the parallel execution of both the *HyperLCA Transform* and the *HyperLCA Entropy Coder* blocks in the same targeted device does not surpass the boundaries.

The PE=10 version needs a special remark. Such version represents an anomaly in the linear behaviour of the resource demand. The total capacity of the BRAM used to instantiate *SBuffer* is clearly oversized for that datawidth to assure that a hyperspectral block and its transformations could be stored in-circuit. Secondly, in addition to the resources needed by the HWacc of the *HyperLCA Transform*, it is necessary to take into account those corresponding to the other components in the system such as the *HyperLCA Entropy Coder* or the DMA (*Direct Memory Access*) used to move the hyperspectral data ($\mathbf{M}_k$) from/to DDR to/from the HWacc.

6.3.5.2 Evaluation of the HW-LbL-FAD Hardware Accelerator

In order to evaluate the HW-LbL-FAD HWacc, the proposed architecture explained in Section 6.3.3 has been also implemented using the Vivado Design suite from Xilinx, as well as targeting the XC7Z020-CLG484 version of the Xilinx Zynq-7000 SoC. The experiments addressed in this Section have been made using different configurations of the PE parameter within the map-reduce programming model. Unlike the preceding analysis made about the HyperLCA algorithm, N_{bits} and CR parameters have been not included in the assessment since they only take part in the compression process.

BS	*PEs*	BRAM18K	DSP48E	FFs	LUTs
1024	1	107 (76.43%)	14 (6.36%)	8073 (7.59%)	6744 (12.68%)
	2	91.5 (65.36%)	22 (10%)	8624 (8.11%)	7470 (14.08%)
	4	95.5 (68.21%)	38 (17.27%)	9981 (9.38%)	9115 (17.13%)
	8	98.5 (70.36%)	70 (31.82%)	12666(11.90%)	12411(23.33%)
	10	102.5 (73.21%)	86 (39.09%)	14787 (13.90%)	14856 (27.92%)
	16	96.5 (68.93%)	134 (60.91%)	18433 (17.32%)	18724 (35.20%)
	20	98.5 (70.36%)	166 (75.45%)	23071 (21.68%)	23493 (44.16%)

Table 6.5: Post-Synthesis results for the different versions of the *HW-LbL-FAD* HWacc for a Xilinx Zynq-7020 programmable SoC and image block up to 160 bands.

BS	Maximum frame rate (FPS)						
	PE=1	PE=2	PE=4	PE=8	PE=10	PE=16	PE=20
1024	130.11	259.96	518.87	900.22	1055.30	1423.01	1609.90

TABLE 6.6: Maximum frame rates (FPS) obtained by the HW-LbL-FAD HWacc on a Xilinx ZynQ-7020 programmable SoC at a clock frequency of 143 MHz for hyperspectral images with 160 bands. Evaluation is made according to different PE settings.

Table 6.5 summarizes the average resource utilization required by the HW-LbL-FAD HWacc displayed in Figure 6.9 for the processing of the test-bench described in Section 6.2.1. As it can be seen, all types of hardware resources show a slight increase due to the additional logic required by this HWacc compared with the *HyperLCA Transform* HWacc. In this regard, the *Brightness AD* and the *Stop_cond* HLS modules have been included, as well as the *xMatrix* FIFOs for the storage of the orthogonal $\mathbf{Q}$ and $\mathbf{U}$ vectors representative of the background distribution and intermediate multiplexors. Nonetheless, the total amount of hardware resources expended by the anomaly detection process performed by the HW-LbL-FAD HWacc does not surpass the limited established by the targeted platform. In fact, they ascend to the 70.36% of the BRAM, the 75.45% of the DSPs, the 21.68% of the FFs and the 44.16% of the LUTs for the most demanding scenario in terms of hardware resource, that is, $PE = 20$.

Additionally, another key factor of the targeted application is the minimum number of hyperspectral frames that should be processed in a second in order to ensure a real-time processing. In this sense, Table 6.6 shows the average number of hyperspectral blocks, $\mathbf{M}_k$, that the HWacc is able to compress per second (FPS) for a specific configuration of the HWacc using a clock frequency of 143 MHz. As it can be seen, it surpasses the limit of 330 FPS for configurations of $PE \geq 4$ and also, more than 200 FPS are obtained by $PE = 2$. Indeed, the reached frame rates are slightly higher for the HW-LbL-FAD HWacc than the HyperLCA HWacc. It has totally sensed since the *Brightness* HLS module is not repeated p times once the background distribution is estimated in *Stage 2* and the *Brightness AD* module is just launched once at the end of the process for the identification of the anomalous spectra.

6.3.5.3 Evaluation of the HADeLoC Hardware Accelerator

As it was concluded in preceding Chapter 5, the set of core operations proposed in this Thesis may be perfectly applicable for the simultaneous execution of both anomaly detection and lossy compression of HSIs. The HADeLoC emerged as a resource-optimized solution to those scenarios where the aforementioned hyperspectral analysis techniques have to be concurrently launched in the same piece of hardware. In this Section, we focus on the evaluation of this proposed methodology for the execution of multiple processes in FPGA-based systems through the assessment of the developed HADeLoC HWacc. The architecture of this HWacc is divided as well in two blocks, the block that executes the set of core operations as it was described in Section 6.3.4 and the *Entropy Coder* block that performs the CCSDS prediction error mapping and the Golomb–Rice entropy-coding algorithms on the different vectors that will constitute the compressed data. Although slight differences are introduced in the way as the bitstream is configured, the hardware-design followed for the definition of the *HyperLCA Entropy Coder* block in the HyperLCA HWacc may be also reused for the HADeLoC issue. Therefore, the conclusions drawn in Section 6.3.5.1 about this hardware module can be also extrapolated for the analysis of the HADeLoC approach.

The experiments addressed in this Section have been made using different configurations of the PE parameter within the map-reduce programming model. In this case, it has been discarded the effects of the N_{bits} and the CR since they do not affect either the hardware resource utilization or the average number of hyperspectral blocks, $\mathbf{M}_k$, that the HWacc is able to process per second. As a remainder, the number of p selected pixels from each image block, $\mathbf{M}_k$, for the background estimation is now a function of the spectral variability present in the HSIs and does not depend on the minimum desired CR and the N_{bits} parameter, unlike the *Initialization* stage of the HyperLCA compressor. Regarding N_{bits}, it only intervenes in the packing of the estimated $\mathbf{V}$ vectors to be compressed and transmitted. Therefore, it could slightly affect the number of cycles required for the execution of the *HyperLCA Entropy Coder* block. Nevertheless, the two modules that compose the HADeLoC HWacc work within a dataflow architecture and thus, the latency of the whole process corresponds to the maximum one, that is, the delay of the set of core operations computed by the former module.

Table 6.7 summarizes the average resource utilization required by the HADeLoC block displayed in Figure 6.10 for the processing of the test-bench described in Section 6.2.1. As it can be seen, only the FFs and the LUTs are increased in barely 0.2-0.3% compared

with the HW-LbL-FAD HWacc. On the contrary, the BRAM and DSP utilisation remains constant. It is due to some minor changes introduced in the FSM described in Figure 6.3.4 for connecting the *Brightness AD* module with the *SBuffer*. As it can seen, two hyperspectral analysis techniques, that is, the lossy compression of HSIs and the detection of anomalous spectra, can be simultaneously managed in the same piece of hardware with the developed HADeLoC HWacc using the 70.36% of the BRAM, the 75.45% of the DSPs, the 21.71% of the FFs and the 45.04% of the LUTs for the most demanding scenario in terms of hardware resource, that is, $PE = 20$. The limiting factor is indeed the BRAM, which are employed in roughly 68.21%-76.43% of the totality. Nonetheless, the total hardware resources expended by the parallel execution of both the HADeLoC block displayed in Figure 6.10 and the *HyperLCA Entropy Coder* block in the same targeted device does not surpass the boundaries. Additionally, the extension of the set of core operations for the execution of both processes only involves an increase of 2.15% of the BRAM, 10.90% of the DSPs, 4.84% of the FFs and 10.51% of the LUTs compared with the execution of only the lossy compression of the HSIs performed by the HyperLCA compressor.

BS	*PEs*	BRAM18K	DSP48E	FFs	LUTs
	1	107 (76.43%)	14 (6.36%)	8181 (7.69%)	6857 (12.89%)
	2	91.5 (65.36%)	22 (10%)	8731 (8.21%)	7624 (14.33%)
	4	95.5 (68.21%)	38 (17.27%)	100041 (9.44%)	9261 (17.41%)
1024	8	98.5 (70.36%)	70 (31.82%)	12728(11.96%)	12576(23.64%)
	10	102.5 (73.21%)	86 (39.09%)	14840 (13.95%)	14846 (27.91%)
	16	96.5 (68.93%)	134 (60.91%)	18491 (17.38%)	18970 (35.66%)
	20	98.5 (70.36%)	166 (75.45%)	23096 (21.71%)	23959 (45.04%)

TABLE 6.7: Post-Synthesis results for the different versions of the HADeLoC HWacc for a Xilinx Zynq-7020 programmable SoC and image block up to 160 bands.

Regarding the execution times, Table 6.8 shows the average number of hyperspectral blocks, $\mathbf{M}_k$, that the designed HWacc is able to process per second (FPS) for a specific configuration of the HWacc using a clock frequency of 150 MHz. Once again, it surpasses the limit of 330 FPS for configurations of $PE \geq 4$. In this case, it is not right a comparison with the HyperLCA HWacc since different CRs are reached by both proposals. Regarding the HW-LbL-FAD HWacc, the average decrease in the achieved frame rates ascends to roughly the 4%, which indeed matches with the proportion of anomalous frames since more $\mathbf{e}_n$ are extracted for these abnormal behaviours. Nonetheless, the maximum obtained frame rates substantially exceeds the minimum limit set in 330 FPS by the target

application when both image processing analysis are being simultaneously running in the same piece of hardware.

BS	Maximum frame rate (FPS)						
	PE=1	PE=2	PE=4	PE=8	PE=10	PE=16	PE=20
1024	122.63	245.05	489.26	855.45	1006.04	1366.99	1552.60

TABLE 6.8: Maximum frame rates (FPS) obtained by the HADeLoC HWacc on a Xilinx ZynQ-7020 programmable SoC at a clock frequency of 143 MHz for hyperspectral images with 160 bands. Evaluation is made according to different PE settings.

6.3.5.4 General discussions about the obtained results

As fully discussed in Sections 6.3.5.1-6.3.5.3, the developed HWaccs for the implementation of the HyperLCA, the HW-LbL-FAD and the HADeLoC algorithms on FPGA-based systems meet the requirements imposed by the targeted application in terms of the maximum frame rate defined by the data acquisition platform. In this section, we would like to also provide a comprehensive benchmarking studio among these proposed HWaccs in terms of the resource utilisation and the maximum reached date rates. To this end, we support on Figures 6.11-6.12.

Figure 6.11 graphically shows the percentage of the total number of hardware resources (BRAM, DSPs, FFs and LUTs) required by the developed HWaccs according to the different configurations of the PE parameter considered in the analysis made along this Section. Additionally, it is also displayed a hypothetical situation in which the HyperLCA and the HW-LbL-FAD algorithms are independently implemented in the same piece of device for a parallel and simultaneous execution of both processes. With it, we want to represent those situations in which different mathematical methods are selected from the wide assortment of proposals encountered in the literature for each hyperspectral data analysis technique to be addressed and accelerated using parallel computing devices. As it can be seen, it is clear that the implementation of both the HW-LbL-FAD detector and the HyperLCA compressor could not be addressed as independent entities in the selected XC7Z020-CLG484 FPGA due mainly to the total amount of available BRAM resources for any PE configuration. This situation is also repeated for the DSPs with $PE = [16, 20]$, although it is not a critical situation since the minimum target frame rate can be also fulfilled with $PE = 8$. Nonetheless, this scenario is solved by the HADeLoC proposal since it carries out both the lossy compression of HSIs and the detection of anomalous spectra

employing almost the same resources as only one process, specially, the HW-LbL-FAD method. Therefore, it is deduced the importance of the design of algorithmic solutions based on the same mathematical method in order to allow sharing blocks of operations among them in the pursuit of reducing the resource utilization.

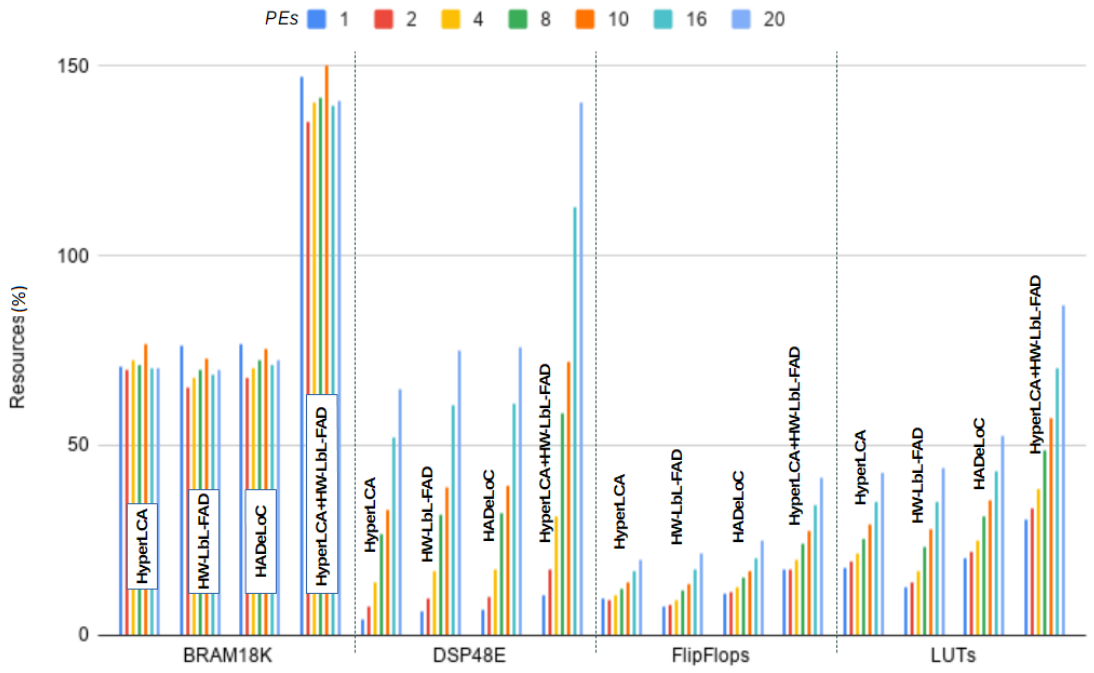


FIGURE 6.11: Comparison in terms of the resource utilization made by the HWaccs developed for the FPGA implementation of the HyperLCA, the HW-LbL-FAD and the HADeLoC methods. Additionally, they have been also compared with a parallel and independent implementation of the HyperLCA and the HW-LbL-FAD methods on the same hardware device.

On this basis, the HyperLCA and the HW-LbL-FAD methods are based on the same mathematical method and, more specifically, in the Gram-Schmidt method performed by the set of core operations proposed in this Thesis. For this reason, it could be inferred that they could be sequentially executed in the same piece of hardware without an excessive increment of the hardware resources and above all, within the limits imposed by the target computing device in terms of the available BRAM resources. Figure 6.12 shows a comparison in terms of the maximum processing data rates reached by each of the developed HWaccs, as well as, by the above mentioned scenario in which the HyperLCA and the HW-LbL-FAD methods are serially launched. From it, we can conclude that,

although configurations of $PE \geq 8$ are able to fulfil the requirements imposed by the targeted application according to the number of hyperspectral frames to be processed in a second, the execution times get affected in such a way that they are increased in almost the 50% compared with those required by HADeLoC approach, which actually addresses both processes as well, and by the single execution of each the HyperLCA and the HW-LbL-FAD methods. This fact can be deduced since the columns of Figure 6.12 corresponding with the HyperLCA + HW-LbL-FAD are half as high as those related with the HADeLoC, the HyperLCA and the HW-LbL-FAD methods. Results also show that the execution times required for the implementation of both the lossy compression of HSIs and the detection of anomalous spectra handled by the HADeLoC approach is almost the same as just the performance of only one process, specially, the HW-LbL-FAD algorithm. Therefore, these discussions are certainly in the line of the research goals to be accomplished in this Thesis. Additionally, the conclusions drawn in above lines could be also extrapolated to those scenarios in which different mathematical methods have to be sequentially addressed in the same computing device, besides, it may not have sufficient available resources for that purpose.

Finally, although the implementation of the ADeLoC solution has not been addressed in this Thesis, it can be inferred that its implementation may be also efficiently carried out in the the selected XC7Z020-CLG484 FPGA since there is enough additional DSPs, FFs and LUTs for such purpose and, the employed resources in terms of BRAM should be almost the same as the HADeLoC solution. In this sense, the execution times would not be so affected as the serial execution of the HyperLCA + HW-LbL-FAD methods but the same results are ensured, unlike with the HADeLoC solution as it was further discussed in Chapter 5.

6.4 Real-time implementation of the HyperLCA algorithm on embedded GPUs

This section focuses on the parallel implementation of the HyperLCA algorithm on NVIDIA GPUs through the NVIDIA Computer Unified Device Architecture (CUDA) programming model. In this Section, we evaluate the suitability of the HyperLCA algorithm for the real-time lossy compression of hyperspectral data for applications characterized by high data-rates. Likewise, it is focused on remote sensing applications where the

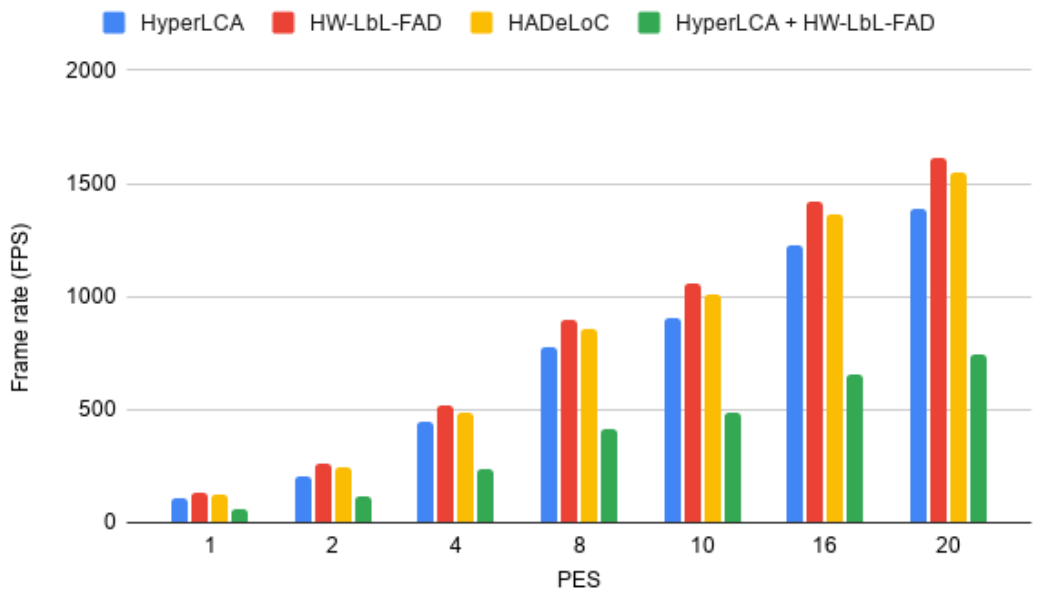


FIGURE 6.12: Comparison in terms of the maximum processing data rates achieved by the HWaccs developed for the FPGA implementation of the HyperLCA, the HW-LbL-FAD and the HADeLoC methods. Additionally, they have been also compared with a sequential execution of the HyperLCA and the HW-LbL-FAD algorithms on the same hardware device.

available computational resources are limited, due to power, weight or space limitations. Concretely, we focus on the smart farming application described in preceding Section 6.1. In this context, a hyperspectral pushbroom scanner is mounted onto a UAV for collecting periodical information of the crops, which results in a huge amount of data that needs to be managed, processed and analysed. The hyperspectral camera provides a maximum frame rate of 330 FPS when collecting 224 spectral bands, or even more for less number of them. Therefore, this maximum frame rate sets the minimum target compression frame rate to be reached by the conduced hardware implementations. By achieving this frame rate, it can be guaranteed that the collected hyperspectral data can be always compressed in real-time and, the capturing frame rate can be specified according to the characteristics of the application (such as flight height, intensity of the light, area to be covered, etc.) without being limited by the data compression process. It must be mentioned that this scenario represents the most extreme case since capturing frame rate settings are normally lower, roughly 150-200 FPS.

Additionally, the data acquisition platform needs a processing board for the management of many tasks at the same time, such as, the data acquisition, the data calibration, the

data storing and/or their transfers, the camera controlling and also the drone flight control. Therefore, some of the characteristics to be met by the selected processing board in order to be useful for the target application can be determined by the necessities and limitations imposed by the drone and the hyperspectral imaging acquisition system. In particular, it has to be relatively small and light and it has also to demand a relatively low power consumption. Additionally, this processing device has to be able to manage and process the high amount of data provided by the hyperspectral sensor in real-time. Due to these reasons, we have put our attention in the NVIDIA Jetson developer boards, and more specifically in the Jetson TK1, the Jetson TX2, Jetson Nano and the most recent supercomputer Jetson Xavier-NX developer kits. The Jetson TK1 module integrates the less advanced, oldest generation of the four GPU architectures, while the Jetson Nano instantiates the fewer execution units or CUDA cores. On the contrary, the Jetson Xavier-NX represents one of the latest NVIDIA power-efficient products, which offers more than 10x the performance of its widely adopted predecessor, Jetson TX2. Moreover, these boards embed a LPGPU that allows parallel programming for speeding up the executed processes. This is specially useful for accelerating the compression of the collected hyperspectral data. Although, the experiments carried out are oriented to the necessities imposed by a specific smart farming application, all drawn conclusions are extrapolated to other fields in which remotely sensed HSIs have to be compressed in real-time.

In this sense, the *HyperLCA Transform* is the most computational demanding processing stage of the HyperLCA compressor in terms of the number of operations to be performed and the execution times. For this reason, it has been implemented on the LPGPUs available in the Jetson boards through the use of NVIDIA CUDA programming language. Furthermore, three different implementation models of the whole compression model have been studied, seeing them as an evolution towards an optimal configuration that fulfils the constraints imposed by the targeted application. These strategies are focused on exploiting the parallelism of the HyperLCA compressor beyond the thread level parallelism inherent to the GPU programming model, more concretely, pipelining the execution of the compression stages of the HyperLCA algorithm for each independently processed image block, as well as the communications and memory transfers.

6.4.1 Graphics Processing Units

A GPU can be understood as an array of many independent processors that correspond with an independent execution thread. Each of these execution threads are able to only

execute one operation per cycle. However, the high parallelism inherent to the GPUs arises from their capability of executing the same operation in many different threads at the same time using different data. Figure 6.13 shows a graphic example of the thread-level parallelism achieved in a GPU in which three groups of independent operations; GA, GB and GC; must be performed on each image pixel. As it can be seen, each $i \in [1, n]$ thread processes an image pixel independently and each operation is serially executed, one after the other, in different clock cycles for all employed GPU threads. In order to better understand the GPU architectures and their parallelism, the basic components of any NVIDIA GPU architecture will be briefly described in Sections 6.4.1.1 and 6.4.1.2.

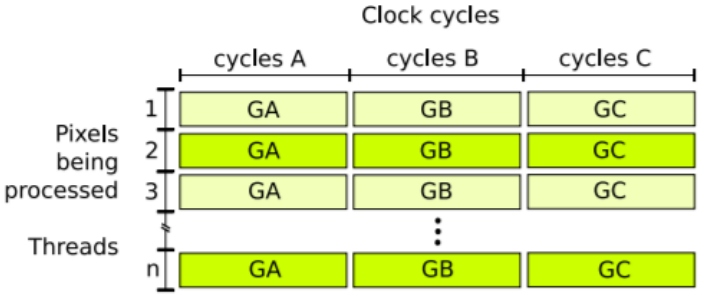


FIGURE 6.13: GPU thread-level parallelism (image extracted from [4]).

6.4.1.1 GPU hardware platforms

A GPU is built around a scalable array of *Streaming Multiprocessors* (SMs) that support the concurrent execution of multiple threads. The replication of this architectural block leads to the high hardware parallelism of the GPUs [231]. The basic components that compose a SM are: execution units or CUDA cores, shared memory/L1 cache, register file, load/store units, special function units, instruction dispatch units and warp schedulers.

From a software point of view, the GPU threads are organized into thread blocks when a kernel grid is launched in the CUDA programming model. These thread blocks are a pure software concept that actually do not exit in the hardware design. From a hardware point of view, these thread blocks are scheduled on only one SM but, a SM can handle more than one thread block at the same time. Once that a thread block is assigned to a SM, CUDA manages these threads in groups of 32, called warps. All threads in a warp execute the same instructions at the same time. The warp schedulers select active warps on each clock cycle and dispatch them to the execution units. After several clock cycles, a pipeline of instructions is scheduled within the SM, as it is shown in Figure 6.14. The objective of this pipeline is to hide the instruction latencies. Hence, it is mandatory that

some warps are available in every clock-cycle to launch instructions in them while other warps are busy running previous instructions. In Figure 6.14, it can be seen how warp 0 is not selected again at least until it finishes executing the received instruction.

However, this ideal situation may be limited by the different GPU resources, causing gaps of time where no eligible warps are available in a clock cycle and consequently, new instructions cannot be launched. Figure 6.15 shows an example of this situation where, in a particular clock cycle, there are no warps that can be eligible by the warp scheduler 2. Some factors that negatively affect the instructions pipeline are:

1. The number of resident warps per SM, as well as, the number of resident thread blocks and threads. Generally, it is desired to achieve a high number of active warps per SM in such a way that it is more likely to find eligible warps in every clock cycle.

2. Shared memory and register files. The amount of shared memory is fixed and partitioned among the thread blocks scheduled on a SM while register file is partitioned among the threads. Hence, the higher the consumption of registers and shared memory by threads and blocks, the fewer warps that can be simultaneously scheduled on a SM.

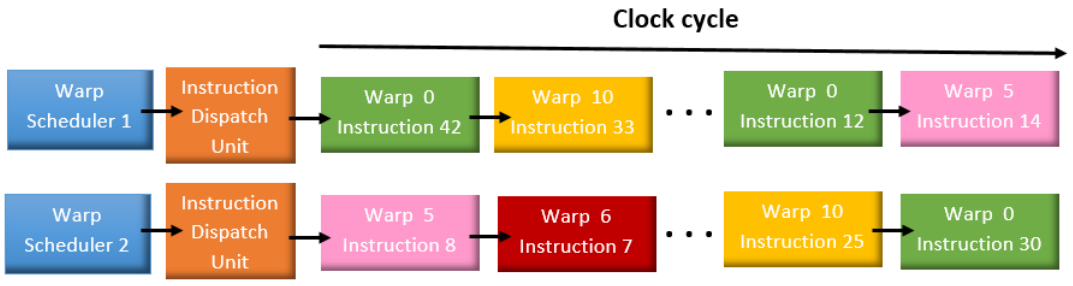


FIGURE 6.14: Pipeline of warps in a SM with two warp schedulers.

6.4.1.2 Streams and Concurrency

CUDA programming model permits the concurrent execution of kernels and memory transfers through a mechanism named *CUDA Streams*. A CUDA stream refers to a queue of operations; such as kernel launches, host-device data transfers and other commands; that are executed in the device in a strict ordering managed by the host code. However, the execution of operations in a stream are totally asynchronous with respect to the host

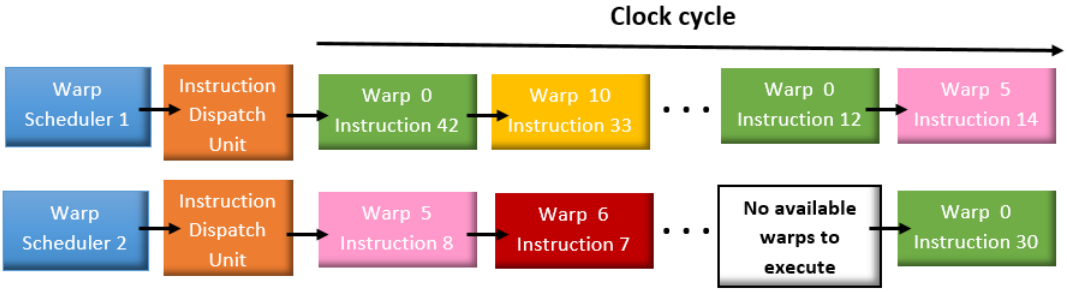


FIGURE 6.15: Pipeline with a gap of time where no warps are eligible.

and the operations running in other streams. It means that each stream may execute its operations out of order with respect to other streams. As a consequence, it permits overlapping multiple kernel launches and data transfers, which are being executed in different streams, if there are enough available GPU resources.

Nonetheless, if no stream is specified, all kernel launches and data transfers are implicitly queued in a default stream named *NULL stream*. The use of this default stream implies that all kernel launches and memory transfers are blocking calls and therefore, serially executed in the Host-Device model. It means that the aforementioned concurrency of multiple GPU operations is lost. As a consequence, those operations managed by non-default streams are blocked until the default stream is idle. Therefore, if the pipeline of multiple GPU operations is desired, non-default streams must be explicitly created and the GPU operations to be performed must be assigned to them.

The use of non-default streams is a powerful tool of the CUDA programming model that allows to deepen in the parallelism offered by the GPUs. Hence, they have been exploited in the hardware implementations conducted in this Section.

6.4.2 CUDA implementation of the HyperLCA algorithm

In this Section, the various methodologies proposed for the implementation of the HyperLCA lossy compressor in embedded LPGPUs are extensively described. Firstly, a GPU implementation of the *HyperLCA Transform* stage has been adequately handled in Section 6.4.2.1 following the traditional Host-Device CUDA programming model. Secondly, three different implementation models of the whole HyperLCA compressor have been addressed

in Section 6.4.2.2, using additional parallelization strategies beyond the thread level parallelism inherent to the GPU programming model. These parallelization strategies aim to make a more efficient use of the resources available in the targeted Jetson boards.

It is also important to mention that the four original computing stages of the HyperLCA compressor, that is, *Initialization, HyperLCA Transform, Preprocessing* and *Entropy Coding*, were defined from an algorithmic point of view. Nonetheless, they have been redefined for being implemented using a Host-Device model depending of whether they are potential candidates for being executed in the host domain, that is the central processing unit (CPU), or they are more prone to be executed in the device, that is the LPGPU. Hence, from an implementation point of view, the HyperLCA algorithm has been structured in three main stages:

1. *Initialization.* The number of p pixels to be selected from each image block, $\mathbf{M}_k$, is estimated according to the desired minimum compression ratio (CR) to be reached. This operation simply consists of a division where a configuration setting is defined. Moreover, it is computed once at the beginning of the compression process with independence of the number of image blocks to be compressed.

2. *HyperLCA Transform.* This current stage involves not only the operations of the *HyperLCA Transform*, but also the *Scaling of V vectors*. This stage comprises the greatest number of operations and consequently, it is the most computationally intensive part. Taking advantage of the high level of parallelism of the involved operations, they can be easily implemented in the LPGPUs embedded in the Jetson boards in order to decrease the overall time required by the execution of the compression process and ensure the high frame rates imposed by the targeted application.

3. *Entropy Coding.* This stage involves now the *HyperLCA Entropy Coding* stage and the *Error Mapping* step. The involved operations are much less computational demanding than those inherent to the aforementioned *HyperLCA Transform* and they have also a more sequential nature. For this reason, they are potential candidates for being executed in the host domain, that is, the CPU.

6.4.2.1 GPU implementation of the HyperLCA Transform

As it was further explained in Section 4.4.6 of Chapter 4, the *HyperLCA Transform* is the most computationally intensive part of the HyperLCA compressor, executing up to three orders of magnitude more number of operations than the *Entropy Coding* stage. For this reason, it has been implemented on the LPGPUs embedded in the targeted Jetson boards using CUDA as design language and following the traditional Host-Device CUDA model shown in Figure 6.16.

The CUDA programming model has been commonly used for developing applications for PCIe (Peripheral Component Interconnect Express) GPUs. In these systems, the GPU memory is separated from the Host memory. Accordingly, the traditional Host-Device model implies the data transfer from the host to the device. In the more recent embedded computing devices, such as the Jetson boards, the physical memory is shared between both the GPU and the CPU. In this situation, it is not necessary to copy the data from the host memory to the device memory. Additionally, the latest versions of the NVIDIA CUDA offer new mechanisms that allow taking advantage of these situations, such as the use of the unified memory concept. However, in the application targeted in this research work, there are many processes running in the Jetson boards that need to access to the same memory positions. For instance, the processes involved in the synchronization and data acquisition have to be able to store the data in memory places that are later read by the processes involved in the processing of them, such as the calibration or the compression processes. For doing so the benefits of the Linux shared memory concept has been exploited.

For being able to integrate the compression process with the rest of processes of the system, the captured frames are copied from the Linux shared memory space to other RAM memory spaces initialized in the style and manner proper to the CUDA programming language, which makes the data understandable and accessible by the processes running in the LPGPUs. Despite the evident disadvantage of introducing one extra copy for each captured frame, this also provides the advantage of following the traditional Host-Device model, copying the data from the Host memory to the Device memory, which is just another location on the same physical memory in this situation. Accordingly, the implementation of the HyperLCA compressor described below could be straightforward compiled for any system that uses PCIe GPUs with its own separated memory.

In general terms, the implementation of the HyperLCA Transform using the Host-Device CUDA model comprises 3 main stages:

1. *Write* $H \rightarrow D$: a preliminary transfer of BS hyperspectral pixels from the host ($\mathbf{M}_{\mathbf{host}_k}$) to the device ($\mathbf{M}_k$).
2. *Kernels*: seven different operation blocks or kernels are launched in the GPU to perform the operations involved in the *HyperLCA Transform*.
3. *Write* $D \rightarrow H$: transfer of the *HyperLCA Transform* outputs, that is $\hat{\boldsymbol{\mu}}$, $\mathbf{E}$ and $\mathbf{V}$, from the device to the host.

The operation blocks or kernels that encompass the second stage are briefly outlined below. In addition, the kernel configurations with respect to the set CUDA thread blocks and CUDA threads are also shown in Table 6.9, where $\lceil - \rceil$ means rounding-up to the nearest whole number. Since BS directly affects the fixed number of threads per CUDA block, this parameter has been defined in terms of this HyperLCA input parameter. It is also noted that the employed kernel configurations have been selected taking into account that the maximum number of threads per CUDA block for NVIDIA GPUs with compute capabilities higher than 2.0 is 1024 [232]. In addition, kernels have been defined for HSIs up to 256 spectral bands and for a maximum BS of 1024 pixels.

1. Kernel 1: Cast to float.
   The *HyperLCA Transform* was originally designed to use floating-point arithmetic. In addition, GPUs are generally optimized to execute single precision floating-point operations. Accordingly, the data captured by the hyperspectral sensor, which are stored as unsigned integer values, are firstly converted to single precision floating point representation. This process is fully parallelizable since each GPU thread independently processes one element within the image block, $\mathbf{M}_k$.
2. Kernel 2: Centroid Calculation.
   This kernel computes the average pixel, $\hat{\boldsymbol{\mu}}$, of the image block, $\mathbf{M}_k$. To issue this calculation, nb CUDA thread blocks are defined. Threads of a common block independently work with the pixel elements within the same spectral band. To compute the mean value per band, a reduction strategy is employed, making use of the GPU shared memory. For this reason, the number of CUDA threads has been defined as the power of two whole number closer to the half of BS. In addition, each element

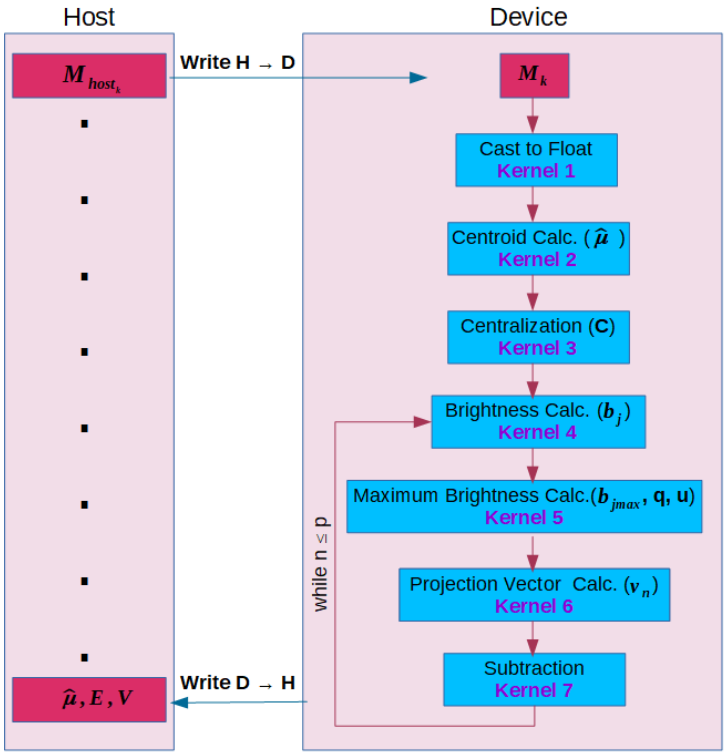


FIGURE 6.16: Flow-chart of the *HyperLCA Transform* in the Host-Device model

of the average pixel is rounded to the closest integer value before starting the *HyperLCA transform* stage. This not only eases the posterior *Preprocessing* and *Entropy coding* stages, since these two stages work with integer values, but also guarantees to use the exact same vector when the inverse process is performed to decompress the image, which increases the overall accuracy of the compression-decompression process. Nonetheless, since the *HyperLCA Transform* uses floating point arithmetic, the resulting centroid pixel after the rounding is again casted to float.

3. Kernel 3: Centralization.
   To centralize $\mathbf{M}_k$, the average pixel, $\hat{\boldsymbol{\mu}}$, is subtracted from each image pixel, getting the centralized version of the image, $\mathbf{C}$. To issue this operation, BS CUDA thread blocks of nb threads are defined where each block independently processes each pixel. In fact, each thread within the same block works with a pixel band, subtracting the information of the corresponding band in vector $\hat{\boldsymbol{\mu}}$.

4. Kernel 4: Brightness calculation.
   The brightness calculation of a pixel involves the addition of the squared values of each spectral bands. To issue this set of operations in just one kernel, 2D CUDA thread blocks are defined where x dimension matches up with each image pixel and

y dimension with the spectral bands. Once each thread performs the square of each image element, a reduction strategy along y dimension is followed to perform the addition of the squares for each image pixel. To do this, the number of threads along y dimension has been defined as the half of the maximum number of spectral bands, that is, 256/2. A portion of the GPU shared memory is also employed.

5. Kernel 5: Maximum Brightness Calculation.
   This kernel calculates the brightest pixel index and its value, as well as the orthogonal $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors. To address all these issues, just one CUDA thread block is required. In order to find the brightest pixel index and its value, a reduction strategy is followed making use of the GPU shared memory. Unlike kernel 2, the number of threads must be defined as the power of two whole number closer to BS in this case, since we need to work with thread indexes. The $\mathbf{q}_n$ vector corresponds to the pixel in $\mathbf{C}$ with the highest brightness. The $\mathbf{u}_n$ vector is equal to the $\mathbf{q}_n$ vector but whose elements have been divided by the maximum brightness. Since kernels have been defined for a maximum of 256 spectral bands and hence, $\mathbf{u}_n$ and $\mathbf{q}_n$ vectors will have 256 elements as maximum, a minimum of 256 CUDA threads is required in this case.

6. Kernel 6: Projection Vector Calculation.
   This kernel computes the projection vectors, $\mathbf{v}_n$. This calculation may be tackled in the same way as Kernel 4, where the brightness vector calculation is performed. In this case, instead of computing the addition of the squares of each pixel element, each pixel band is multiplied by its homologous in vector $\mathbf{u}_n$.

7. Kernel 7: Subtraction.
   This kernel addresses the subtraction of the spectral information spanned by the selected $\mathbf{q}_n$ vector from $\mathbf{C}$. Due to this reason, $\mathbf{C}$ contains the remaining spectral information which cannot be represented by the previous extracted pixels. To address these operations, nb CUDA blocks of BS threads are defined. Each block independently processes each spectral band and each thread within the same block works with a pixel band. Taking advantage of this thread distribution, the *Scaling of V vectors* is also performed.

Since p pixels must be extracted, kernels 4 to 7 are launched p times for each image block, $\mathbf{M}_k$.

Kernel	CUDA thread blocks	CUDA threads
Cast to Float	$\lceil \frac{BS \cdot nb}{1024} \rceil$	1024
Centroid Calculation	nb	$2^{(\lceil log_2(BS/2) \rceil)}$
Centralization	BS	nb
Brightness Vector Calc.	$\lceil \frac{BS}{\frac{1024}{256/2}} \rceil$	$(\frac{1024}{256/2}, \frac{256}{2})$
Maximum Brightness Calc.	1	$2^{(\lceil log_2(BS) \rceil)} \geq 256$
Projection Vector Calc.	$\lceil \frac{BS}{\frac{1024}{256/2}} \rceil$	$(\frac{1024}{256/2}, \frac{256}{2})$
Subtraction	nb	BS

TABLE 6.9: Kernel configurations in terms of CUDA threads and thread blocks.

6.4.2.2 Host-Device Model of the HyperLCA lossy compressor

In general terms, the ultimate goal of the developed GPU-based implementations of the HyperLCA algorithm is to design a model that permits to compress the hyperspectral frames in real-time for applications with limitations on computational resources. It means being able to compress each hyperspectral frame in less time than the required for capturing it. Accordingly, it is avoided the accumulation of high data volumes of hyperspectral pixels until being able to process them. Moreover, the targeted Jetson boards share the same physical RAM memory between the CPU and the GPU. Consequently, the maximum amount of data that can be hold in RAM memory for the compression process is limited, specially considering that, at the same time, other processes are being handled by the computing device.

In particular, we present an application where a hyperspectral sensor is mounted on a UAV that captures frames with a very high frame rate, using a computer platform with an embedded LPGPU. Many processes are running at the same time in the computer platform apart from the compression process. On this basis, the main objective of the addressed implementations lies in the acceleration of the most demanding parts of the HyperLCA compressor in order to guarantee the high frame rates imposed by the targeted application. At the same time, it is also necessary to lessen the computational burden of the CPU, which is executing many other processes apart from the compression one. As a consequence, different configurations of the Host-Device model have been studied, seeing them as an evolution towards an optimal configuration that fulfils the constraints of the particular application.

The first and simplest approach is to accelerate the second stage of the HyperLCA compressor, corresponding to the *HyperLCA Transform*, executing it in the LPGPU. It gives rise to the Host-Device model already described in Section 6.4.2.1 and shown in Figure 6.16. In this case, the three different HyperLCA compressor stages are managed sequentially by a single CPU process but, operations involved by the *HyperLCA Transform* stage are executed in the LPGPU. Here, each block of *BS* pixels is also serially compressed, one after the other, as Figure 6.17 shows. Throughout this Section, this configuration is referred as Parallel Model 1.

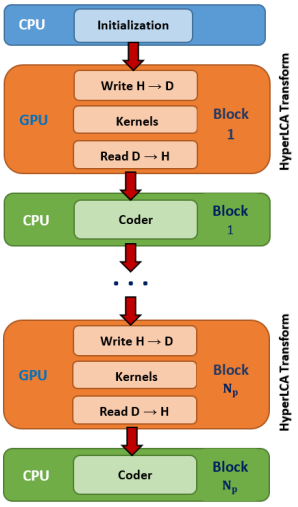


FIGURE 6.17: Parallel Model 1: HyperLCA compressor stages are sequentially performed in a single CPU process but the *HyperLCA Transform* stage is accelerated in the GPU [5].

A more in-depth analysis of the behavior of the different HyperLCA stages reveals that the execution of the *Entropy Coding* stage can be performed with total independence of the *HyperLCA Transform* stage. Once the coder receives the *HyperLCA Transform* outputs of a pixel block, that is $\hat{\boldsymbol{\mu}}$, $\mathbf{E}$ and $\mathbf{V}$, it can work with them in the background while the *HyperLCA Transform* stage processes a new block of pixels. In this context, two independent CPU processes have been used to implement this solution. One of them is in charge of the execution of the *HyperLCA Transform*, which involves transferring the data from the host to the device, launching the kernels that are executed in the LPGPU and bringing back the outputs of the *HyperLCA Transform* to the Host memory. The other CPU process is constantly waiting for new available *HyperLCA Transform* outputs for codifying them. As it was explained in more detail in Section 4.4.6 of Chapter 4, the codification is much less computational demanding than the *HyperLCA Transform*. Therefore, it is efficiently executed in the CPU. Throughout this Section, this configuration is referred as Parallel Model 2 and its flow-chart is shown in Figure 6.18.

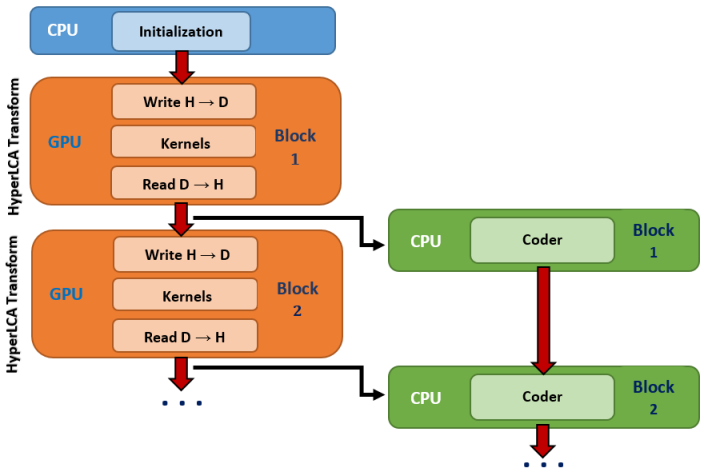


FIGURE 6.18: Parallel Model 2: the *HyperLCA Transform* and the *Entropy Coding* stage are managed by two independent CPU processes. The *HyperLCA Transform* is also executed by the GPU [5].

Finally, the task parallelism introduced by the CUDA streams has been also considered giving rise to the third and the last tested configuration named Parallel Model 3. As it was already introduced in Section 6.4.1.2, a CUDA stream represents a queue of operations executed in the GPU in a strict and specific order. However, different non-default streams run asynchronously and concurrently if there are enough available GPU resources. This has been used for pipelining the data transfers between the host and the device and the kernel executions for different block of pixels as shown in Figure 6.19. Specifically, one stream, named *GPU Stream Write*, is used for copying the pixel blocks to be compressed, one by one, from the host to the device. Another stream, named *GPU Stream Kernels*, manages the execution of the different kernels that perform the *HyperLCA Transform* operations for each pixel block, $\mathbf{M}_k$. Finally, a third stream, named *GPU Stream Read*, is used for copying the *HyperLCA Transform* outputs for each pixel block from the device to the host. For instance, the copy of the image block number 3, $\mathbf{M}_3$, from the host to the device, the execution of the *HyperLCA Transform* kernels for the image block number 2, $\mathbf{M}_2$, and the copy from the device to the host of the *HyperLCA Transform* outputs estimated from the image block number 1, $\mathbf{M}_1$, are concurrently launched. As in Parallel Model 2, the execution of the *Entropy Coding* stage is also pipelined with the *HyperLCA Transform* stage using another CPU process.

It is also important to highlight how the data synchronization among processes has been faced. As it was already mentioned, Jetson boards share the same physical RAM memory between the CPU and the GPU. Accordingly, the maximum amount of data that can be

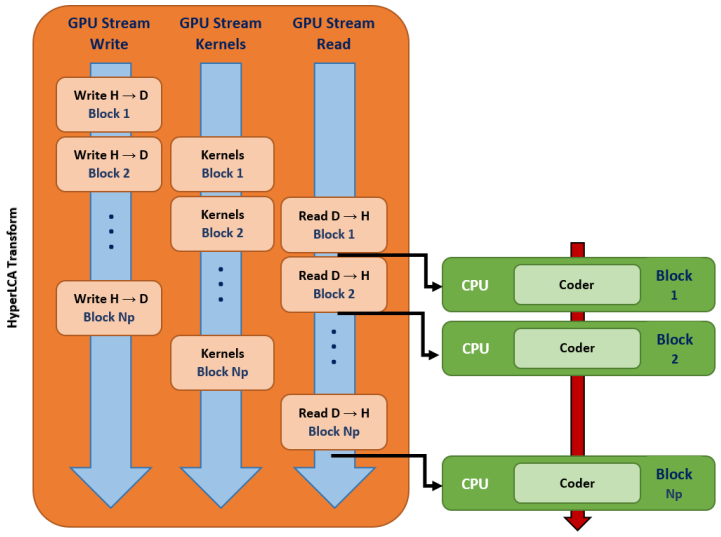


FIGURE 6.19: Parallel Model 3: the *HyperLCA Transform* has been implemented on the GPU using three non-default streams while the *Entropy Coding* stage is running in another CPU process [5].

hold in RAM memory for the compression process is limited, specially considering that at the same time, the boards are managing other processes such as the HSI capturing and the storage or transfers of the compressed images. Against this backdrop, we have used ring buffers in an asynchronous manner to stream the data. For instance, when the capturing process finishes sensing the first frame, this frame is stored in the first position of the capturing ring buffer and, starts capturing and storing the second frame in the second position of this buffer. At this time, the compression process starts processing the data that were stored in the first position of this buffer. Similarly, the results provided by the *HyperLCA Transform* are stored in another ring buffer from which the codification process reads the data to be coded. Finally, the coded data are stored in another ring buffer from which the storing and/or transferring processes read the information to be stored and/or transmitted. In order to avoid losing portions of data if any of the aforementioned processes momentarily stalls, these buffers need to be able to hold a relatively high amount of data in relation to the amount of data that is processed in one iteration. Hence, an image block, $\mathbf{M}_k$, composed of $BS = 1024$ pixels with 256 spectral bands each implies the definition of ring buffers intended for the capturing process of X times that size, where X is the number of frames that this buffer can hold. This is also applied to the other ring buffers involved in the whole process.

Due to all these reasons, BS has been kept as small as possible without compromising the quality of the compression results, as well as the maximum number of spectral bands

to be processed. Concretely, we have limited the BS to 1024 pixels and the number of spectral bands to 256, as maximum. However, since BS is a parameter inherent to the HyperLCA compressor and does not have to match the pixel-wide swath of the sensor, it is applicable to any pushbroom/whiskbroom hyperspectral scanner that provides images with less than 256 hyperspectral bands. For instance, this solution could perfectly work with well-known sensors used in remote sensing, such as AVIRIS (whiskbroom scanner, 677 pixel-wide swath, 224 bands), HYDICE (pushbroom scanner, 320 pixel-wide swath, 210 bands), CASI (pushbroom scanner, 512 pixel-wide swath, 288 bands) and HYPERION (pushbrrom scanner, 256 pixel-wide swath, 242 bands). In addition, it could also sit with most of the hyperspectral sensors collected by [31] in its Table 2 for being coupled with UAVs and certainly, with the application targeted in this Section. It is also worth to mention that kernels from 1 to 6, except for kernel 5, described in Section 6.4.2.1 can be actually launched for a BS up to 2048 pixels. For $BS \geq 1024$, kernels 5 and 7 must be redefined since the maximum number of CUDA threads per block that the CUDA compiler can manage is 1024 threads.

6.4.3 Experimental results

Several experiments have been carried out in order to evaluate the performance of the various implementation models developed for the execution of the HyperLCA compressor in some Jetson boards. In this regard, two assessment metrics have been used, that is, the speed-up reached by the parallel HyperLCA models compared with a serial reference model and, the obtained compression frame rate, which is defined as the number of image blocks composed of BS hyperspectral pixels that are compressed using the HyperLCA algorithm in one second. In the aforementioned serial model, the three stages of the HyperLCA compressor are managed sequentially, one after the other, giving rise to a Host model where the performance of the device is discarded. It means that once the compression of a pixel block has been completed, the next image block is processed. Throughout this Section, this model is referred as Reference Model.

6.4.3.1 Performance of the parallel implementations of the HyperLCA compressor in embedded LPGPUs in terms of speed-up

The experiments covered in this Section evaluate the speed-up achieved by the proposed parallel implementation models of the HyperLCA compressor compared with the Reference Model. These implementations have been described in Section 6.4.2.2 and they are referred to as *Parallel Model 1*, *Parallel Model 2* and *Parallel Model 3*. Additionally, the effects of different configurations of the HyperLCA input parameters have been also included in the analysis made. In this sense, the minimum desired *CR* parameter has been set to 12, 16 and 20, the *BS* to 1024, 512 and 256 and the N_{bits} parameter to 12 and 8. Regarding the *BS*, we have selected the above mentioned configurations since they are multiple of 32, that is, a warp size. As it has been explained in Section 6.4.1, the CUDA thread blocks are scheduled on only one SM of the GPU and CUDA manages these threads in groups of 32, called warps, where all threads in the same warp execute the same instructions at the same time. In order to define efficient kernels and making the best use of the GPU resources, we have defined CUDA thread blocks whose number of threads are multiple of 32. Hence, it must be also the case for *BS* since it directly affects them, as it can be inferred from Table 6.9.

Figures 6.20 and 6.21 graphically show the average speed-up obtained by the parallel implementations of the HyperLCA compressor in relation to its reference version for different configurations of the HyperLCA input parameters. For simplicity, we have only included the results for the Jetson TK1 and the Jetson TX2 boards. Specifically, Figure 6.20 shows the results obtained in the Jetson TK1 developer kit while Figure 6.21 shows the results obtained in the Jetson TX2. The speed-up is calculated as the ratio between the average time spent by each parallel model and the reference model to complete the entire compression process under the same settings of the HyperLCA input parameters. For doing this, the test bench described in Section 6.2.1 was used.

Several conclusions can be drawn from Figures 6.20 and 6.21. First of all, it can be observed that the *Parallel Model 1*, labeled as *Model 1*, is already 3 to 5 times faster than the reference version for both targeted boards. This speed-up is obtained just by executing the *HyperLCA Transform* in the corresponding LPGPU, using the developed kernels, as described in Figure 6.17. Secondly, it is also appreciable that this speed-up considerably increases for the *Parallel model 2* and *Parallel model 3*, labeled as *Model 2* and *Model 3*, respectively, being *Model 3* generally faster than *Model 2* in both targeted boards.

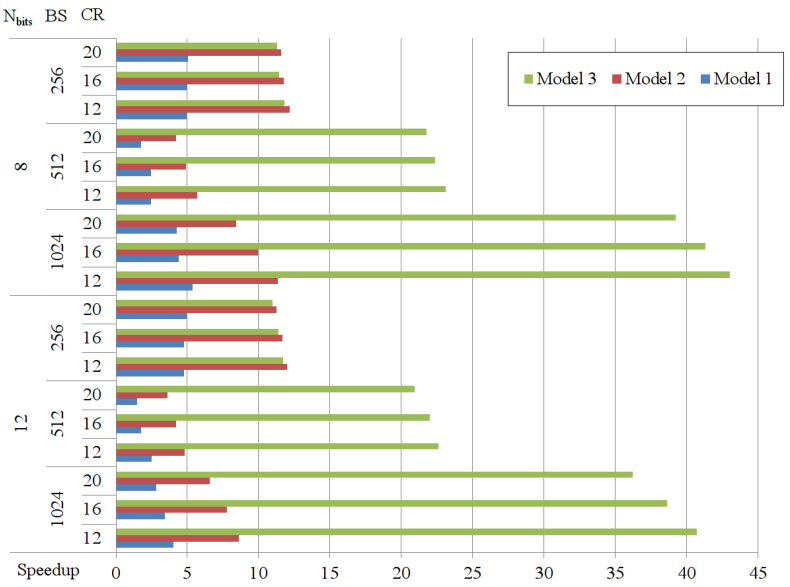


FIGURE 6.20: Speed-up obtained by the described parallel models with respect to the reference version of the HyperLCA compressor in the NVIDIA Jetson TK1 board.

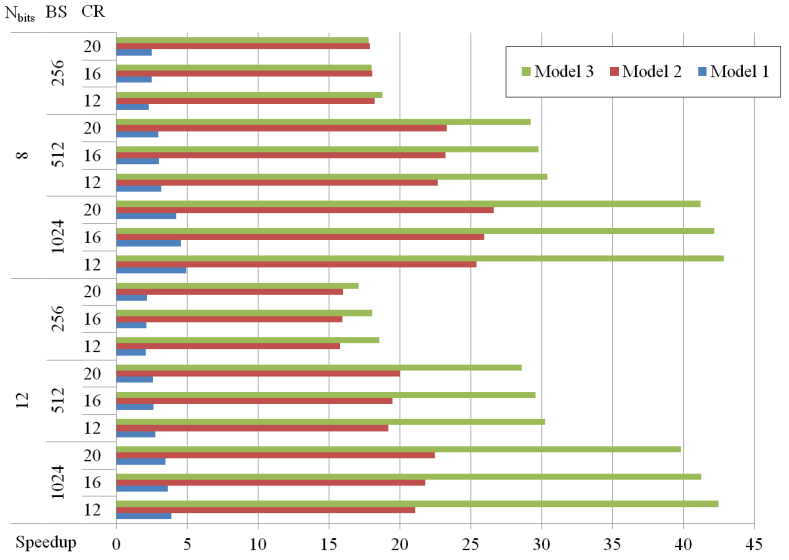


FIGURE 6.21: Speed-up obtained by the described parallel models with respect to the reference version of the HyperLCA compressor in the NVIDIA Jetson TX2 board.

Further conclusions can be drawn by making a deeper analysis of these results. As explained in Section 6.4.2.2, *Model 1*, *Model 2* and *Model 3* introduce different levels of parallelism, as shown Figures 6.17, 6.18 and 6.19, respectively. These parallelism strategies reduce the time required for compressing the hyperspectral data. However, for being able to apply these parallel models and use the LPGPUs, the image blocks as well as the *HyperLCA Transform* outputs need to be copied to different parts of the memory.

These copies are not needed in the reference model, and hence, negatively affect the obtained speed-up. Accordingly, the obtained speed-up corresponds to the time saved by applying the different parallelization strategies minus the time spent in making the copies of the image blocks and the *HyperLCA Transform* outputs. The time spent in each of these tasks depends on the specified configuration of the HyperLCA compressor and the characteristics of the Jetson TK1 and Jetson TX2 boards.

The impact of the N_{bits} parameter is not so high in the achieved speed-up. Basically, this parameter is used for calculating the p value, which determines the number of iterations of the *HyperLCA Transform* and hence, the number of times that the kernels are launched. In this sense, $N_{\text{bits}} = 8$ produces higher p values than $N_{\text{bits}} = 12$. The time saved in each kernel execution should be the same with independence of the number of times that the kernel is executed, hence, the speed-up should also be the same. However, the transfers of the image blocks and the *HyperLCA Transform* outputs are executed just once per image block. Due to this reason, the time lost in the memory transfers is proportionally smaller in relation with the time required by the *HyperLCA Transform* for processing the data when more iterations are to be executed (higher p values). Accordingly, the achieved speed-up is slightly higher for $N_{\text{bits}} = 8$ than for $N_{\text{bits}} = 12$. Something similar happens with the CR input parameter. Higher CR values result in lower p values and less iterations to be performed by the *HyperLCA Transform* for each image block.

Additionally, the CR and N_{bits} values also affect the number of outputs given by the *HyperLCA Transform*. Therefore, higher CR and N_{bits} values slightly decrease the amount of data to be copied for each image block. Due to these reasons, the variations in the speed-up obtained when using different CR and N_{bits} values will depend in how much time is saved in the *HyperLCA Transform* operations using these parallel models and how much time is lost in the memory transfers. This also depends on the characteristics of the hardware. For instance, higher CR values provide faster results in the Jetson TX2 board when using the *Model 2*, while the opposite happens for the Jetson TK1 board using the same parallel model. This, together with the fact that the speed-up obtained by the *Model 2* in the Jetson TX2 is much higher than the speed-up obtained with this parallel model in the Jetson TK1, suggests that the time spent in memory transfers has a higher impact in the Jetson TK1 than in the Jetson TX2 board.

Finally, the compression processes are pipelined as well as the memory transfers in the *Model 3* and hence, the time spend in memory transfers can be neglected. This model

results in the highest speed-ups, specially for bigger image blocks ($BS = 1024$). Additionally, the speed-up obtained by the *Model 3* is much higher than the speed-up obtained by the *Model 2* for bigger image blocks ($BS = 1024$), but it is almost the same for smaller blocks ($BS = 256$) in both boards. It is because when a kernel or a memory transfer is launched, the consumed time is not only spent on executing the kernel/transferring the data itself but also, on setting up and launching the instructions to do this. This means that the time consumed in transferring image blocks of 256 pixels is negligible compared with the overhead of launching the memory transfers and the additional logic required in the *Model 3*. Nonetheless, the time required for transferring image blocks of 1024 pixels is considerably higher than the overhead of initializing the copy and the additional logic required by the *Model 3*. On the basis that Model 2 and 3 only differ in the pipeline of the data transfers, we can conclude that bigger the BS, better the performance of the *Model 3* in terms of speed-up than the *Model 2*, since it better hides the extra time required to transfer bigger image blocks.

6.4.3.2 Performance of the parallel implementations of the HyperLCA compressor in embedded LPGPUs in terms of average compression frame rates

In addition to the speed-up analysis made in preceding Section 6.4.3.1, we have also evaluated the average number of hyperspectral frames that the developed parallel models are theoretically able to compress using the NVIDIA Jetson TK1 and Jetson TX2 boards. Table 6.10 shows the average results obtained after the compression of the HSIs that compose the test-bench introduced in Section 6.2.1 for the same settings of the HyperLCA input parameters considered in the previous Section 6.4.3.1. These frame rates are estimated by a rule of three using the average execution times required for processing each HSI within the defined test bench. Therefore, the average frame rate can be approximated by dividing these runtimes by 825 and rounding up to the nearest entire whole number.

As described in Section 6.4.3.1, the average compression frame rate is a key factor for the utility of this research work in the targeted application. Ideally, the desirable frame rates to be obtained by the developed implementations of the HyperLCA compressor should be higher than 330 FPS. This is due to the fact that the employed hyperspectral sensor is able to provide up to 330 FPS when collecting 224 spectral bands, and even more when collecting less number of spectral bands. By achieving this frame rate, it can

be guaranteed that the collected hyperspectral data can be always compressed in real-time and, the capturing frame rate can be specified according to the characteristics of the application (such as flight height, intensity of the light, area to cover, etc.) without being limited by the data compression process. It must be mentioned that the desirable frame rate represents the most extreme case since the employed frame rates are normally smaller. For instance, the images included in the test bench were collected at 150 and 200 FPS.

According to the results shown in Table 6.10, it can be concluded that the parallel implementation of the HyperLCA compressor named *Model 3* executed in the Jetson TX2 board is able to achieve more than 330 FPS with independence of the configuration used. It is also observable that using bigger N_{bits} values, ($N_{\text{bits}} = 12$), always guarantee faster compression results (higher FPS). This is due to the fact that less $\mathbf{V}$ vectors are extracted by the *HyperLCA Transform*, which results in less iterations required by the *HyperLCA Transform* and less output vectors to be copied back to the host. Additionally there are less vectors to be processed by the entropy coder. *Model 2* executed in the Jetson TX2 board is also able to achieve more than 330 FPS for some configurations, specially for higher compression ratios ($CR = 20$ and $CR = 16$). The fact of obtaining faster results for higher compression ratios represents an extra advantage, since it means the rise of the amount of compressed data to be stored and/or transmitted. It is also worth to mention that compression frame rates higher than 330 FPS are only achievable using *Model 3* when using the Jetson TK1 developer kit. As it occurs with the Jetson TX2 board, the fastest results are obtained using $N_{\text{bits}} = 12$, higher compression ratios and bigger image blocks. Despite the achieved frame rates using *Model 2* and *Model 3* do not surpass the desired 330 FPS for all the situations, specially in the Jetson TK1 board, the achieved FPS are still relatively high in relation with the frame rates typically used in the targeted application.

On the basis that *Parallel Model 3* achieves the highest speed-up and compression frame rates, we have also assessed the performance of this hardware implementation model on other two embedded computing devices, that is, the Jetson Nano and the Jetson Xavier-NX. These modules have been selected for the reasonable computational power provided at a relatively low power consumption. Therefore, the Jetson TK1 module integrates the less advanced, oldest generation of the four targeted GPU architectures, followed by the Jetson Nano. On the contrary, the Jetson Xavier-NX module represents one of the latest NVIDIA power-efficient products, which offers more than 10x the performance of its widely adopted predecessor, the Jetson TX2 board.

			Jetson TK1								Jetson TX2							
Input parameters			Time (s)				Frame Rate (FPS)				Time (s)				Frame Rate (FPS)			
N_{bits}	BS	CR	Ref.	Mod.1	Mod.2	Mod.3	Ref.	Mod.1	Mod.2	Mod.3	Ref.	Mod.1	Mod.2	Mod.3	Ref.	Mod.1	Mod.2	Mod.3
12	1024	12	84.32	20.96	9.78	2.07	12	49	105	495	77.91	20.02	3.70	1.83	13	51	277	558
		16	65.66	19.07	8.43	1.70	16	54	122	603	60.66	16.72	2.78	1.47	17	61	368	697
		20	53.26	18.99	8.10	1.47	19	54	127	697	49.16	14.19	2.19	1.24	21	72	468	829
	512	12	71.96	28.86	15.01	3.18	14	35	72	322	65.41	23.88	3.41	2.16	16	43	300	473
		16	59.44	33.78	14.11	2.70	17	30	76	379	53.92	20.45	2.77	1.82	19	50	370	561
		20	46.93	31.45	12.92	2.24	22	33	87	457	42.50	16.56	2.12	1.49	24	62	483	689
	256	12	59.78	12.53	4.97	5.11	17	82	206	201	53.94	25.99	3.42	2.91	19	39	300	352
		16	47.09	9.86	4.03	4.14	22	104	254	248	42.39	20.13	2.66	2.35	24	51	385	436
		20	34.40	6.94	3.06	3.14	30	148	335	327	30.87	14.26	1.93	1.81	33	72	530	567
8	1024	12	114.05	21.25	10.04	2.65	9	48	102	386	105.21	21.35	4.14	2.46	10	48	247	417
		16	89.50	20.33	8.98	2.17	11	50	114	473	82.60	18.10	3.18	1.96	12	57	322	523
		20	71.10	16.65	8.43	1.81	14	62	122	565	65.63	15.55	2.46	1.59	16	66	416	643
	512	12	95.88	39.31	16.82	4.14	11	26	61	247	87.25	27.38	3.85	2.87	12	37	266	357
		16	71.17	28.93	14.51	3.18	14	35	72	322	64.78	21.58	2.79	2.18	16	48	367	471
		20	58.81	33.54	14.02	2.70	17	31	76	379	53.43	18.17	2.29	1.83	19	57	448	560
	256	12	71.65	14.41	5.89	6.05	14	71	174	169	64.82	28.53	3.56	3.45	16	36	288	296
		16	52.86	10.61	4.49	4.62	19	97	228	221	47.68	19.14	2.64	2.65	21	54	388	387
		20	46.60	9.24	4.03	4.12	22	111	254	248	41.99	16.93	2.35	2.36	24	61	436	434

TABLE 6.10: Evaluation of the average execution times and frame rates obtained by the NVIDIA Jetson TK1 and the Jetson TX2 boards.

Table 6.11 collects the average results in terms of the compression frame rate obtained by the four Jetson boards executing the *Parallel Model 3*. As it can be seen, the Jetson Xavier-NX clearly outperforms the other targeted devices regardless the HyperLCA input parameter settings. Nevertheless, it should be noticed by the reader that the Jetson Xavier-NX represents one of the latest, most advanced NVIDIA single-board computers. In general terms, more than 200 FPS can be processed using any of the computing boards contemplated in the studio for almost all targeted algorithm configurations. In this sense, compression rates higher than 330 FPS are obtained for bigger image blocks ($BS = 1024$) and N_{bits} ($N_{bits} = 12$). In addition, these configurations also get the best possible results in terms of the quality of the decompressed data, as it was analysed in Section 4.4.5 of Chapter 4 using different distortion evaluation metrics, such as, the *Signal-to-Noise Ratio (SNR)*, the *Root Mean Squared Error (RMSE)*, the *Maximum Absolute Difference (MAD)* and the *Structural Similarity Index* (SSIM).

Input parameters			FPS			
N_{bits}	BS	CR	Jetson TK1	Jetson Nano	Jetson TX2	Jetson Xavier-NX
12	1024	12	495	533	558	2062
		16	603	638	762	2422
		20	697	729	923	2682
	512	12	322	351	506	1405
		16	379	408	599	1582
		20	457	487	748	1767
	256	12	201	225	372	909
		16	248	275	465	1052
		20	327	357	620	1251
8	1024	12	386	421	452	1730
		16	473	511	568	2020
		20	565	606	700	2294
	512	12	247	276	384	1149
		16	322	350	511	1409
		20	379	409	596	1574
	256	12	169	192	309	794
		16	221	249	414	973
		20	248	276	463	1053

TABLE 6.11: Evaluation of the average frame rates obtained by the NVIDIA Jetson TK1, the Jetson Nano, the Jetson TX2 and the Jetson Xavier-NX boards using the *Parallel Model 3*.

6.5 Benchmarking between the different parallel devices for the acceleration of the HyperLCA algorith

Among the research goals to be reached with the realization of this Thesis, it was contemplated the evaluation of the different parallel devices considered in the analysis made according to the characteristics of the target application. Consequently, this Section also provides a comprehensive analysis between the HWaccs targeting the HyperLCA implementation on FPGAs and LPGPUs developed in this Thesis. For the latter, we exclusively focus on the *Parallel Model 3* since it achieves the highest speed-up, especially for bigger image blocks ($BS = 1024$), which is indeed the only configuration evaluated in Section 6.3 for the FPGA-based HWaccs. Two assessment metrics have been selected for the comparison: the number of frames compressed in a second (FPS) and the power efficiency in terms of FPS per watt. The latter figure of merit is of great importance given the target application, since it is critical to maximize the battery life of a UAV-based system as the drone used for collecting the data set employed in the experiments made.

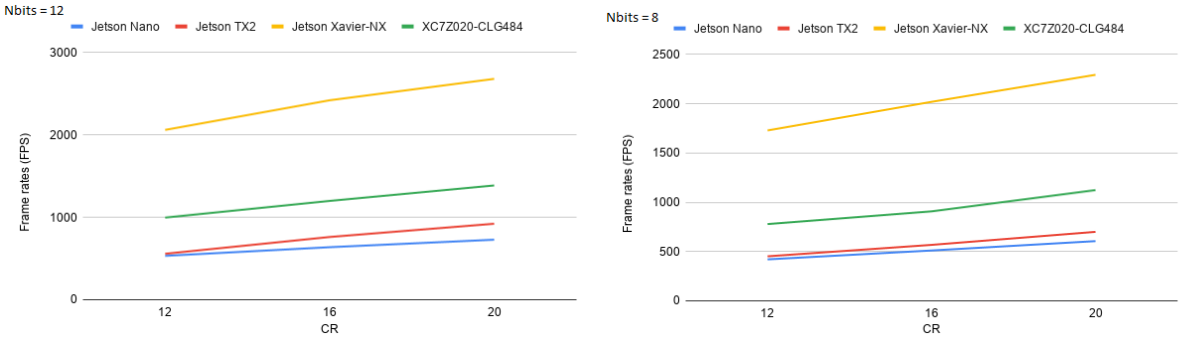
Regarding the GPU-based *Parallel Model 3*, its performance was evaluated on different NVIDIA Jetson embedded computing devices in Section 6.4. However, the analysis made in the following lines will be focused solely in the Jetson Nano, the Jetson TX2 and the most recent supercomputer, the Jetson Xavier NX. The Jetson Nano module integrates the less advanced, oldest generation of the three GPU architectures, instantiating the fewer execution units or CUDA cores as well. On the contrary, the Jetson Xavier-NX represents one of the latest NVIDIA power-efficient products, which offers more than 10x the performance of its widely adopted predecessor, the Jetson TX2.

Table 6.12 collects the performance results obtained for the three GPU-based implementations of the HyperLCA and the most powerful implementation of the HWacc in a Zynq-7020 SoC ($PE = 20$) for different settings of the algorithm input parameters. It is also specified the clock frequency and power budget for each addressed implementation. Additionally, Figure 6.22 displays the obtained frame rates according to different configurations of the N_{bits} parameter and the minimum desirable compression ratio, CR. As expected, the achieved frame rates increase when higher CR are desirable for all target devices. However, the FPGA-based implementation on a Xilinx Zynq-7020 SoC surprisingly outperforms those performed on the Jetson Nano and the Jetson TX2 boards. On the contrary, the GPU-based implementation on the Jetson Xavier-NX clearly exceeds the maximum number of FPS achieved by the FPGA for all algorithm settings. Nevertheless, it should be noticed that the Jetson Xavier-NX represents one of the latest, most advanced NVIDIA single-board computers whereas the Xilinx Zynq-7020 SoC that mounts the ZedBoard (i.e., XC7Z020-clg484) is a mid-range FPGA several technological generations behind. In fact, there are more powerful FPGA devices currently on the market, nonetheless, one of the main purpose of this comparison is to explore the minimum requirements of an FPGA-based computing platform that is able to fulfil the performance demands and constraints of the hyperspectral application under study. Thus, the selected version of the Xilinx Zynq-7020 SoC meets the demand for all algorithm configurations, at a lower cost as it will be seen in following lines.

Going deeper in the analysis of the results, Figure 6.23 plots the power efficiency for each target device, measured as the FPS achieved divided by the average power budget. In this sense, the Jetson boards are designed with a high-efficient Power Management Integrated Circuit that handles voltage regulators, and a power tree to optimize power efficiency. According to [233–235], the typical power budgets of the selected boards amount to 10 W, 15 W and 10 W for the Jetson Nano, Jetson TX2 and Jetson Xavier NX modules, respectively. In the case of the XC7Z020-clg484 FPGA, the estimated power consumption

N_{bits}	BS	CR	Maximum frame rate (FPS)			
			FPGA	GPU		
			XC7Z020-CLG484	Jetson Nano	Jetson TX2	Jetson Xavier-NX
			143 MHz 2.51W	921.6 MHz 10W	1.12 GHz 15W	800 MHz 10W
12	1024	12	998	533	558	2062
		16	1200	638	762	2422
		20	1388	729	923	2682
8	1024	12	779	421	452	1730
		16	908	511	568	2020
		20	1124	606	700	2294

TABLE 6.12: Maximum frame rates obtained in the compression process by a Xilinx Zynq-7020 programmable SoC and some NVIDIA power-efficient embedded computing devices, such as the Jetson Nano, Jetson TX2 and Jetson Xavier-NX.



(a) (b)

FIGURE 6.22: Comparison of the speed-up obtained in the compression process, in terms of FPS and the input parameter CR, reached by a Xilinx Zynq-7020 programmable SoC and some NVIDIA power-efficient embedded computing devices, such as the Jetson Nano, Jetson TX2 and Jetson Xavier-NX. (a) $N_{bits} = 12$, $BS = 1024$. (b) FPS $N_{bits} = 8$, $BS = 1024$.

after *Place & Route* stage in the Vivado toolchain goes up to 2.59 W at 143 MHz. Based on the trend lines shown in Figure 6.23, it can be concluded that the FPGA-based platform is by far more efficient in terms of power consumption that the target Jetson boards, for all algorithm configurations. Nonetheless, this gap has been decreased compared with the latest Jetson Xavier-NX. The reason that explains this behaviour roots in the fact that GPU-embedded platforms have been able to significantly increase their performance while maintaining or even reducing the power demand. The combination of architectural improvements and better integrated system (IC) manufacturing processes have paved the way to an scenario where embedded-GPU platforms are gaining ground and could be seen as competitors of FPGAs concerning power efficiency in the near future.

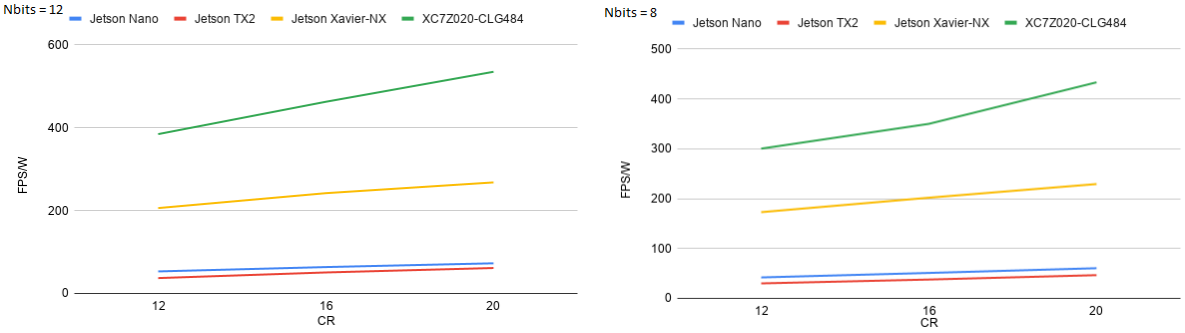


FIGURE 6.23: Comparison of the energy efficiency in the compression process, in terms of the ratio between obtained FPS and power consumption and the input parameter CR, reached by a Xilinx Zynq-7020 programmable SoC and some NVIDIA power-efficient embedded computing devices, such as the Jetson Nano, Jetson TX2 and Jetson Xavier-NX. (a) $N_{bits} = 12$, $BS = 1024$. (b) FPS $N_{bits} = 8$, $BS = 1024$.

6.6 Conclusions

In this Chapter, we tackle two research goals within those defined initially for the realization of this Thesis. Firstly, it has been verified the suitability of the algorithms proposed along the preceding Chapters for real-time applications. Secondly, it has also confirmed the benefits of developing algorithmic solutions based on the same mathematical method in terms of reducing the execution-times, the hardware resources and the human endeavours. For doing this, the algorithmic solutions proposed in this Thesis, that is; the HyperLCA, the HW-LbL-FAD and the HADeLoC methods; have been efficiently implemented in a heterogeneous mid-range Zynq-7000 SoC chip (XC7Z020-clg484) from Xilinx. Additionally, the HyperLCA algorithm has been also accelerated in low-power GPUs (LPGPUs) embedded in NVIDIA boards in order to evaluate the adaptability of the proposed set of core operations to other scenarios according to the characteristics of the targeted applications.

In order to stablish the real-time constraints to be fulfilled, we focus on remote sensing applications where the available computational resources are limited, due to power, weight or space limitations. Concretely, we present a smart farming application where a VNIR hyperspectral pushbroom scanner is mounted onto a UAV for collecting periodical information of the crops, which results in a huge amount of data that needs to be managed, processed and analysed. In this sense, the developed hardware accelerators must reach frame rates higher than 330 FPS, being a frame composed of 1024 hyperspectral pixels, in order to surpass the maximum frame rates provided by the acquisition system. By

achieving this frame rate, it can be guaranteed that the collected hyperspectral data can be always processed in real-time and, the capturing frame rate can be specified according to the characteristics of the application (such as flight height, intensity of the light, area to cover, etc.) without being limited by the data processing methods.

The definition of the FPGA-based HWaccs that implement the several algorithms proposed in this Thesis has been carried out by using a combination of generated HLS modules and custom glue logic in VHDL. In this sense, FPGA-based modules that implement each of the proposed core operation are defined using HLS tools. These defined modules are reused in the hardware implementation of the HW-LbL-FAD, the HyperLCA and the HADeLoC methods. Consequently, efforts have been focused on the interconnections among them for each targeted algorithm to be accelerated. In this sense, memory buffers and custom logic that integrates and orchestrates all the components in the design are instantiated and implemented using customized VHDL language. Therefore, it implies less time and effort during the stage of hardware acceleration since the same product can be reused for several algorithms targeting different applications. Additionally, FPGA devices are generally more efficient dealing with integer operations with a close-to-hardware programming approach. For this reason, the *Int16-rd* version of the proposed set of core operations has been selected for the definition of the HWaccs under evaluation, due to its good behaviour in terms on the quality and the precision of the obtained results and the resource savings it brings.

Regarding the addressed LPGPU-based implementation of the HyperLCA compressor, the parallelism inherent to this hyperspectral lossy compressor has been exploited beyond the thread-level concurrency of the GPU programming model and more specifically, pipelining the execution of the compression stages of the HyperLCA algorithm for each independently processed image block, as well as, the communications and memory transfers. In this sense, three different implementation models of the whole compression model have been studied, seeing them as an evolution towards an optimal configuration that fulfils the constraints imposed by the targeted application. In particular, the third approach, referred to as *Parallel Model 3*, achieves the highest speed-up, especially for bigger image blocks ($BS = 1024$). This implementation model bets for pipelining the data transfers between the host and the device and the kernel executions for the different image blocks, $\mathbf{M}_k$. To do this, such proposal exploits the benefits of the concurrent kernel execution through the management of CUDA streams. Unlike the FPGA-based HWacc also

described for this particular algorithm, only the *HyperLCA Transform* stage is accelerated in the LPGPU. In this sense, the codification stage is pipelined with the *HyperLCA Transform* stage but it is executed on the Host using another parallel CPU process.

Several experiments have been carried out in order to evaluate the performance of the hardware accelerators described in this Chapter targeting both the FPGAs and the GPUs. In all cases, developed implementations have overcome by far the requirements imposed by the maximum capturing frame rate of the hyperspectral sensor. Regarding the FPGA-based HWacc, a map-reduce programming model that permits the parallel management of several spectral bands (PE) is able to obtain between 778.90-1600 FPS with PE = 20. For the three targeted HWAccs, the employed hardware resources in terms of BRAM, DSPs, FFs and LUTs do not exceed the limits imposed by the targeted device. Indeed, it is used the 70.36% of the BRAM, the 75.45% of the DSPs, the 21.71% of the FFs and the 45.04% of the LUTs for the most unfavourable situation. Nonetheless, this scenario corresponds with the HADeLoC approach that actually carries out both the lossy compression process and the detection of anomalous spectra.

From the obtained results, it can be also drawn two additional conclusions. Firstly, it is clear that the implementation of both the HW-LbL-FAD detector and the HyperLCA compressor could not be addressed as independent entities in the selected XC7Z020-clg484 FPGA due the available hardware resources. Nonetheless, this issue is solved by the formulation of algorithmic solutions based on the same mathematical method in order to reuse block of operations among the targeted analysis, as the HADeLoC does. Secondly, the execution times required for the implementation of both the lossy compression of HSIs and the detection of anomalous spectra handled by the HADeLoC approach is almost the same as just the performance of only one process, specially, the HW-LbL-FAD algorithm. Therefore, these discussions are certainly in the line of the research goals to be accomplished in this Thesis.

In relation to the implementation of the HyperLCA algorithm on NVIDIA boards, it can be concluded that the parallel *Model 3* executed in both the Jetson TX2 and the Jetson Xavier NX boards is able to achieve more than 330 FPS with independence of the input parameter settings. The frame rates obtained by the former reaches the 452-923 FPS while the latter is able to process 1730-2682 FPS. As it can be seen, the newest embedded-GPU platforms, such as the Jetson Xavier NX, are gaining ground and they can be seen as competitors of FPGAs concerning both the computing performance and even power efficiency [217].

Finally, it is important to note that although experiments carried out in this work are oriented to the current necessities imposed by an application based on drones, all drawn conclusions can be extrapolated to other fields in which remotely sensed hyperspectral images have to be compressed in real time, such as spaceborne missions that employs next-generation space-grade FPGAs.

Chapter 7

Conclusions and further research lines

This Chapter summarizes the main contributions of this Thesis and proposes further research topics, which are expected to complement and enhance the future developments of this work.

7.1 Conclusions

Hyperspectral imagery has gained an increasing interest by the scientific community in the last years in such a manner that it has been consolidated as one of the mainstream terrestrial Earth observing systems. Its attraction lies in the large amount of information along the electromagnetic spectrum that this technology brings for each single image pixel. Human eyes are sensitive to the visible light in the way that they are able to distinguish mostly three wavelength ranges corresponding to the red, green and blue colours. Nonetheless, hyperspectral images (HSIs) are able to capture the reflection distribution from the observed objects along a broader range of the electromagnetic spectrum dividing it in many contiguous spectral bands. Therefore, any single image pixel is associated with a full continuous spectrum, commonly called spectral signature of the pixel. This in fact permits the identification of certain Earth surface materials due to their unique "fingerprint" along the electromagnetic spectrum.

On this basis, remote sensing is the field in which hyperspectral imaging techniques have traditionally acquired more relevance. In this sense, the hyperspectral data analysis enables detailed examinations of the land surfaces and the identification of visually similar materials, as well as the estimation of physical parameters of many complex surfaces. Within this domain, three different hyperspectral acquisition platforms have been extensively utilized, such as aeroplanes, satellites and unmanned aerial vehicles (UAVs). Indeed, the latter is gaining momentum and has become a very popular solution for the data collection in applications oriented to the inspection, surveillance and monitoring due to its lower-cost and more flexible revisit time than the other aforementioned Earth-observation platforms.

Nevertheless, despite the existence of a wide variety of remote sensing data acquisition systems, the onboard hyperspectral image processing still poses several challenges mainly due to the management of large amounts of data. On one hand, it jeopardizes the real-time performance of this kind of applications and, on the other hand, it demands a significant computing capability, which also means a greater degree of hardware resources utilisation. Additionally, the steadily growing data-rate and resolution of the latest-generation sensors aggravate the issue even further. Consequently, on-Earth processing has been the mainstream solution for remote sensing applications that sense HSIs. In this regard, images acquired by Earth observation platforms aboard satellites or manned/unmanned

aerial vehicles are traditionally downloaded to the ground segment for being off-line processed on supercomputing systems. Unfortunately, the data transmission from the remote sensing platforms to the Earth surface introduces important delays related to the communication of large amount of data. Consequently, this operating mode clearly compromises the real-time response of time-sensitive applications due to the large amount of data to be transmitted and the bottleneck represented by the limited communication bandwidth of the downlink system.

Against this backdrop, the onboard processing of remotely sensed HSIs has experienced a steady surge in popularity in recent years. Indeed, it represents a potential solution for those applications that demand quick response in which the conventional approaches based on the off-line data handling do not meet the actual needs. Nevertheless, it is still necessary to overcome many other obstacles imposed by the available aboard hardware devices in order to efficiently carry out this alternative and to execute onboard the hyperspectral data processing. Regrettably, the algorithms traditionally proposed for the hyperspectral analysis have been addressed as independent entities, using those mathematical methods that better maximize the results for each particular case. In addition, these approaches normally give rise to complex algorithms characterized by computationally costly operations, intensive memory requirements, high implementation costs and a non-scalable nature. For these reasons, a great deal of effort has been made to reduce the heavy computational burden required by the involved operations in order to accelerate the process through alternative mathematical methods or through parallel computing via high-performance architectures.

The aforementioned scenario becomes even more challenging when different time-sensitive applications coexist in the same computing device. In this regard, the simplest and the most commonly adopted solution is to select a different mathematical algorithm from the wide assortment of proposals encountered in the literature for each hyperspectral image processing type to be performed and then, to accelerate them using parallel computing devices. The issue arises when they have to be sequentially processed onto the same computing device due to restrictions in terms of power, weight and size. The main rationale behind this is based on the detaching between the algorithm design phase and the hardware implementation stage, resulting in very inefficient hardware implementations. Therefore, there is a need in the literature for new algorithmic solutions that take into consideration the above mentioned currently existing constrains imposed by nowadays remote sensing applications. Additionally, the causality inherent to real-time frameworks based on pushbroom/whiskbroom scanners must be also met through the definition of

non-global algorithms capable of independently processing blocks of image pixels. In turn, this prevents the storing and management of large data volumes, thereby reducing the computing resources and speeding up the execution process.

Against this backdrop, we have dealt in this Thesis work with the issue around the on-board execution of multiple hyperspectral image analysis techniques onto the same piece of hardware. In this regard, a new algorithmic solution is proposed to meet the aforementioned requirements, paving the to a real-time performance of the hyperspectral image processing. For this end, this Thesis contributes to the research community with the achievements described below:

- A set of core operations that extract features from the HSIs useful for many applications has been uncovered. This set permits the concurrent execution of many different tasks at the same time; such as anomaly detection, target detection, lossy compression, classification, and unmixing; with the advantage of sharing the most computationally intensive operations. To do this, the proposed set of core operations is based on orthogonal projection techniques and more specifically, on the well-known Gram-Schmidt orthogonalization method. Additionally, this methodology also features low computational complexity since non-complex matrix calculations are involved and previously computed information is reused.

  As a novelty, the proposed set of core operations can be efficiently and independently applied on blocks of image pixels without requiring any specific spatial alignment. This distinguishing feature makes this proposal a promising solution for real or near real-time applications specially when using hyperspectral sensors based on pushbroom scanners, which sense the data in a line-by-line fashion.

  One of the major benefits of the introduced core operations lies in the definition of a set of variables whose values are always within numeric ranges known beforehand. Consequently, it allows to fix in advance the maximum and minimum values of the results obtained in each operation. This feature makes possible to use the fixed-point concept in a custom way using integer arithmetic and bit shifting for representing the integer and decimal parts of the numbers. Therefore, the proposed set of core operations can be easily adapted to the requirements imposed by the targeted devices and thereby, be seamlessly implemented using both fixed-point and floating-point notation. In this context, field programmable gate array (FPGA) devices are in general more efficient dealing with integer operations with a close-to-hardware programming approach, while graphical processing units (GPUs) are

optimised for parallel processing of single floating-point operations using thousands of small cores.

- A new algorithm for the detection of anomalous spectra has been also developed in this Thesis, named *A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery* (LbL-FAD). The LbL-FAD algorithm is a subspace-based anomaly detection algorithm designed to fulfil the constraints imposed by nowadays remote sensing applications based on pushbroom/whiskbroom scanners. In this regard, the LbL-FAD algorithm is able to independently process blocks of hyperspectral pixels with not taking into consideration any spatial alignment requirement. As a result, this feature is well aligned with the needs imposed by the aforementioned acquisition systems since the detection of anomalous pixels can be conducted in a line-by-line fashion.

  For the detection of abnormal spectra, the LbL-FAD algorithm focuses on the calculation of an orthogonal subspace to the one spanned by the background distribution in which the anomalous spectra are better distinguishable. To do this, the LbL-FAD follows an orthogonal projection strategy and, more specifically, by means of the set of core operations introduced in this Thesis. Therefore, this methodology allows excluding the use of traditional linear transformation methods, such as the Principal Component Analysis (PCA) or the Singular Value Decomposition (SVD), that are high computational complex in nature, nor the inverse of big data matrices.

- A performance-enhancing version of the state-of-the-art *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) has been proposed for the spectral decorrelation and compression of HSIs. The HyperLCA algorithm is a low-computational complexity transform-based alternative that provides high compression ratios with a good compression performance at a reasonable computational burden. As a further advantage, the HyperLCA algorithm permits compressing blocks of image pixels independently. This feature promotes, on one hand, the reduction of the data to be managed at once besides the hardware resources to be allocated, and on the other hand, the HyperLCA algorithm becomes a very competitive solution for most applications based on pushbroom/whiskbroom scanners, paving the way to a real-time compression performance.

  The HyperLCA algorithm was for the first time introduced in [2]. As a novelty, the methodology described in [2] has been widened in this Thesis work in order to

be adapted and, thereby, fallen within the proposed set of core operations. Moreover, an efficient and comprehensive compression system has been also introduced. Additionally, it has also extended the definition of the operations performed by the *HyperLCA Transform* with the purpose of analysing their suitability for being executed using integer arithmetic.

Finally, it has also been demonstrated that the HyperLCA algorithm is able to preserve the most different pixels after the compression-decompression process, which is crucial for many posterior hyperspectral image processing techniques. Indeed, the use of the same core operations for the definition of both the HyperLCA compressor and the above-mentioned LbL-FAD algorithm guarantees that the compression process does not seriously affect the posterior anomaly detection performance when it is off-board executed using the compressed/decompressed data. As a consequence, it permits tailoring to different scenarios that impose different requirements, ensuring the same results in all situations.

- The feasibility of the concurrent execution of multiple hyperspectral analysis techniques based on the same mathematical method has been also demonstrated. Concretely, we have analysed the adequacy of the set of core operations proposed in this Thesis work for the simultaneous execution of multiple hyperspectral analysis techniques. This provides two main benefits. Firstly, it implies less time and effort during the stage of hardware acceleration since the same product can be reused for several algorithms targeting different applications. Secondly, it permits the execution of several tasks at the same time with the advantage of sharing the most computationally costly operations, thus reducing the overall computational cost and the required hardware resources.

  In particular, it was verified the suitability of the proposed methodology for the concurrent execution of the lossy compression of HSIs jointly with the detection of anomalous signatures. To meet this issue, two optimized versions were eventually proposed. The former, referred to as *Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (ADeLoC), searches for the highest accuracy in the detection and compression results. In this sense, the ADeLoC approach ensures the same detection and compression results as the original LbL-FAD and the HyperLCA methods but, launching 39-41% less number of operations. The latter, named *Hardware-friendly proposal for the simultaneous*

detection of anomalous pixels and the lossy compression of HSIs (HADeLoC) prioritizes the optimization of the hardware resources and the minimization of the execution times at the expense of a loss of accuracy in the compression results compared with the original HyperLCA compressor. In this case, it was verified that roughly 55-59% fewer operations are executed than if both processes were independently implemented and 27-30% less than the ADeLoC version. Therefore, the overall conclusion that was drawn is that there is always a trade-off among the quality of the results, the computational resources and the execution times.

- To confirm the benefits of developing algorithmic solutions based on the same mathematical method in terms of a reduction in the execution-times, the hardware resources and the human endeavours, and also to verify the suitability of the developed algorithms for real-time applications, the main proposals of this Thesis have been also implemented on different parallel devices, namely GPUs and FPGAs. Concretely, the LbL-FAD, the HyperLCA and the HADeLoC methods were accelerated on a Xilinx system on chip (SoC) FPGA device, while the HyperLCA was adapted to be launched in embedded computing boards from NVIDIA.

  The definition of the FPGA-based hardware accelerators (HWaccs) that implement the several algorithms proposed in this Thesis has been carried out by using a combination of generated high level synthesis (HLS) modules and custom glue logic in VHDL (Very High Density Language). In this sense, FPGA-based modules that implement each of the proposed core operation are defined using HLS tools. These modules are reused in the hardware implementation of the HW-LbL-FAD, the HyperLCA and the HADeLoC methods. Consequently, efforts have been focused on the interconnections among the described HLS modules particularly for each targeted algorithm to be accelerated. In this sense, memory buffers and custom logic that integrates and orchestrates all the components in the design have been instantiated and implemented using customized VHDL language. Therefore, it implies less time and effort during the stage of hardware acceleration since the same product can be reused for several algorithms targeting different applications. In relation to the average obtained processing rates, the developed HWaccs are able to manage between 778 and 1600 frames of 1024 hyperspectral pixels in a second (FPS), which surpass the minimum of 330 FPS imposed by the targeted application and hence, ensuring a real-time performance.

  Regarding the GPU-based implementation of the HyperLCA compressor, three different implementation models of the whole compression model have been studied,

seeing them as an evolution towards an optimal configuration that fulfils the constraints imposed by a high data-rate application. The last implementation model, referred to as *Parallel Model 3*, bets for pipelining the data transfers between the host and the device and the kernel executions for the different image blocks, $\mathbf{M}_k$. To do this, such proposal exploits the benefits of the concurrent kernel execution through the use of the CUDA stream concept. In terms of the obtained compression frame rates, this *Parallel Model 3* is able to ensure a real-time performance since 452-923 FPS can be efficiently processed using the Jetson TX2 developer kit, while more than 1730 FPS are obtained with the Jetson Xavier NX, which is also one of the latest embedded GPU NVIDIA platforms.

- Finally, we have also briefly discussed the possibility of extending the use of orthogonal subspace projections and, in particular, the Gram-Schmidt orthogonalization method by means of the set of core operations proposed in this Thesis work, in the fields of band selection, target detection, unmixing and classification. Although the analysis made is far from being as exhaustive as those carried out by LbL-FAD detector and the HyperLCA compressor, it indeed represents a turning point in the way of future research works.

7.2 Future Research Lines

Several topics could be derived from those dealt in this Thesis as part of future research works that extend and complement the conclusions drawn along this dissertation. Some of them are briefly introduced in below lines.

- Future research lines are focused on extending the use of the proposed set of core operations to other fields such as, band selection, target detection, unmixing and classification. Although the first approximations towards the realization of these hyperspectral image processing analysis are briefly illustrated in Appendix A, a more comprehensive study is required in order to set the strengths and weaknesses of the proposed methodology for these particular issues. This fact also opens up new avenues towards the concurrent execution of multiple hyperspectral imaging applications. Therefore, it is also desirable to analysis the viability of the simultaneous performance of some of the aforementioned methods in a similar way as it has been done in this Thesis for the anomaly detection and the lossy compression issues.

- The set of core operations proposed in this Thesis selects the p most spectrally different pixels as representative of a certain distribution of spectral signatures. Nonetheless, the presence of undesirable distortions derived from sparkles, scattering or from the sweeping motion of the data acquisition platform might lead to the selection of some pixels of interest that they are not really representative of the spectra contained in the data to be analysed. Hence, it could jeopardize the quality of the compression results and also the accurate modelling of the background distribution within the anomaly detection issue. Consequently, there is also a need for the development of outlier detectors that are able to discard these undesirable spectra and minimize the impact of the above-mentioned phenomena.

- Regarding the performance of the LbL-FAD anomaly detector proposed in this Thesis, it is also desirable to study other alternatives in order to estimate a more accurate threshold that automatically segments the abnormal spectra from the background. With it, we achieve to increase the detection probability of mixed pixels placed on the rounded edges of the anomalous entities. In this regard, these pixels are composed in its majority by background spectra mixed with the anomalous signature and for this reason, their detection is not straightforward.

- Nowadays, there are other types of entropy coders that the one used in this Thesis in conjunction with the *HyperLCA Transform* for the lossy compression of HSIs. Concretely, the entropy coder used in the HyperLCA algorithm description done in this dissertation is a sample-adaptive entropy coding approach. Nonetheless, the CCSDS organization also includes in its recommended standards other solutions, such as the block-adaptive coder and the hybrid encoding approach. Therefore, the analysis of these other existing solutions could be interesting in order to maximize the trade-off between the reached compression ratios and the rate-distortion performance.

- During the assessment of the proposed HADeLoC method, it was shown the importance of the calibration and the radiometric correction of the hyperspectral data to obtain accurate reflectance values before performing any further analysis on them. Indeed, it is a determining factor since luminance conditions over the course of the data acquisition campaigns are likely to bear the brunt of the environmental shifts. In addition, there is an increasing scientific motivation towards the definition of more accurate radiometric correction methods under operational conditions that take into account the incident radiance at each instant [31, 219–222]. Any improvements done

in this field will have repercussion in hyperspectral imaging applications under real-time constraints since they are actually the first step required in any hyperspectral imaging application.

- Further research work is being carried out at this moment related with the transmission of the acquired hyperspectral data or the results obtained after the onboard processing of these data. It allows the rapid download of the acquired data to the ground stations where an operator could analyse and visualize them in real-time. Currently, we are working on the design of a complete working solution that, starting from the GPU-based hardware accelerator described in this Thesis, it has been adapted for being incorporated in the acquisition platform described in [89] for testing the goodness of the HyperLCA compressor in a real operation mode [236]. However, we have also come across some operational limitations related with the use of non-volatile memory and the transmission process based on a wireless local area network (WLAN) via the Secure Shell Protocol (SSH) connection. Therefore, a great deal of work remains to be done in this field but, not only targeting the lossy compression of HSIs but also, the other algorithmic solutions proposed in this dissertation and other computing devices such as FPGAs.

Appendix A

Application of the proposed methodology to other hyperspectral image processing research fields

Future research lines are focused on extending the use of the set of core operations proposed in this Thesis work to other fields such as, band selection, target detection, unmixing and classification. In this Appendix, we briefly illustrate the first approximations towards the performance of these hyperspectral image analysis techniques.

A.1 Rationale

In preceding Chapters, the potential of the orthogonal projection techniques performed in the way as does the set of core operations defended in this Thesis work has been corroborated for the detection of anomalous spectra and the lossy compression of hyperspectral images (HSIs). Nonetheless, there are other relevant and distinctive hyperspectral data analysis techniques in which this methodology could be efficiently applied as well, such as unmixing, target detection, classification and band selection.

HSIs offer a high spectral resolution in return for a limitation in the spatial resolution, which is normally less than the size of most land object types and so, spectrally mixed pixels exist. Hence, the signal recorded by a hyperspectral sensor at a given band and from a given pixel is a mixture of the "light" scattered by the constituent substances located in the respective pixel coverage [55]. In this context, the hyperspectral unmixing paradigm provides a solution to the spectral mixing modelling issue, which consists of determining the spectrally pure components, also called endmembers, present in mixed pixels, as well as the amount of spectral information collected by each image pixel that can be represented by each endmember, also called abundances. The linear mixing model (LMM) is one of the most popular approximations to the mixing simulation. It assumes that measured spectra can be expressed as a linear combination of the endmember signatures present in the mixed pixel. On this basis, a HSI can be represented as a function of some image pixels, $\mathbf{E} = \{\mathbf{e}_n, n = 1, ...p\}$, and their corresponding abundances, $\mathbf{A} = \{\mathbf{a}_{j,n}, j = 1, ...np, n = 1, ...p\}$, which can be derived from the projection of each image pixel onto each endmember [105]. As it can be noticed, the set of core operations proposed in this Thesis actually addresses these issues as long as results in a set of the most characteristic or distintive pixels within a set of image blocks that are obtained by orthogonal projection techniques. Moreover, the linear hyperspectral unmixing process could be also a preliminary step for many other hyperspectral processes, such as classification and target detection, since it allows a better understanding of the scene under analysis.

Hyperspectral imagery has been used in reconnaissance and surveillance applications where targets of interest are detected and identified. In some applications, there is a prior knowledge about the spectral characteristics of the desired targets. In this scenario, target detection consists of searching these known spectral signatures of interest, for example from a database. In this sense, the mixing model can be used to characterize the targets and the interfering background. Indeed, a subpixel target is mixed with

the background spectra resulting in an image pixel with a combined spectral signature. Therefore, subpixel target detection issue could involve some kind of linear separation of pixel constituent elements [108] that could be solved by addressing the linear hyperspectral unmixing process. Additionally, unlike the spectral signature of interest that is known in advance, the background subspace could be estimated from the HSI using statistical or geometrical techniques. A good statistical approximation of the background can be done using the eigenvectors of the hyperspectral cube correlation matrix. On the contrary, the extraction of the p most representative pixels of the background, also commonly referred to as undesired signatures, could arise as an accurate geometrical description of the background distribution that will be used later to annihilate the spectral information that does not belong to the desired target. As it can be seen, this feature extraction issue is equivalent to the modelling of the background distribution carried out for anomaly detection and lossy compression, which was efficiently settled by the set of operations proposed in this Thesis.

Regarding the latter, orthogonal projection techniques may be also used to select the spectral bands that best differentiate the desired target and the background signatures in order to maximize the spectral differences between both classes. The proposed set of core operations is indeed able to identify the most characteristic pixels within a HSI. This makes us think that processing the transpose of the data applying this methodology could result in an accurate selection of the most distinctive spectral bands within spectral classes inherent to the data set instead of pixels. Therefore, the set of core operations could be potentially used to tackle feature selection, in particular, band selection, which would also ease the posterior target detection.

Lastly, classification of a HSI entails the identification of which pixels contain various spectrally distinct materials that have been specified by the user. As it can be noticed, the classification problem can be treated as a problem of abundance calculations as in the unmixing field where user-defined spectra can be seen as the endmembers. In this sense, each image pixel could be classified as a function of its spectral similarity with each reference spectra, which would be assessed by the maximum estimated abundance factor, a_n, [237].

The possibility of extending the use of the set of core operations proposed in this Thesis work to the above mentioned fields has been analysed in this Chapter. Although these hyperspectral data analysis themes have not been studied in as much detail as the detection of anomalies and the lossy compression of HSIs discussed in preceding Chapters, we

briefly illustrate the first approximations towards the performance of these hyperspectral processing techniques.

A.2 Dimensionality Reduction: band selection

Hyperspectral sensors gather spectral information of hundreds of continuous and narrow wavelengths along the electromagnetic spectrum. Its abundant spectral information provides the potential of accurate object identification. Nonetheless, this feature also involves the handling of large data volume, which brings about problems in data transmission and storage. Moreover, adjacent spectral bands are highly correlated and hence, spectral information contains in them is redundant. In this context, dimensionality reduction (DR) techniques emerge in order to reduce the very high-dimensional data inherent to the HSIs to a manageable low-dimensional space where data analysis can be performed in a more effective way [238].

In general terms, there are two main categories of DR methods found in the literature: transform-based and band selection-based methods. The former consists in transforming the data onto a low-dimensional space by means of certain criteria. Some examples of these approaches are the well-known Principle Component Analysis (PCA), Minimum Noise Fraction (MNF), Independent Component Analysis (ICA), Orthogonal Subspace Projection(OSP), among others. In addition, the *HyperLCA Transform* proposed in this Thesis work for the lossy compression of HSIs is also an example of this kind of DR methodologies. However, transform-based solutions normally change the physical meaning of the original data because components of the resulting low-dimensional space do not correspond to individual original spectral bands [239]. By contrast, selection-based methods search for a subset of original bands in pursuit of preserving the physical meaning of pixel spectra. It is on this latter that the set of core operations proposed in this Thesis work may be also potentially employed. While the goodness of this methodology was analysed as a data transform-based point of view in Chapter 4, its expanded use in the field of band selection-based approaches is briefly outlined in this Section.

The basic idea behind the unsupervised band selection methods is to find the most distinctive bands among a set of spectra that still allow the detection and discrimination of classes present in the original data. For this reason, state-of-the-art approaches proposed to search for distinctive spectral signatures in endmember extraction has been commonly

used to address the band selection issue, as it was analysed in [239]. The main difference lies in the domain in which data is analysed. In the case of band selection, algorithms are applied in the spatial domain instead of being applied in the spectral domain for endmember extraction. On this basis, the set of core operations proposed in this Thesis has been extensively analysed so far for the extraction of the most characteristic pixels, which play a similar role as endmembers. Therefore, it is straightforward to think that this methodology could be also applied for band selection after introducing some minor changes in the way the data are read, that is, using the transpose of the original data.

Figure A.1 shows a graphical representation of the variables and their dimensions involved by each proposed core operation when using the original hyperspectral data (Figure A.1a) for the extraction of characteristic pixels, $\mathbf{E}$, and the transpose of the data ($'$) for band selection (Figure A.1b). For the sake of clarity, a HSI is composed of np pixels that contain spectral information along nb wavelengths and hence, each pixel is represented as $\mathbf{r}_{ji}, j = 1, ..., np, i = 1, ..., nb$. When the selection of the most characteristic pixels, $\mathbf{E}$, is addressed, input data is arranged in such a way that image rows contain the nb elements of each image pixel, which are aligned in the image columns. On the contrary, the transpose of the data is used for band selection and hence, image rows contain the values of the np pixels for each spectral band, which are aligned in the image columns. Regarding the set of core operations, they are further analysed in what follows:

1. Average pixel calculation, $\hat{\boldsymbol{\mu}}$, and centralization of the input image:

   In the original version (see Figure A.1a), the average pixel, $\hat{\boldsymbol{\mu}}$, is a vector of nb components that contain the average value of the np pixels within the HSI for each particular spectral band, i. Conversely, when the transpose of the input image is used for band selection (see Figure A.1b), $\hat{\boldsymbol{\mu}}$ is now a vector of np components that contain the average value of the nb spectral bands within the HSI for each particular image pixel, j. As a consequence, $\hat{\boldsymbol{\mu}}$ in this latter configuration actually acts as a panchromatic image composed of the average values along the whole range of the electromagnetic spectrum covered by the sensor. Moreover, these sensors normally present a poorly spectral response for extreme spectral bands. Therefore, these spectral bands are far from being zero after the centralization and thus, have a high probability to be selected as distinctive spectral bands, which is in fact not desirable. Consequently, these two computing stages of the proposed set of core operations are not tackled for band selection.

2. Brightness calculation:

   When pixel selection is faced, the vector of brightnesses, $\mathbf{b}$, collects the square of the $l2\text{-}norm$ of each image pixel with itself, also named the brightness of a pixel. Therefore, the selected pixels, $\mathbf{e}_n$, are those with the highest $l2\text{-}norm$, $b_{j_{max}}$, and thus, $\mathbf{q}_n$ and $\mathbf{u}_n$ are vectors with nb components as well. Nonetheless, the vector of brightnesses, $\mathbf{b}$, for the second data configuration is composed of nb components that contain the brightness of each spectral band, i. Consequently, the maximum brightness, $b_{i_{max}}$, corresponds now with a spectral band rather than with a pixel. For this reason, $\mathbf{q}_n$ and $\mathbf{u}_n$ comprise np components instead of nb.

3. Projection vector, $\mathbf{v}_n$:

   As it has been mentioned along this manuscript, projection vectors, $\mathbf{v}_n$, collect the scalar values of the np pixels projected on the direction spanned by $\mathbf{u}_n$. However, $\mathbf{u}_n$ is a vector of np components in the proposed methodology for band selection that contains the np pixel values for the most distinctive spectral band above selected. Therefore, $\mathbf{v}_n$ collects the scalar values of the nb spectral bands projected on the direction spanned by the np components contained by $\mathbf{u}_n$.

4. Subtraction:

   As it can be noticed, $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors are in fact orthogonal vectors derived from image pixels when the original configuration of the input image is used for pixel selection. For this reason, the spectral information retained by each image pixel for the next iterations does not contain the information spanned by the direction of the already selected pixel. Consequently, $\mathbf{q}_n$ and $\mathbf{u}_n$ vectors are orthogonal vectors with each other, but not their components in the spectral domain. Nonetheless, $\mathbf{q}_n$ and $\mathbf{u}_n$ in the second case study contain the information of the np pixels spanned by the most distinctive spectral bands selected in each algorithm iteration. As a consequence, spectral information retained for the next iteration is orthogonal to the pixel information contained in the wavelength already selected and hence, $\mathbf{q}_n$ and $\mathbf{u}_n$ are uncorrelated among each other.

The methodology proposed in this Chapter for band selection is very close to the one followed by the traditional Orthogonal Subspace Projection (OSP) [239] method but, the orthogonal subspace spanned by the selected wavelengths is approached in a more hardware-friendly way using the Gram-Schmidt orthogonalization method conducted by

the proposed set of core operations. The goodness of this proposal for band selection is evaluated in next Section in which the detection of spectral targets of interest is addressed.

It is also important to mention that the proposed methodology works best when a subset of the most characteristic pixels of the classes contained in the image are used as input. It is due to the brightness calculation since it implies the addition of the squared pixel values for each spectral band. Therefore, if many more pixels of one particular class are present in the training set, they count for more in the selection of the wavelength with the highest $l2\text{-}norm$.

A.3 Target Detection

In this Section, we analyse the suitability of the core operations proposed in this Thesis work for their expanded used in the field of target detection. In the past decades, many algorithms have been proposed in the literature for the detection of targets of interest. According to the way of modelling the spectral variability problem, two approaches exist: geometric models and statistical models. The former describes the background geometrically while the latter patterns it statistically through the computation of the correlation/covariance of the background [57]. Within the geometric group, we can find the OSP algorithm and the Spectral Angle Mapper (SAM). Within the statistical group, the most employed algorithms are the Constraint Energy Minimization (CEM) algorithm, the Adaptive Coherence/Cosine Estimator (ACE) detector and the Matched Filter (MF). In the recent years, sparsity has been also considered in the field of target detection [240, 241]. Nonetheless, the majority of the state-of-the-art solutions act as global detectors that need the entire HSIs to start the target detection process. Therefore, few solutions are found in the literature that fulfil the limitations imposed by real-time applications based on pushbroom scanners where hyperspectral frames are sensed in a line-by-line fashion. In this context, some recent works have been proposed to address this issue by intraline approaches, such as [242–244].

The methodology analysed in this Section is an hybrid approach that physically and statistically models the background using the linear mixing model and the correlation matrix estimation. It is based on the well-known Gram-Schmidt orthogonalisation method behind the set of core operations proposed in this Thesis work, which is used for three different purposes. Firstly, it is employed to extract the p_u most representative pixels

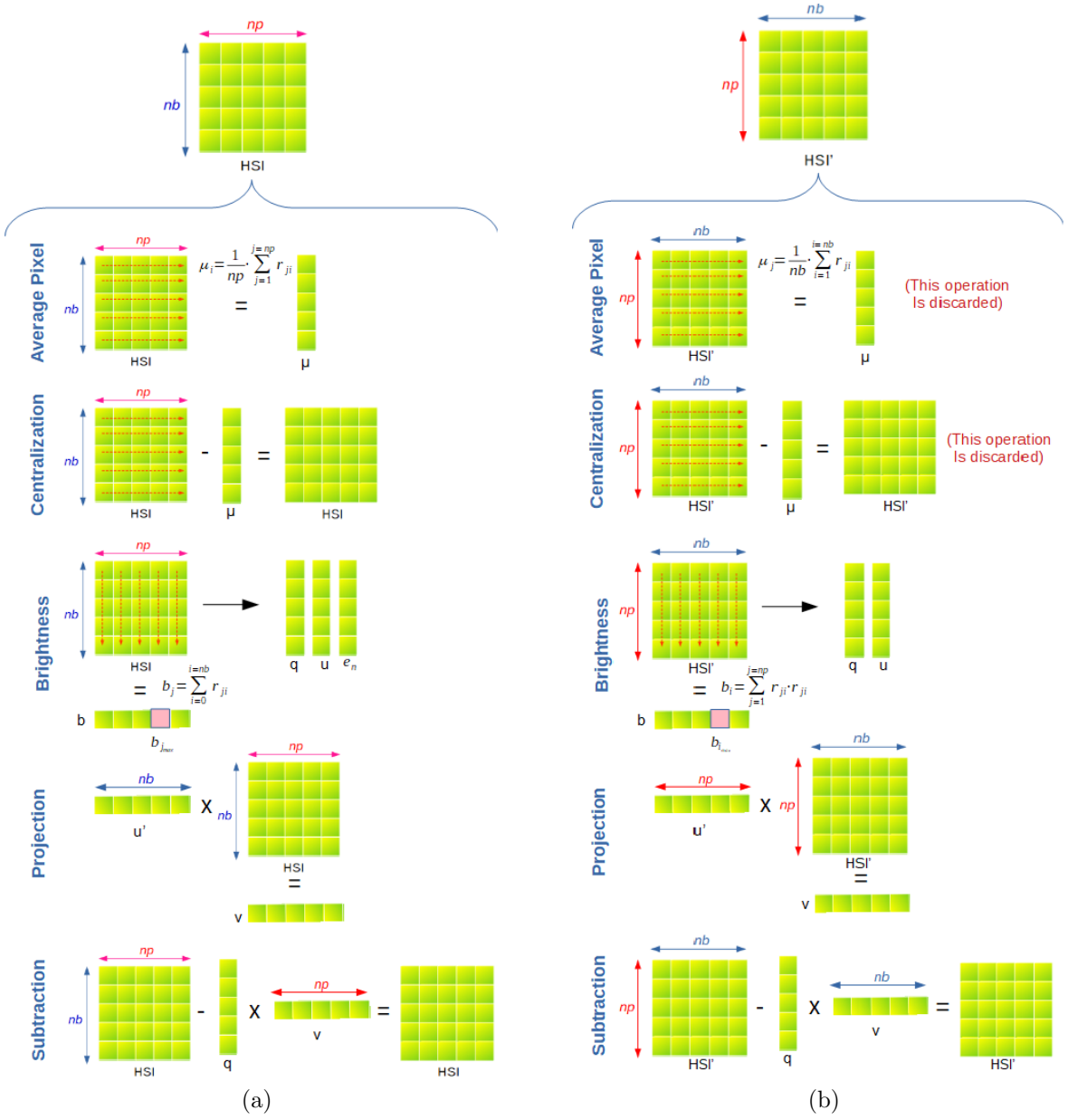


FIGURE A.1: Graphical representation of the variables and their dimensions involved by each proposed core operation when using the original hyperspectral data for the extraction of characteristic pixels and the transpose of the data for band selection.

within the background distribution, which can be seen as undesired signatures, $\mathbf{Ud} = \{\mathbf{ud}_n, n = 1, ...p_u\}$. To this end, the first n_f hyperspectral frames are employed as it is done in the LbL-FAD algorithm introduced in Chapter 3 for anomaly detection. Secondly, and in order to maximize the spectral differences between the desired target, $\mathbf{d}$, and the background signatures, $\mathbf{Ud}$, the set of core operations is used once again in order to select the spectral bands that best differentiate both classes. The target detection process is performed on the following received hyperspectral frames but just using the previously

selected spectral bands. In order to obtain a lower-dimensional feature subspace with uncorrelated predictors, these spectrally reduced frames are transformed using for the third time the Gram-Schmidt orthogonalization method. With it, we achieve a diagonal correlation matrix that significantly reduces the computational burden required to perform the inverse of this matrix. The output of the proposed method for each frame pixel is computed following a similar methodology to the one used by the CEM detector.

In the next Section, the different computing stages behind the proposed methodology for addressing the target detection issue are further described.

A.3.1 Description of the proposed methodology for the detection of targets of interest

This Section collects short descriptions about the four different stages performed by the proposed methodology for addressing the target detection issue.

A.3.1.1 Line-by-Line extraction of the background reference spectra

In order to annihilate all spectral information that does not only belong to the desired target, $\mathbf{d}$, we follow a similar strategy as in the *Stage 1* of the LbL-FAD algorithm for the modelling of the background distribution. To this end, the first sensed n_f hyperspectral frames are independently processed using the proposed set of core operations in order to select the p_u most representative pixels, $\mathbf{E}$, within each targeting image block, $\mathbf{M}_k$.

A.3.1.2 Overall background subspace estimation

One of the main novelties of the methodology followed by the proposed set of core operations is that image blocks, $\mathbf{M}_k$, are independently processed ruling out any spatial alignment restriction. For this reason, many pixels extracted from previous n_f hyperspectral frames actually represent the same materials or entities. In order to eliminate redundancies and select the purest undesired vectors, $\mathbf{Ud}$, the set of core operations is applied once again, though input matrix, $\mathbf{M}_k$, is now replaced by a matrix $\mathbf{B^*} = \{\mathbf{E}_k, k = 1, ..., n_f\}$ whose columns collect the background reference vectors extracted from each first n_f frames. Consequently, a subset of the p_u most representative undesired pixels, $\mathbf{Ud} = \{\mathbf{ud}_n, n = 1, ...p_u\}$, is obtained.

As it can be noticed, this computing stage also matches with the *Stage 2* of the LbL-FAD algorithm. Lastly, it is also worth mentioning that samples obtained in previous flights may be used instead of those obtained from the first n_f frames.

A.3.1.3 Selection of the most representative spectral bands

Once that the subspace of undesired signatures is obtained, **Ud**, the next stage consists in the selection of those spectral bands that best separate both **d** and **Ud**. It also permits to compress the image under analysis onto a lower-dimensional feature subspace where predictors are uncorrelated. The Principal Component Analysis (PCA) has been traditionally employed to perform dimensionality reduction in hyperspectral images. Nonetheless, we also propose the use of the set of core operations to this end, as it was described in Section A.2. On the one hand, they do not perform computationally demanding calculations as the eigenvalues and eigenvectors decomposition, covariance matrix and inverse matrix computations. On the other hand, it also permits the reuse of the hardware resources and the human endeavours intended for the implementation of the targeted set of core operations, which are common to the different algorithm stages.

As it was commented in Section A.2, some minor considerations have to be taken into account in order to use the proposed set of core operations for the selection of a reduced number of the most characteristic spectral bands, *nbr*. In this sense, the transpose of the set of pixels consisting of **d** and **Ud** signatures is used as input. It is done in this way in order to select the spectral bands with the biggest brightness in each iteration of the set of core operations, instead of selecting the most different pixels as it is done in preceding stages described in Sections A.3.1.1 and A.3.1.2. Accordingly, output vectors **E**, **Q** and **U** consist now of *nbr* vectors with $p_u + 1$ elements. Anyway, indexes of the *nbr* most characteristic bands, $\mathbf{Index} = \{ind_b, b = 1, ...nbr\}$, are just needed for the subsequent algorithm stage.

A.3.1.4 Target Detection

After selecting the *nbr* most characteristic spectral bands using the first n_f hyperspectral frames, the target detection process itself is performed on the following received image blocks. In this stage, we work with spectrally reduced image blocks, $\mathbf{M}_k$, whose pixels just retain the spectral information contained in the selected *nbr* spectral bands. Therefore, elements of $\mathbf{M}_k$ are still correlated.

In order to obtain a new subspace with uncorrelated variables, the modified Gram-Schmidt method displayed in Lines 9 to 12 of Algorithm 2 in Chapter 2 is applied to the transpose of $\mathbf{M}_k$ and $\mathbf{d}$, resulting in outputs vectors $\mathbf{Q}$ and $\mathbf{U}$ where the nbr variables within each $\mathbf{q}_j$ and $\mathbf{u}_j$ are orthogonal among them and hence, uncorrelated. In this case, operations involved in Lines 9 to 12 of Algorithm 2 in Chapter 2 are repeated nbr times.

Similar to the CEM detector, the output of the proposed method for each pixel $\mathbf{r}_j$ is shown in Equation A.1, where $\mathbf{R}^{-1}$ is the inverse of the correlation sample matrix of the image block under analysis, $\mathbf{M}_k$. Nevertheless, $\mathbf{R}^{-1}$ is a diagonal matrix in the proposed methodology, since the predictors of the new subspace are orthogonal, and equal to $\mathbf{U} \cdot \mathbf{U}' = (\mathbf{Q} \cdot \mathbf{Q}')^{-1}$, which is much less computationally expensive than computing the inverse of a matrix.

$$y = \frac{\mathbf{q}(d) \cdot \mathbf{R}^{-1} \cdot \mathbf{q}(r_j)'}{\mathbf{q}(d) \cdot \mathbf{R}^{-1} \cdot \mathbf{q}(d)'} \tag{A.1}$$

For the sake of clarity, the overall description of the proposed methodology for the detection of targets of interest is summarized in Algorithm 7. The three-dimensional (3D) input hyperspectral cube, $\mathbf{HI} = [\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_k]$, is a sequence of nr hyperspectral frames or lines of pixels composed of nc pixels with nb spectral bands. Pixels within $\mathbf{HI}$ are grouped in blocks of BS pixels, $\mathbf{M}_k = \{\mathbf{r}_j, j = 1, ..., BS\}$, being normally BS equal to nc, or multiple of it, and k spans from 1 to $\frac{nr \cdot nc}{BS}$. The output is a map $\mathbf{Y} = \left\{\mathbf{y}_{kj}, k = 1, ..., \frac{nr \cdot nc}{BS}, j = 1, ..., BS\right\}$ where each image pixel has a score as a function of its spectral similarity with the target signature to be found. In this regard, $\mathbf{d}$ is the desired target, which is a spectral signature with nb spectral bands. $\mathbf{Ud}$ is a set of p_u undesired pixels or background spectra. nbr is the number of the most characteristic spectral bands and $\mathbf{F}$ is the input block composed of $\mathbf{d}$ and pixels within $\mathbf{M}_k$ whose spectral bands are the selected nbr wavelengths. The transpose of a vector is represented as $'$. Finally, $\mathbf{R}^{-1}$ represents the inverse of the correlation sample matrix, which is actually a diagonal matrix of size [nbr, nbr].

A.3.2 Experimental Results

In this Section, we briefly analyse some of the detection results obtained by the proposed methodology for the detection of targets of interest using some HSIs sensed over real scenarios and one taken over a laboratory-controlled scene.

Algorithm 7 Proposed methodology for target detection.

Inputs:

$\mathbf{HI} = [\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_k], \mathbf{d}, n_f, \alpha$

Outputs:

$\mathbf{Y} = [\mathbf{y}_{11}, \mathbf{y}_{12}, ..., \mathbf{y}_{kj}]$

Algorithm:

Stage 1: Line-by-Line extraction of the background reference spectra

for k = 1 **to** n_f **do**
  $\mathbf{E}_\mathrm{k}$ = Set of core operations ($\mathbf{M}_\mathrm{k}, \alpha$);
  $\mathbf{B^*} = [\mathbf{B^*}, \mathbf{E}_\mathrm{k}]$;
end for

Stage 2: Overall background subspace estimation

Ud = Set of core operations ($\mathbf{B^*}, \alpha$); // Set of Undesired signatures

Stage 3: Selection of the most representative spectral bands

Index = Set of core operations ([**d**, **Ud**]', α); // Indexes of the selected spectral bands

Stage 4: Target Detection

for k = $n_f + 1$ **to** $\frac{nr \cdot nc}{BS}$ **do**
  $\mathbf{F}$ = [**d**(Index), $\mathbf{M}_k$ (Index)]'; // Size of $\mathbf{F}$ is [BS+1, nbr]
  for b = 1 **to** nbr **do**
    $\mathbf{q} = \mathbf{F}$(b);
    $\mathbf{u} = \mathbf{q}/(\mathbf{q}' \cdot \mathbf{q})$;
    $\mathbf{v} = \mathbf{u}' \cdot \mathbf{F}$;
    $\mathbf{F} = \mathbf{F} - \mathbf{q} \cdot \mathbf{v}$;
    $\mathbf{Q}$ = [$\mathbf{Q}$, $\mathbf{q}$']; // Size of $\mathbf{Q}$ is [nbr, BS]
    $\mathbf{U}$ = [$\mathbf{U}$, $\mathbf{u}$']; // Size of $\mathbf{U}$ is [nbr, BS]
  end for
  $\mathbf{R}^{-1} = \mathbf{U} \cdot \mathbf{U}'$; // Size of $\mathbf{R}^{-1}$ is [nbr, nbr]
  for j = 1 **to** BS **do**
    $\mathbf{y}_{kj} = \frac{\mathbf{q}(1) \cdot \mathbf{R}^{-1} \cdot \mathbf{q}(j)'}{\mathbf{q}(1) \cdot \mathbf{R}^{-1} \cdot \mathbf{q}(1)'}$
  end for
end for

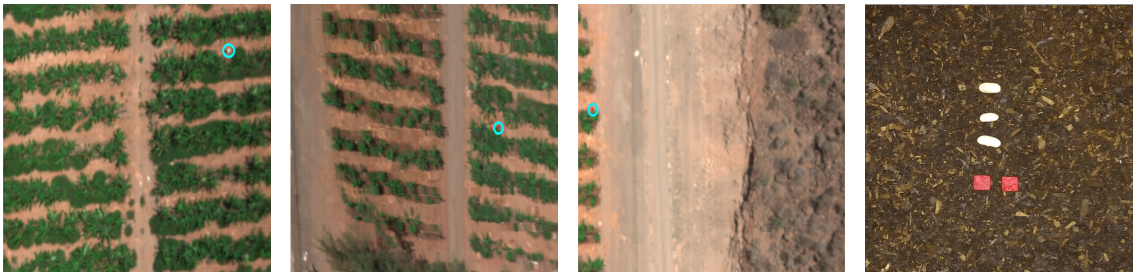
A.3.2.1 Reference Hyperspectral Data

In order to test the goodness of the proprosed method, we have used a subset of the bunch of real hyperspectral data collected by the acquisition platform described in [89] and employed in preceding Chapters 3-5. This data set was collected over multiple farming areas on the island of Gran Canaria (Spain) by a pushbroom sensor mounted on a UAV. In particular, the reference images are selected portions of some swaths collected in one flight campaign. These data cover the spectral information from 400 to 1000 nm using 160 spectral bands and consist of 825 lines height, each line comprising 1024 hyperspectral pixels with 12-bits depth. A RGB representation of these hyperspectral image portions are displayed in Figure A.2 a-c. These images were taken at a height of 72 m over the ground at a speed-rate of 6 m/s with a camera frame-rate of 125 frames per second (FPS), resulting in a ground sampling distance in line and across line of approximately 5 cm. The targets to be detected are the vegetation and the main background materials are the ground and some rock walls. The **d** signature is calculated as the average pixel of some spectra selected from *Drone Image 1* and represented as yellow points in Figure A.3a.

In order to test the goodness of the proposed methodology in other scenarios, we have also used a second data set that was generated in our hyperspectral imaging laboratory using the Headwall hyperspectral sensor Hyperspec®, which operates in the shortwave infrared range (SWIR) between 0.9-2.5 μm. It is a pushbroom camera that provides 384 spatial pixels and 273 spectral bands. However, due to the low-signal-to-noise ratio (SNR) of the first and last spectral bands, they were removed (1-4, 269-273), so that, 264 available bands were retained. The image scene covers an area of 151x126 pixels. Targets are some legumes and two plastic squares. The main background material is peat. This last material is quite challenging since it is composed of multiple different elements and consequently, being less homogeneous. The **d** signatures are calculated as the average pixel of some spectra selected from pixels highlighted in blue and red colors, respectively, in Figure A.3b.

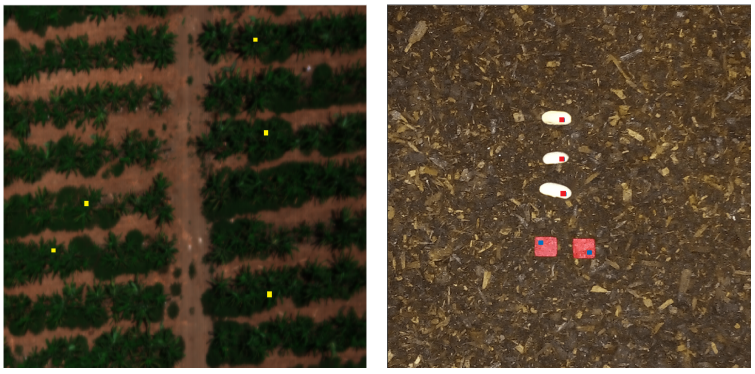
A.3.2.2 Target Detection performance of the proposed methodology

Figure A.4 shows the detection maps given as result by the proposed methodology for the two real data sets. These detection maps are coloured maps where dark red colour



(a) (b) (c) (d)

FIGURE A.2: RGB representation of the HSIs employed in the experiments (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Laboratory-controlled scene.



(a) (b)

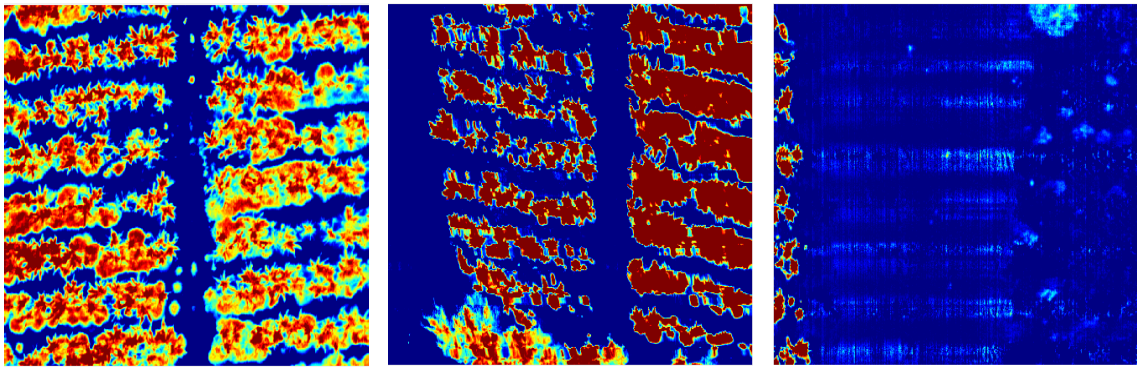
FIGURE A.3: Location of the spectra employed to generate the average target signatures to be detected.

represents pixels very similar to the desired targets to be detected while blue colour represents very dissimilar pixels. Figures A.4a-A.4c display the detection results for the smart farming areas collected by the UAV over scenes shown in Figures A.2a-A.2c. Figures A.4d and A.4e display the detection results for the legumes and plastic squares, respectively, present in the second real data set shown in Figure A.2d. Experimental results have been carried out for image blocks, $\mathbf{M}_k$, composed of $BS = nc$ hyperspectral pixels. The number of image blocks, $\mathbf{M}_k$, used to estimate the undesired signatures, n_f, has been specified in the captions of each representation within Figure A.4.

As it can be seen, our proposal has been able to accurate detect the desired targets in the five case studies. For the smart farming scenes, it has been also demonstrated that although the target signature was estimated from some pixels located in solely *Drone Image 1*, it has been accurately detected in the other HSIs as well. In addition, the separability between the desired target and the background is quite precise, resulting in less noisy detection maps for *Drone Image 1* and *Drone Image 2*. *Drone Image 3*

covers a small arid area on the right side characterized by dry thorn bushes, which have been slightly distinguished from the background. Nonetheless, some sparkles and spectral distortions introduced by the acquisition system movement have been also marked with a comparable score as those aforementioned entities. Anyway, they are far from being misclassified with respect to vegetation placed on the left side of the scene in question. In general, our proposal has been capable of detecting even isolated plants as those placed in the middle of *Drone Image 1*. Regarding the second data set, the proposed methodology has been also able to detect the legumes and plastic squares with high precision. However, these detection results are a bit more noisy than those represented in Figures A.4a-A.4c. It is due to the high variability of materials that conform the background and undesired flashes occurred during the capturing process.

Finally, it is worth mentioning that our proposal is able to detect any target regardless of the proportion of the image covered by them, unlike the traditional concept of target detection where the presence of the desired signatures to be detected is scarce.



(a) $n_f = 100$ (b) $n_f = 100$ (c) $n_f = 100$

(d) $n_f = 10$ (e) $n_f = 10$

FIGURE A.4: Target Detection results. (a) Drone Image 1. (b) Drone Image 2. (c) Drone Image 3. (d) Laboratory-controlled scene - bean spectra. (e) Laboratory-controlled scene - plastic spectra.

A.4 Unmixing

Linear spectral unmixing has been consolidated as an essential tool for the analysis of remotely sensed HSIs. It is based on the idea that each captured pixel in a HSI can be represented as a linear combination of a collection of constituent spectra, also called endmembers, weighted by an abundance factor. The endmembers are generally assumed to represent the pure materials present in the image and the abundances at each pixel establish the proportion of each endmember in the pixel under inspection. This linear mixture model assumes that secondary reflections and scattering effects can be neglected from the data collection procedure, and hence, the measured spectra can be expressed as a linear combination of the spectral signatures of materials present in the mixed pixels. Therefore, it can be concluded that a HSI can be represented as a function of some image pixels, $\mathbf{E}$, and their corresponding abundances, which can be derived from the projection of each image pixel onto each endmember.

By analogy, it can be deduced that the set of core operations proposed in this Thesis work follows an unmixing-based strategy where the most characteristic pixels selected from each image block, $\mathbf{E}$, can be seen as local endmembers, while abundances can be inferred from the projection vectors, $\mathbf{V}$. Indeed, the Gram-Schmidt orthogonalization method was firstly employed in the field of hyperspectral unmixing by the Fast algorithm for linearly UNmixing hyperspectral images (FUN)[105]. The FUN algorithm allows performing the estimation of the number of endmembers and their extraction simultaneously using the aforementioned Gram–Schmidt method. In addition, this algorithm is also able to calculate the abundance factors employing very similar operations based on orthogonal projection techniques as well. Although the FUN algorithm was designed to overcome the limitations imposed by the computationally complex nature of the hyperspectral imagery processing, it still shows some constraints linked to the following points:

1. The FUN algorithm has to be applied globally to the entire HSI in order to address the unmixing process. In this regard, our proposed methodology is able to estimate the endmembers present in an image from the local endmembers extracted from each image block, $\mathbf{M}_k$, which are processed independently. It allows to be adapted to those scenarios in which images are sensed in a line-by-line fashion as those addressed in this Thesis work.

2. In the original FUN algorithm, vectors $\mathbf{U}$ are orthonormalized vectors resulting from dividing each $\mathbf{q}_n$ by its norm, which involves resolving a squared root. Naturally,

it is not a suitable solution from the hardware implementation point of view and hence, it was proposed in [245] to redefine $\mathbf{U}$ vectors as the division of each $\mathbf{q}_n$ by its own bright. On this basis, we have also followed this recommendation in the definition of the proposed set of core operations, which was demonstrated in Section 2.3.1 of Chapter 2 to be equivalent to the approximation shown in the original FUN algorithm.

3. The FUN algorithm tries to select as endmembers the most different pixels, or the most extreme ones, within the captured dataset. Therefore, in order to select the first endmember according to this criterion, the FUN algorithm also uses the average pixel, $\hat{\boldsymbol{\mu}}$, as our proposal does. Nonetheless, the FUN algorithm projects all image pixels onto the average pixel in order to quantize how much information of each pixel is not contained in the centroid pixel and hence, showing how different each pixel is from the average. Finally, the FUN algorithm selects the pixel with the largest orthogonal projection as the first endmember. On the contrary, our proposal employs the average pixel to centralize the input image and selects the first endmember of those with the highest brightness, that is, the one most deviated from the media. It implies to work with the centralized version of the image along the remaining computing operations and also, to reduce the number of operations to be executed.

4. The main difference between the FUN algorithm and the methodology introduced in this Section lies in the calculation of the abundance factors. To this end, the FUN algorithm uses orthogonal projections as well for the definition of these parameters. In this sense, the FUN algorithm transforms the p selected endmembers in a set of p vectors, $\mathbf{U}^*$, that contain the spectral information in the direction spanned by each endmember, $\mathbf{e}_n$, and not shared with the others. Nonetheless, this results in a set of $\mathbf{U}^*$ vectors which are really not orthogonal among each other. Figure A.5 shows an example where two vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, and their corresponding $\mathbf{u}_1^*$ and $\mathbf{u}_2^*$, are displayed. As can be seen, $\mathbf{u}_1^*$ is orthogonal to $\mathbf{e}_2$ and, also $\mathbf{u}_2^*$ with respect to $\mathbf{e}_1$, but $\mathbf{u}_1^*$ and $\mathbf{u}_2^*$ are not orthogonal among each other. In this sense, our proposal does estimate a set of orthogonal vectors for computing the orthogonal projection matrix, $\mathbf{P}$, in a similar way as described by the LbL-FAD algorithm.

5. Finally, the FUN algorithm was designed to be used with floating-point notation. Nonetheless, the methodology for the unmixing of hyperspectral imagery proposed in this Section is actually based on the set of core operations proposed in this Thesis

work. Therefore, this methodology could be potentially implemented using fixed-point notation by means of integer arithmetic as well.

In the next Section, the proposed methodology for addressing the linear unmixing of HSIs is briefly described.

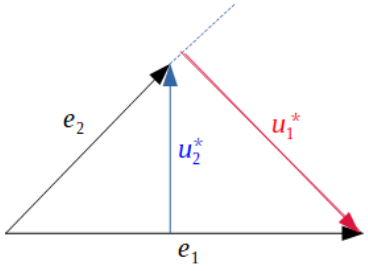


FIGURE A.5: Example of two vectors, $\mathbf{u}_1^*$ and $\mathbf{u}_2^*$, estimated by the FUN algorithm for the calculation of the abundances.

A.4.1 Description of the proposed methodology for linearly unmixing HSIs

The whole process for linearly unmixing a given HSI may be seen as a three-stage process: 1) estimation of the number of endmembers that are present in the hyperspectral image under consideration, 2) extraction of these endmembers from the hyperspectral data set and, 3) calculation of the abundances associated with the endmembers induced in the previous step per each mixed pixel of the image. This Section collects short descriptions about these three different stages performed by the proposed methodology for addressing the linear unmixing of HSIs.

A.4.1.1 Estimation of the number of endmembers and their extraction

The extraction of the p most characteristic pixels or endmembers present in a HSI has been treated in the same way as in the other algorithmic solutions proposed in this Thesis work, such as the LbL-FAD and the HyperLCA. To this end, the set of core operations introduced in Chapter 2 has been used. In this sense, the first endmember to be extracted is the one with the highest deviation from the average pixel. Then, subsequent endmembers are those identified with the largest orthogonal projections to the endmembers already

extracted. Nonetheless, the strength of the proposed methodology lies in the independent processing of blocks of image pixels, $\mathbf{M}_k$. For this reason, the endmember extraction is addressed in a line-by-line fashion in order to extract the local endmembers inherent to each $\mathbf{M}_k$. Subsequently, the global endmembers are extracted from the set of selected local signatures using the above mentioned methodology.

Regarding the estimation of the number of endmembers, the proposed methodology uses the same stopping condition as the LbL-FAD algorithm for the selection of the bakcground spectra. As it was explained in Chapter 2, each time that a pixel $\mathbf{e}_n$ is selected, the spectral information that could not be represented by the already extracted pixels remains in image matrix $\mathbf{C}$. It means that if the image is represented using the selected $\mathbf{e}_n$ pixels, according to the LMM, a small part of the spectral information is lost when the image is reconstructed using the p selected $\mathbf{e}_n$ pixels and besides, equal to the remaining information contained in $\mathbf{C}$. In this sense, the maximum brightness, $b_{j_{\max}}$, after the $\mathbf{e}_p$ vectors have been selected may be representative of the spectral losses introduced by the unmixing process and consequently, it could be used to set p. In this context, the endmember extraction process finishes when the loss, in percentage terms, is less than an input parameter that represents the percentage of the spectral information that will be considered as noise, α. This stop condition can be seen in Equation A.2 where $(\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})$ represents the initial value of $\mathbf{r}_{j_{\max}}$ in $\mathbf{C}$.

This way, the proposed methodology manages to extract the endmembers and to estimate the number of endmembers in a single process.

$$\frac{b_{j_{\max}}}{(\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})' \cdot (\mathbf{r}_{j_{\max}} - \hat{\boldsymbol{\mu}})} \cdot 100 < \alpha \rightarrow \text{Stop selecting } p \text{ } e_n \text{ pixels} \tag{A.2}$$

A.4.1.2 Abundance Estimation

Linear unmixing is based on the idea that each captured pixel in a hyperspectral image can be represented as a linear combination of a set of p spectrally pure constituent spectra or endmembers weighted by an abundance factor that establishes the proportion of each endmember in the pixel under inspection. On this basis, the proposed methodology considers that image pixels can be perfectly reconstructed by the space spanned by these characteristic pixels in order to estimate the abundance vectors. To this end, the rationale behind the orthogonal projection matrix, $\mathbf{P}$, is exploited as it was done by the LbL-FAD algorithm for the detection of anomalous spectra.

Let's set an example where three endmembers, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, and its corresponding abundance vectors, a_1, a_2 and a_3, are considered. Therefore, an image pixel, $\mathbf{r}_j$, can be represented following the LMM shown in Equation A.3. For the estimation of the abundance vector, three different vector space basis are estimated, one per concerned endmember. Since each abundance factor represents the proportion of each endmember, $\mathbf{e}_n$, in the pixel under inspection, $\mathbf{r}_j$, each a_n can be estimated from the pixel projection over the orthogonal subspace to the one spanned by the other endmembers. For instance, the subset of endmembers $\mathbf{e}_2$ and $\mathbf{e}_3$ can be orthogonalized using the Gram-Schmidt method obtaining the $\mathbf{Q}$ and $\mathbf{U}$ vectors. As it was further explained in Chapter 3, the computation of the orthogonal projection matrix, $\mathbf{P}$, to the space spanned by $\mathbf{e}_2$ and $\mathbf{e}_3$ is equivalent to $\mathbf{P}^{\perp}_{\mathbf{e}_2,\mathbf{e}_3} = \mathbf{I} - \mathbf{Q} \cdot \mathbf{U}'$. Hence, the pixel projection over the space spanned by $\mathbf{e}_1$ could be computed as $a_1 = \mathbf{r}'_{\mathrm{j}} \cdot \mathbf{P}^{\perp}_{\mathbf{e}_2,\mathbf{e}_3} \cdot \mathbf{r}_{\mathrm{j}}$. The other abundances, a_2 and a_3, could be estimated as an analogous way. In this way, we manage to have a new subspace on which to project the data whose basis vectors are really orthogonal with each other. Naturally, the explicit computation of the orthogonal projection matrix, $\mathbf{P}$, could be replaced by the Gram-Schmidt method as well, as it was explained in Chapter 3 for the HW-LbL-FAD.

$$\mathbf{r}_j = a_1 \cdot \mathbf{e}_1 + a_2 \cdot \mathbf{e}_2 + a_3 \cdot \mathbf{e}_3 \tag{A.3}$$

Nonetheless, the resulting abundances do not meet one of the two physical constraints imposed to the LMM, in particular, the abundance sum-to-one constraint, which establishes that $\sum_{n=1}^{n=p} a_n = 1$. Nonetheless, experience has shown that the obtained abundances are indeed very close to meet the constraint. Taking this into account, it is possible to apply the aforementioned constraint to the estimated abundances by dividing each element with the abundance vector by the sum of their terms.

A.4.2 Experimental Results

In this Section, we briefly analyse the performance of the proposed methodology for the unmixing of HSIs. For this purpose, the common-used AVIRIS Cuprite image has been used in this work in order to test the proposed algorithm in a more realistic scenario. In addition, it also permits to compare the performance in terms of endmember extraction with the FUN algorithm and the state-of-the-art Vertex Component Analysis (VCA). Regarding the accuracy of the abundance calculation process, we have evaluated it using a set of 3 real HSIs commonly used in the field of unmixing of hyperspectral imagery,

namely Samson, Jasper Ridge and Urban data sets. On the basis of a set of reference endmembers, we evaluate the goodness of the abundance maps obtained by our proposal with an available ground-truth using the *root mean squared error (RMSE)* as an assessment metric. All these data sets and their corresponding ground truths were downloaded from [246], and detailed a explanation about them is provided in [247].

A.4.2.1 Reference Hyperspectral Data

As it was above mentioned, four real hyperspectral datasets are used for conducting the experiments to assess the performance of the proposed methodology for hyperspectral unmixing. A short description about these scenes are given below:

Samson data: Samson dataset is originally of size 952x952 pixels. Nonetheless, a short region of 95x95 pixels is used in the experiments. Each pixel is recorded at 156 channels covering the wavelengths from 401 nm to 889 nm. There are three target endmembers in the dataset, including "Rock", "Tree", and "Water". Figure A.6a shows the Samson data and the corresponding ground truth of the abundance distributions.

Jasper Ridge data: Jasper Ridge is one of the most widely used hyperspectral unmixing datasets. It is originally of size 512x614 pixels, recording spectral information at 224 channels covering the wavelengths from 380 nm to 2500 nm. Nonetheless, a short region of 100x100 pixels is used in the experiments. In addition, the channels 1–3, 108–112, 154–166 and 220–224 have been removed due to dense water vapor and atmospheric effects, lastly retaining 198 channels. There are four endmembers latent in this data: "Road", "Soil", "Water" and "Tree". Figure A.6b shows the Jasper Ridge data and the corresponding ground truth abundance maps.

Urban data: Urban dataset is a very popular hyperspectral dataset for unmixing studies. The images are composed of 307×307 pixels, and the spatial resolution is 2 m. Spectral channels 1–4, 76, 87, 101–111, 136–153 and 198–210 have been removed due to dense water vapor and atmospheric effects, lastly retaining only 162 channels. There are six endmembers in the dataset, including "Asphalt Road", "Grass", "Tree", "Roof", "Metal", and "Dirt". Figure A.6b shows the Urban data and the corresponding reference abundance maps.

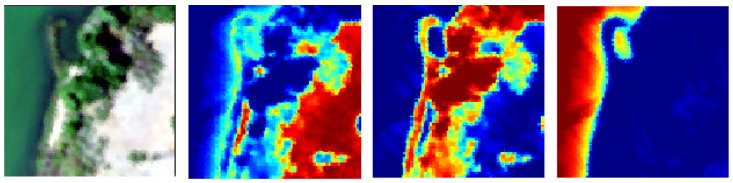
Cuprite data: Cuprite dataset is well understood from the mineralogical point of view and consists of 224 spectral bands between 0.4 and 2.5 μm. Prior to the analysis, several

bands have been removed due to water absorption and low SNR, resulting in a total of 188 spectral bands. Particularly, we have used in our experiments a portion of the full Cuprite image, with a spatial size of 250 × 190 pixels, where there are 14 types of minerals. Nonetheless, in order to compare the extracted endmembers with those obtained by the FUN and the VCA algorithms, results shown in [105] are used in the analysis. In it, five available spectral signatures known in the AVIRIS Cuprite scene are used as ground-truth, which are Alunite, Buddingtonite, Calcite, Kaolinite, and Muscovite. The accuracy of the pixel selected as endmember is evaluated by calculating the spectral angle (°) between the reference spectra and the selected ones. Figure A.6d shows the AVIRIS Cuprite data and the reference endmember spectra.

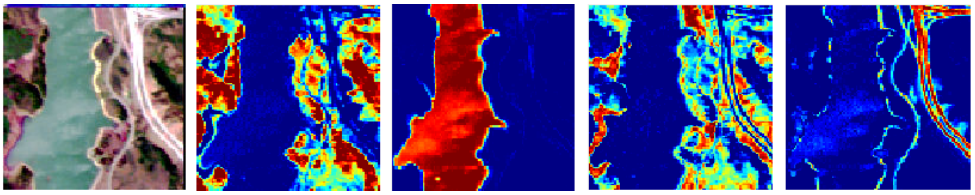
A.4.2.2 Performance of the proposed method for hyperspectral unmixing and abundance calculation

Firstly, we evaluate the accuracy of the proposed method for the identification and extraction of the endmembers. For this purpose, the reference spectral signatures of alunite, buddingtonite, calcite, kaolinite, and muscovite, which are present in the Cuprite images, are compared with those obtained with the proposed method through the analysis of the spectral angle. In addition, the obtained results are compared with the FUN algorithm and the VCA method analysed in [105]. Table A.1 shows the spectral angle scores obtained by the proposed method, the FUN algorithm and the VCA algorithm when comparing the respective extracted endmembers with respect to the reference pure spectral signatures. The experimental results demonstrate that the accuracy obtained by the proposed algorithm when extracting the endmembers is very similar to the FUN algorithm and for four of the spectral signatures outperforms the VCA algorithm. In this sense, there is a slight deviation from these two state-of-the-art methods for the muscovite mineral. Nonetheless, the spectral angle values are quite better for the calcite and kalonite examples compared with the FUN and the VCA methods. Lastly, the proposed method extracts in general spectral signatures more similar to the reference spectra, as can be seen from the last column of Table A.1 that collects the average value of the reached spectral angles.

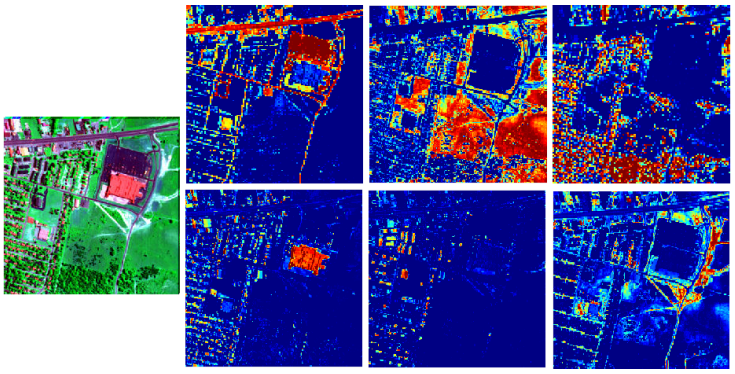
Regarding the abundance estimation, Table A.2 collects the *RMSE* values obtained by the comparison between the ground-truth and the abundance maps estimated by the proposed methodology using the Samson, the Jasper Ridge and the Urban data sets. To do this, reference endmember signatures are employed for a fairer comparison. In addition, Figure A.7 displays some classification maps obtained from the reference abundance maps and



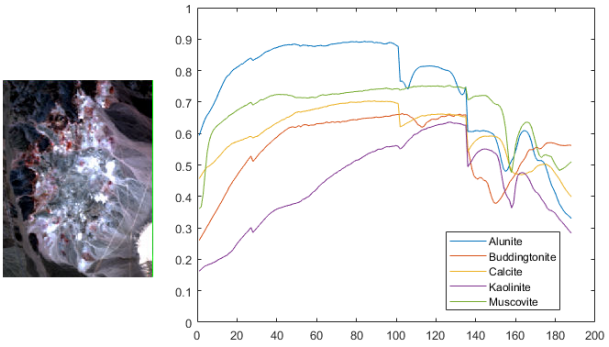
(a)



(b)



(c)



(d)

FIGURE A.6: RGB representation of the HSIs employed in the experiments. (a) Samson data and the ground truth abundance maps. (b) Jasper Ridge data and the ground truth abundance maps. (c) Urban data and the ground truth abundance maps. (d) AVIRIS Cuprite data and the reference endmember spectra.

the ones obtained by our proposal. In this sense, colours represent the different classes established as a function of the similarity of each pixel with each endmember. To this end, each pixel has been classified within one of each endmember class according to its largest abundance factor, a_n. As it can be concluded, very small errors are obtained

Algorithms	Spectral Angle (°)					
	Alunite	*Buddingtonite*	*Calcite*	*Kaolinite*	*Muscovite*	Average
Proposed Method	4.92	4.15	4.72	5.06	7.97	5.36
FUN	4.83	4.26	9.62	10.98	4.64	6.87
VCA	11.95	5.85	6.45	14.90	5.57	8.94

TABLE A.1: Comparison of the spectral signatures obtained by the proposed method, the FUN algorithm and the VCA algorithm.

when estimating the abundances maps with the proposed method. Therefore, it can be concluded that this could be an accurate approximation for this purpose.

Algorithm	*RMSE*		
	Samson data	*Jasper Ridge data*	*Urban data*
Proposed Method	4.73E-04	6.32E-04	2.67E-04

TABLE A.2: *RMSE* values obtained after the comparison between the reference abundance maps and the ones obtained by our proposal using the reference signatures of the endmembers for the Samson, Jasper Ridge and Urban data sets.

A.5 Classification

Since the early days of the study and analysis of the remotely-sensed hyperspectral data, the LMM and the concept of orthogonal subspace projections (OSPs) have been extensively explored in hyperspectral image classification. The first time that orthogonal projections were used to address this analysis was in [110]. In this previous work, the LMM was rewritten as an addition of three components, that is, the desired signatures, $\mathbf{d}$, a set of undesired targets, $\mathbf{Ud}$, and the random noise. The reason for separating $\mathbf{Ud}$ is to design an OSP to annihilate $\mathbf{Ud}$ from an observed pixel prior to classification. For doing so, the orthogonal subspace projection matrix to the space spanned by $\mathbf{Ud}$, $\mathbf{P}_{Ud}{}^{\perp}$, can be applied to the LMM in order to eliminate the $\mathbf{Ud}$ from the equation. On this basis, a mixed pixel classification can be carried out by a two-stage process, an undesired signature rejecter followed by a matched filter. Nonetheless, this first approximation assumes the complete knowledge of the abundances of the spectral signatures in advance,

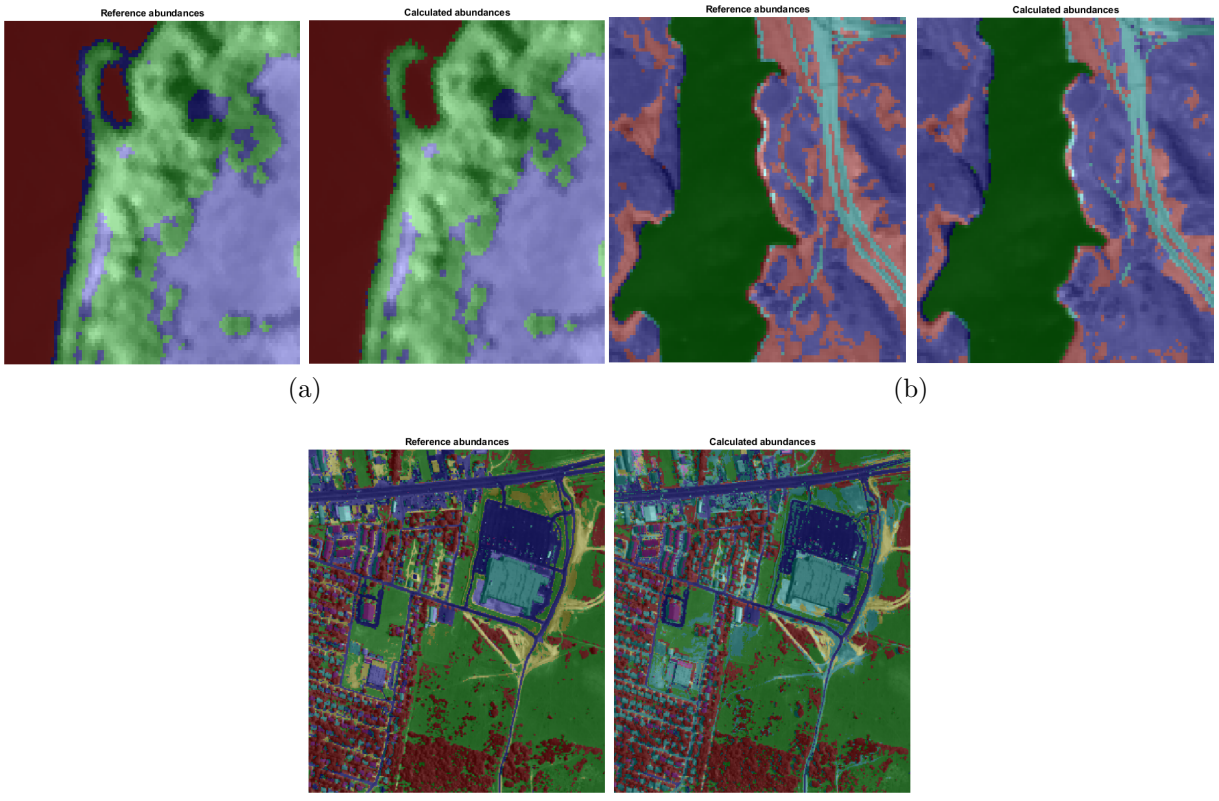


FIGURE A.7: Classification maps obtained from the ground-truth of the abundance estimation and the ones obtained by our proposal for the Samson data, the Jasper Ridge data and the Urban data.

which unfortunately is not possible in real-world applications. For this reason, *a posteriori* OSP solutions emerged to fill this gap, which also contemplated the estimation of the abundances [109, 242, 248].

Against this backdrop, the unmixing model described in Section A.4 could be extended to tackle the problematic behind the classification issue. It is based on OSP by means of the computation of the orthogonal projection matrix for the calculation of the abundance vectors. On the basis of the LMM described in [110], the desired signatures, $\mathbf{d}$, could be seen as each cluster of spectral signatures of one material type, whereas $\mathbf{Ud}$ represents the other classes concerned for the classification of a set of hyperspectral images. Finally, the classification of a spectrum in one of the concerned classes could be made according to the largest abundance factor.

A.5.1 Description of the proposed methodology for the classification of the HSIs

This Section collects short descriptions about the different algorithm stages performed by the proposed methodology for addressing the classification of HSIs. The whole process may be seen as a two-stage process: 1) estimation of the orthogonal subspace to each class in order to calculate the abundance maps and, 2) classification from the abundance maps.

A.5.1.1 Abundance Estimation

There are some studies using abundance maps as inputs for the classification of image pixels within a set of user-defined classes [237, 249–251]. For this reason, we have focused on the methodology proposed in Section A.4.1.2 for the estimation of the abundance factors. In this sense, as many vector space basis as endmembers present in the HSI are computed. Nonetheless, we have spectral classes instead of endmembers in the targeted classification issue. For this reason, they would be estimated as many orthogonal projection matrix, $\mathbf{P}$, as classes are concerned. To do this, the set of labeled training samples within the other classes not included in the targeted one are used as input of the proposed set of core operations in order to obtain the orthogonal vectors $\mathbf{Q}$ and $\mathbf{U}$ and thus, the orthogonal projection matrix, $\mathbf{P}$, that projects the image in the direction spanned by solely the class in question. Another approximation would be to use the average pixel of each class instead of the bunch of spectral signatures selected as training samples [237].

For the sake of clarity, let's set an example with 3 classes composed of p labeled training samples, $\mathbf{c}_1 = [\mathbf{r}_{\mathbf{1}_n}, n = 1, ...p]$, $\mathbf{c}_2 = [\mathbf{r}_{\mathbf{2}_n}, n = 1, ...p]$ and $\mathbf{c}_3 = [\mathbf{r}_{\mathbf{3}_n}, n = 1, ...p]$. Firstly, the orthogonal projection matrix, $\mathbf{P_1}$, that projects the data on the direction spanned by $\mathbf{c}_1$ is computed as $\mathbf{P_1} = \mathbf{I} - \mathbf{Q} \cdot \mathbf{U}'$, in which $\mathbf{Q}$ and $\mathbf{U}$ are orthogonal basis of the space composed by the other classes, $\mathbf{c}_2$ and $\mathbf{c}_3$. Secondly, the pixel projection over the space spanned by $\mathbf{c}_1$ could be computed as $a_1 = \mathbf{r}'_{\mathrm{j}} \cdot \mathbf{P_1} \cdot \mathbf{r}_{\mathrm{j}}$. The other abundances, a_2 and a_3, could be estimated as an analogous way.

Nonetheless, the resulting abundances do not meet one of the two physical constraints imposed to the LMM, in particular, the abundance sum-to-one constraint, which establishes that $\sum_{n=1}^{n=p} a_n = 1$. Nonetheless, experience has shown that the obtained abundances are indeed very close to meet the constraint. Taking this into account, it is possible to apply

the aforementioned constraint to the estimated abundances by dividing each element with the abundance vector by the sum of their terms.

A.5.1.2 Classification using abundance maps

Finally, the maximum abundance classification (MAC) is proposed as the final stage within the classification process. In this sense, each image pixel, $\mathbf{r}_j$, is classified within one of the concerned spectral classes according to its largest abundance factor.

A.5.2 Experimental Results

In this Section, we briefly analyse the performance of the proposed methodology for the classification of HSIs. The evaluation of the detection performance is visually made at object-level through the description of the resulting classification maps where image pixels are sorted in one of the user-defined material classes by its largest abundance factor. Labelled training samples are a reduced number of image pixels directly selected from the images.

A.5.2.1 Reference Hyperspectral Data

The data set used to evaluate the proposed methodology was sensed by the aerial platform described in [89] and employed in preceding Chapters 3-5. Nonetheless, other portions of the hyperspectral scenes sensed in the different flight campaigns are used in the experiments to consider a major variety of spectral classes for the classification. In particular, the reference images are selected portions of some swaths within two different flight campaigns. These data cover the spectral information from 400 to 1000 nm using 160 spectral bands. A RGB representation of these hyperspectral image portions are displayed in Figure A.8. These images were taken at a height of 45 m over the ground and at a speed of 4.5 m/s with the hyperspectral camera capturing frames at 150 FPS. This results in a ground sampling distance in line and across line of approximately 3 cm.

Regarding the spectral classes in which we classify image pixels, they basically consists of vegetation and bare soil. Nevertheless, other materials are also included in the experiments. For instance, *Image 1*, shown in Figure A.8a, mainly contain signatures of three classes: vegetation, a white roof of a wine facility and some very dark areas where the

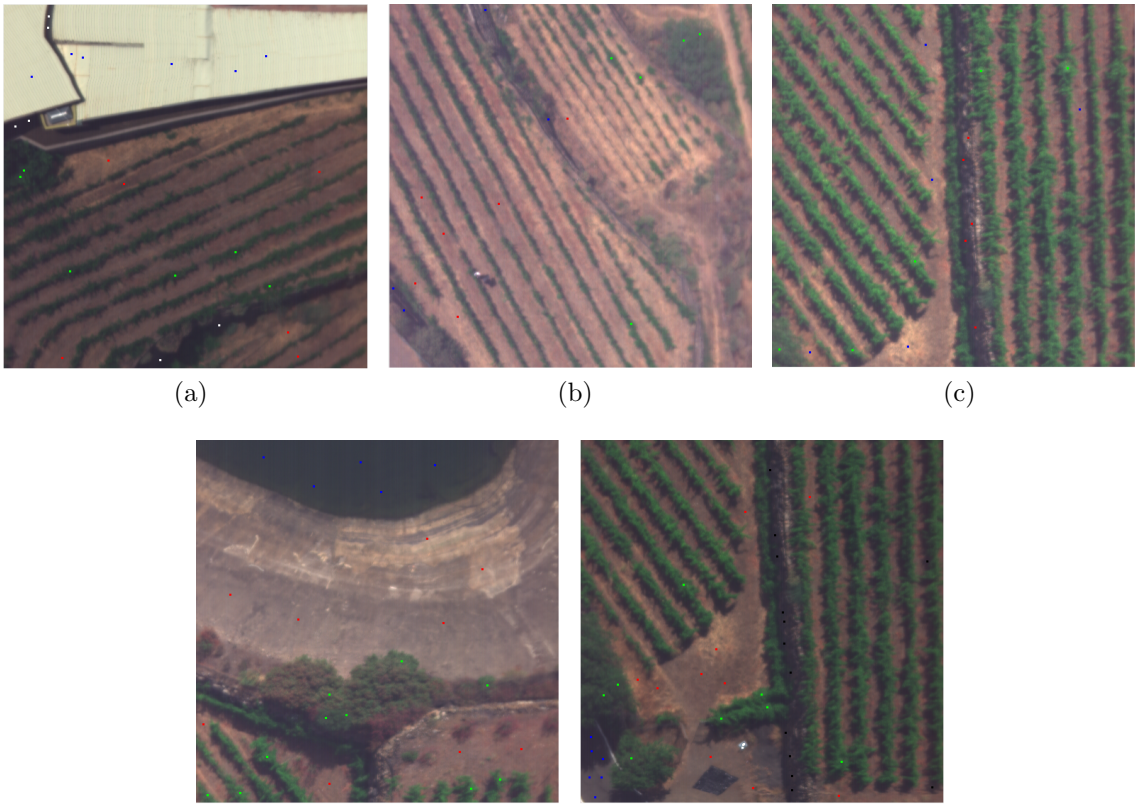


FIGURE A.8: RGB representation of the HSIs employed in the experiments. (a) Image 1. (b) Image 2. (c) Image 3. (d) Image 4. (e) Image 5. Pixels highlighted in colour are used as labelled training samples.

sun light does not reach or composed of black volcanic rocks, such as the external walls of the building and curtain walls among vineyard crop fields. Labelled training samples are composed of 6 pixels, respectively, for each considered class. *Image 2*, shown in Figure A.8b, is composed four spectral classes, that is vegetation, bare soil, volcanic rocks within the curtain walls among vineyard crops, as well as, the white shirt of a person in the middle of the corps. Labelled training samples are composed of 5, 6, 4 and 1 hypespectral pixels, respectively, for each considered class. *Image 3*, shown in Figure A.8c is the most simple scenario since only vegetation, bare soil and volcanic rocks are classified. Labelled training samples are composed of 5 pixels, respectively, for each considered class. In the case of *Image 4*, shown in Figure A.8d, apart from vegetation and bare soil classes, it is also defined a new spectral class composed of spectral signatures of some water pond. Labelled training samples are composed of 10, 9 and 5 hypespectral pixels, respectively. The last image, *Image 5*, shown in Figure A.8e, contains 5 different classes of materials,

that is, vegetation, bare soil, volcanic rocks, a road and a white panel used to radiometrically correct the acquired images. Labelled training samples are composed of 8, 8, 10, 6, 2 hyperspectral pixels, respectively, for each considered class.

A.5.2.2 Performance of the proposed method for the classification of HSIs

Figure A.9 shows the classification maps given as result by the proposed methodology for the five real data sets used in the experiments. For the sake of clarity, they are superimposed on a panchromatic representation of the scenes to be analysed in order to make easier the result interpretations. In these displays, colours represent the different user-defined spectral classes in which we classify image pixels. In the lines that follow, the performance of the proposed method for the classification of the above mentioned spectral classes are commented by means of the description of the resulting classification maps.

1. Regarding the vegetation class, it is easily distinguishable from the others as a spectral point of view. For this reason, image pixels within this class have been accurate classified for all data sets. In addition, many dry thorn bushes present in *Image 4* characterized by red leaves are also well classified. As a singularity, some aquatic plant species located in the border of the pond are also detected.

2. In relation to the bare soil, image pixels have been also well classified for almost all the test cases. Nonetheless, very small and dry bushes located on the right side of the curtain wall located in the middle of *Image 2* have been misclassified. It is because they really represent a barren area where pixels are mixed by the ground.

3. Concerning the black volcanic rocks that curtain walls are built with, it is the most challenging class and hence, a greater proportion of image pixels are misclassified. For *Image 2, Image 3* and *Image 5*, shadows cast by plants are also classified within this class. It makes sense since very poor sun light reaches these areas. For the same reason, the plastic bag placed at the bottom of *Image 5* has been also sorted in this class. In relation to *Image 1*, some pixels within the shadows projected by the white roof were included in the labelled training samples and hence, shadows have been classified in this class as well as the external wall that surrounds the building. Since this class has been not considered in the analysis of *Image 4*, pixels that are actually part of this class have been classified as bare soil.

4. Regarding the white roof of the building shown in *Image 1*, its colour is easily distinguishable from the other classes and hence, it has been accurately classified. In addition, some pixels located on the top of the boundary wall are also included in this class since they are painted in white as well.

5. Water pond displayed in *Image 4* has been well segmented as well in this image in question, though some dark pixels located near the curtain wall that separates the crop fields with the pond have been misclassified. It is because the sun light hardly reaches this area and shadows are projected. It is not a trivial issue since many researches have concluded that the separation of water pixels from other dark surfaces and shadows is highly confused in image classification [252, 253].

6. Finally, in relation to the strange entities shown in *Image 2* and *Image 5*, they have been well segmented from the other image classes though they represent very few image pixels. In the case of the white reference panel displayed in Figure *Image 5*, some pixels over the edge lines of the road are also classified within this class due to their white color.

A.6 Conclusions

In this Appendix, we establish a first contact towards the expanded use of the orthogonal subspace projections and, in particular, the Gram-Schmidt orthogonalization method by means of the set of core operations proposed in this Thesis work, in the fields of band selection, target detection, unmixing and classification. The analysis made is far from being as exhaustive as those carried out by the anomaly detection issue and the lossy compression of HSIs in preceding Chapters 3 and 4, but represents a turning point in the way of future research works.

The most remarkable aspect is the potential of the proposed methodology for addressing other types of hyperspectral analysis. This is certainly in the line of the research goals to be accomplished in this Thesis. In this sense, it is demonstrated the feasibility of targeting several hyperspectral analysis techniques using the same mathematical method. This fact opens up new avenues towards the concurrent execution of multiple hyperspectral imaging applications. Additionally, it also results in many benefits in view of hardware acceleration for real-time or near real-time performance in terms of a reduction in the

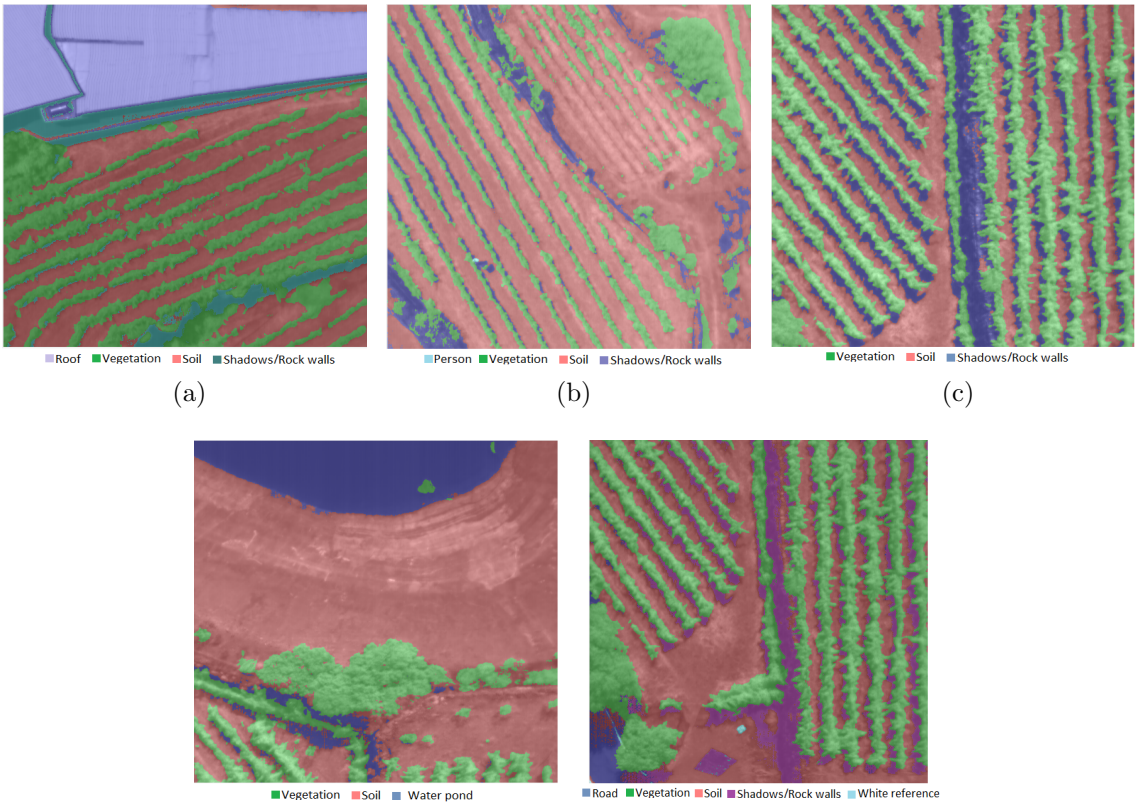


FIGURE A.9: Classification maps obtained by the proposed method. (a) Image 1. (b) Image 2. (c) Image 3. (d) Image 4. (e) Image 5. Colours represent each user-defined class label.

execution times, hardware resources and above all, in human endeavours. Concerning this latter, it implies the studio and analysis of only a single mathematical approach, which consequently permits to focus the efforts from a methodological and productivity points of view, which consequently results in a reduction in the time to market.

Nonetheless, there is always a trade-off among the performance, the hardware utilization and the design requirements. In this regard, the search of optimal quality in the results would involve a custom-made algorithmic solution for the targeted application. Normally, this stringent scenarios require complex and computational demanding mathematical models characterized by intensive memory requirements, low grade of parallelism and a non-scalable nature. All of this may clearly compromise the real-time or near real-time performance for time-sensitive applications and make these approaches not suitable for power-constrained environments. To sump up, the use of the methodological proposal introduced in this Thesis work will depend on the targeted scenario and the objectives set by the application and by the user.

Appendix B

Sinopsis en español

En este Apéndice se recoge una visión general del trabajo de investigación realizado en la presente Tesis Doctoral cuyas principales contribuciones son en el campo del desarrollo de algoritmos de baja complejidad computacional y alta naturaleza paralelizable para el procesamiento de imágenes hiperespectrales a bordo de sistemas aéreos de acquisición de datos. En concreto, este trabajo brinda una solución a la problemática existente en relación a la ejecución simultánea de múltiples técnicas de procesamiento de imágenes en un mismo dispositivo *hardware*, a través de la definición de un conjunto de operaciones comunes cuyas variables de salida puedan ser reutilizadas por distintos métodos de análisis. En particular, la metodología presentada se ha centrado en los campos de la detección de agentes anómalos y la compresión con pérdidas de imágenes hiperespectrales, aunque también se han abordado las primeras aproximaciones para su extensión a otros sectores como la clasificación, la selección de bandas y la deteción de objetivos.

B.1 Introducción

En los últimos años, la tecnología hiperespectral se ha convertido en una herramienta de gran interés para la comunidad científica en el campo de la teledetección y la observación de la superficie terrestre. Hoy en día la podemos encontrar en muchas aplicaciones como la agricultura de precisión, la mineralogía, en acciones de vigilancia y seguridad nacional, en actividades de protección ambiental y desastres naturales, entre muchas otras. Además, su campo de actuación también se ha visto ampliado hacia otros sectores, por lo que actualmente podemos encontrar sistemas de adquisición de imágenes hiperespectrales en biomedicina y aplicaciones comerciales e industriales, como son el control alimentario, separación en plantas de reciclaje, etc.

La creciente popularidad de los sistemas hiperespectrales se fundamenta en la capacidad que presentan para recoger información sobre la escena observada en muchas y continuas longitudes de onda a lo largo del espectro electromagnético. Mientras que el ojo humano puede percibir longitudes de onda dentro del espectro visible (el rojo, el verde y el azul), los sensores hiperespectrales son elementos pasivos capaces de captar la luz reflejada por los objetos y obtener su distribución en cientos de longitudes de onda que incluso abarcan el espectro infrarrojo. Partiendo de la base que muchos elementos puros de la naturaleza presentan una 'huella' única en el espectro electromagnético, la aplicación más importante de la tecnología hiperespectral radica en la detección e identificación de materiales que puedan parecer a simple vista el mismo ente pero que a través del análisis de su firma espectral se ratifique su naturaleza distintiva. De todo ello es claramente deducible el por qué la teledetección es el campo en el que tradicionalmente la tecnología hiperespectral ha tenido mayor impacto. En este sector, los sensores hiperespectrales son colocados a bordo de satélites, aeronaves tripuladas o muy en boga hoy en día, los drones, para la caracterización y el análisis detallado del suelo terrestre, el estudio de la atmósfera, la estimación de parámetros físicos de gran complejidad, la identificación de materiales visualmente similares, etc.

Sin embargo, a pesar de la gran variedad de plataformas de adquisición de datos existentes, el procesamiento abordo de las imágenes hiperespectrales aún presenta muchos retos a abordar, debidos sobretodo al tratamiento y manejo de grandes volúmenes de datos. Por una parte, esto pone en riesgo la actuación en tiempo real de muchas aplicaciones que requieren de resultados inmediatos. Por otra parte, se requiere de dispositivos de cómputo de altas prestaciones. Todo ello se complica aún más con el desarrollo y puesta en el

mercado de los nuevos sensores hiperespectrales que presentan mayores tasas de captura de datos por segundo, además de mayor resolución espacial y espectral. Por todo ello, la corriente dominante ha sido el procesamiento en la superficie terrestre de los datos capturados remotamente por los sistemas anteriormente mencionados tras su transmisión y descarga. Desafortunadamente, la transmisión de datos presenta un cuello de botella debido a los importantes retrasos derivados del limitado ancho de banda de los sistemas de comunicación. Por lo tanto, este sistema de operación claramente compromete la ejecución eficiente, segura y en tiempo real de aplicaciones que requieren de respuestas inmediatas o en un corto periodo de tiempo.

Partiendo de este contexto, el procesamiento a bordo de los datos hiperespectrales adquiridos se ha convertido en una solución muy atrayente y competitiva, pero aún quedan muchos pasos que dar para que se convierta en una realidad. Lamentablemente, los algoritmos tradicionalmente propuestos para el análisis de los datos hiperespectrales normalmente se han abordado como entes independientes utilizando aquellos métodos matemáticos que maximicen la calidad de los resultados obtenidos. Sin embargo, esto da lugar a propuestas algorítmicas muy complejas, difícilmente implementables debido a la ejecución de operaciones computacionalmente costosas, intensivas en consumo de memoria y recursos *hardware* y caracterizadas por altas dependencias de datos. Todo ello ha dado lugar a que en los últimos años se hayan invertido muchos esfuerzos en la búsqueda de métodos alternativos más ligeros, que puedan incluso ser adaptados a los requerimientos impuestos por las arquitecturas de cómputo paralelo más habitualmente empleadas en este tipo de aplicaciones, como son las *FPGAs* y las *GPUs*.

Sin embargo, el escenario anteriormente planteado se vuelve aún más complicado cuando distintos procesos de análisis hiperespectral deben ser ejecutados de manera simultánea y coexistir en un único dispositivo de cómputo asegurando respuestas en tiempo real. En este sentido, la solución más simple y comúnmente utilizada se basa en la selección de distintos algoritmos para cada aplicación a llevar a cabo y acelerarlos usando dispositivos de cómputo paralelo. El problema radica en el momento de su ejecución concurrente en un mismo dispositivo *hardware* en escenarios caracterizados por restricciones en términos de consumo, recursos de cómputo, peso y espacio disponibles, como son, por ejemplo, los drones. El principalmente fundamento de la imposibilidad de realización de la solución anteriormente planteada se basa en que normalmente las etapas de desarrollo algorítmicas y la de implementación y aceleración *hardware* se abordan de manera totalmente independiente, lo que da lugar a implementaciones totalmente ineficientes.

Por lo tanto, hay una gran necesidad en la literatura de soluciones algorítmicas que tengan en consideración las actuales restricciones y limitaciones existentes en las aplicaciones demandadas hoy en día. Además, también se requiere de la definición de nuevas alternativas algorítmicas que tengan en cuenta la causalidad inherente a los procesos de captura realizados por sistemas de tipo *pushbroom* y *whiskbroom*. Es por ello que para obtener una respuesta en tiempo real en este tipo de aplicaciones es necesario disponer de métodos algorítmicos capaces de procesar de manera independiente bloques de píxeles y que además, descarten la definición de restricciones espaciales. Por supuesto, esto también deriva en la reducción de la cantidad de datos a almacenar y procesar en un cierto instante de tiempo y por ende, la cantidad de recursos de cómputo necesarios y los tiempos de ejecución.

Partiendo del contexto descrito hasta ahora, la realización de esta Tesis Doctoral ha contribuido al campo del procesamiento abordo y en tiempo real de las imágenes hiperespectrales en aplicaciones donde múltiples técnicas de análisis deban coexistir en un único dispositivo de cómputo. Por lo tanto, esta Tesis Doctoral ha ayudado a abordar las necesidades existentes con los siguientes logros:

- Desarrollando soluciones algorítmicas *hardware-friendly* que han sido concebidas desde sus etapas iniciales de diseño para ser implementadas y aceleradas en dispositivos de cómputo paralelo.
- Adaptando los métodos propuestos para poder procesar bloques de píxeles de manera independiente y brindar soluciones competitivas para aplicaciones basadas en sistemas de adquisición de datos del tipo *pushbroom* y *whiskbroom*.
- Definiendo un conjunto de operaciones que permitan la ejecución simultánea de múltiples tareas de análisis con la ventaja de compartir las operaciones más computacionalmente complejas y además, reutilizar las variables de salida para la consecución de cada método independiente. Esto por supuesto deriva en la disminución de los recursos cómputo, los tiempos de ejecución y los esfuerzos humanos comparado con la solución tradicionalmente adoptada basada en la implementación independiente de distintos métodos matemáticos.
- Demostrando la capacidad de ejecución en tiempo real de las soluciones algorítmicas propuestas a través de su implementación en dispositivos de cómputo paralelo basados en *FPGAs* y *GPUs* de bajo consumo.

B.2 Objetivos y metodología de trabajo

El principal objetivo de esta tesis es proveer a la comunidad científica con un conjunto de operaciones capaces de extraer información espectral de utilidad para la realización de múltiples técnicas de análisis hiperespectral. El hecho de centrarse en la utilización de un único método matemático es especialmente beneficioso para la aceleración *hardware* de estos procesos. Por una parte, esto se traduce en un ahorro de tiempo, costes y esfuerzo humano durante la etapa de implementación *hardware* de estas soluciones algorítmicas pues sólo un único método matemático debe ser estudiado, comprendido y desarrollado. Por otra parte, esta metodología permite la ejecución conjunta de diversas tareas de procesamiento con la ventaja de compartir las operaciones más computacionalmente costosas y complejas, con los beneficios derivado de ello. Además, la metodología propuesta presenta ya desde su definición una carga computacional relativamente baja a través del uso de una serie de operaciones pensadas para ser eficientemente implementadas en dispositivos de cómputo paralelo.

Esto objetivo, a su vez, se ha dividido en los siguientes seis subobjetivos:

1. Desarrollar un conjunto de operaciones de cómputo *hardware-friendly*. Estas operaciones deben ser capaces de extraer características de las imágenes hiperespectrales de gran utilidad para su análisis desde distintas perspectivas de procesamiento. Además, deben presentar una alta naturaleza paralelizable que facilite su implementación y aceleración en dispositivos de cómputo paralelo con objeto de obtener respuestas en tiempo real. Para ello, deben poder definirse tanto en coma flotante como en aritmética entera, haciendo uso del concepto de punto fijo, en aras de poder adaptarse más eficientemente a la arquitectura *hardware* en la que vayan a ser implementadas. Finalmente, estas operaciones deben poder ser ejecutadas sobre bloques independientes de píxeles en aras de facilitar su procesamiento inmediato y en tiempo real, evitando por ende la definición de restricciones específicas de alineación espacial. Esto permite la adaptabilidad de las operaciones propuestas a aplicaciones basadas en sistemas de captura del tipo *pushbroom/whiskbroom*.

2. Demostrar la viabilidad de la ejecución concurrente de múltiples técnicas de análisis hiperespectral basadas en el mismo método matemático. En este sentido, debe comprobarse que se puede llevar a cabo la ejecución simultánea de distintos tipos de técnicas de análisis hiperespectral sobre el mismo dispositivo *hardware* reutilizando

las variables de salida del conjunto de operaciones descritas en el suobjetivo inmediatamente superior. También se debe demostrar que el costo computacional general, los recursos *hardware* requeridos, así como los esfuerzos humanos invertidos durante la implementación de las soluciones algorítmicas propuestas se reducen considerablemente en comparación con escenarios donde métodos matemáticos distintos son independiente implementados.

3. Desarrollar un nuevo algoritmo para la detección de agentes anómalos basado en el conjunto de operaciones anteriormente citado. Dicho algoritmo debe ser capaz de encontrar píxeles raros que difieran notablemente de la distribución general de las firmas espectrales más dominantes, también conocida como distribución del *background*. Para ello, esta propuesta algorítmica debe cumplir con los requisitos impuestos por las aplicaciones de teledectección actuales y con la causalidad inherente a los procesos de capturas basados en sistemas del tipo *pushbroom/whiskbroom*.

4. Proponer un nuevo algoritmo *hardware-friendly* para la compresión con pérdidas de imágenes hiperespectrales basado en métodos de transformada. Esta propuesta debe ser capaz de realizar el proceso completo de decorrelación y reducción de los datos hiperespectrales así como su codificación entrópica. Además, debe presentar una baja complejidad computacional y un alto grado de paralelismo y escalabilidad en aras de que su ejecución pueda ser abordada de manera eficiente en escenarios a bordo de sistemas del tipo *pushbroom/whiskbroom*.

5. Verificar la idoneidad de los algoritmos desarrollados para aplicaciones que requieran de una respuesta en tiempo real a través de su implementación en diferentes dispositivos de cómputo paralelo tales como *GPUs* y *FPGAs*. Además, también se desea evaluar la eficiencia de estas arquitecturas en función de las características de la aplicación objetivo y su comportamiento utilizando notación en aritmética entera y en coma flotante.

B.3 Contribuciones generales y principales conclusiones extraídas

Las principales contribuciones y conclusiones extraídas de esta Tesis Doctoral en el campo del procesamiento a bordo y en tiempo real de las imágenes hiperespectrales y, en consecución con los objetivos planteados, son:

1. Se presenta un conjunto de operaciones capaces de extraer características de las imágenes hiperespectrales de gran utilidad para la ejecución de múltiples técnicas de procesamiento espectral. En consecuencia, esta propuesta permite la ejecución concurrente de muchas tareas diferentes al mismo tiempo, tales como detección de agentes anómalos; detección de objetivos de interés; compresión con pérdidas de imágenes hiperespectrales; clasificación y desmezclado; con la ventaja de compartir los núcleos operacionales más computacionalmente intensivos. Para ello, el conjunto de operaciones propuesto se basa en técnicas de proyecciones ortogonales y, más concretamente, en el conocido método de ortogonalización de Gram-Schmidt. Además, esta metodología presenta baja carga computacional ya que no involucra cálculos complejos con matrices ni la estimación de autovalores y autovectores, como realizan otros métodos del estado del arte como es el Análisis de Componentes Principales (PCA, de sus siglas en inglés *Principal Component Analysis*).

   Como novedad, el conjunto de operaciones propuesto se puede aplicar de manera eficiente e independiente en bloques de píxeles de la imagen sin requerir la definición de restricciones espaciales específicas. Esta característica distintiva hace de esta propuesta una solución prometedora para aplicaciones en tiempo real, especialmente cuando se trata de escenarios que utilizan sensores del tipo *pushbroom* que sensan los datos línea a línea.

   No obstante, uno de los mayores beneficios del *set* de operaciones propuesto radica en la definición de un conjunto de variables cuyos valores se encuentran definidos dentro de unos rangos numéricos conocidos de antemano. Esto permite conocer desde la etapa de diseño los valores máximos y mínimos que podrá alcanzar cada variable definida dentro de cada operación. Esta característica hace posible explotar el concepto de punto fijo en la aritmética entera de manera personalizada con objeto de representar la parte entera y decimal de las variables envueltas en el proceso. Por lo tanto, el conjunto propuesto de operaciones se puede adaptar fácilmente a los requisitos impuestos por los dispositivos de cómputo paralelo que se van a utilizar para su implementación. En este contexto, las *FPGAs* son en general más eficientes con el manejo de operaciones definidas en aritmética entera, mientras que las *GPUs* están optimizadas para el procesamiento en paralelo de operaciones en coma flotante utilizado para ello cientos de pequeños núcleos de procesamiento.

2. En esta Tesis Doctoral se ha desarrollado un nuevo algoritmo para la detección de espectros anómalos, denominado *A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery* (LbL-FAD). Dicho algoritmo ha sido especialmente diseñado

para contribuir a la literatura con algoritmos capaces de procesar las imágenes línea a línea y por tanto, está especialmente destinado a ser empleado en aplicaciones basadas en sistemas de adquisición del tipo *pushbroom/whiskbroom*. En este sentido, el algoritmo LbL-FAD es capaz de procesar de forma independiente bloques de píxeles hiperespectrales sin tener en cuenta ningún requisito de alineación espacial.

Para la detección de firmas espectrales anómalas, el algoritmo LbL-FAD se enfoca en el cálculo de un subespacio ortogonal al definido por la distribución del *background* donde dichos agentes anómalos son más fácilmente detectables. Para ello, el algoritmo LbL-FAD sigue una estrategia basada en proyecciones ortogonales y más concretamente, mediante el conjunto de operaciones propuesta en esta Tesis Doctoral. Por tanto, esta metodología permite excluir el uso de métodos tradicionales basados en transformaciones lineales, como son el PCA o el *Singular Value Descomposition* (SVD) cuyas operaciones presentan por naturaleza una alta complejidad computacional, y también evita la estimación de inversas de grandes matrices de datos.

3. También se ha propuesto una versión mejorada del algoritmo del estado del arte titulado *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) para la decorrelación espectral y compresión de las imágenes hiperespectrales. El algoritmo HyperLCA es un compresor basado en transformada que proporciona altas relaciones de compresión con una baja carga computacional. Como ventaja adicional, el algoritmo HyperLCA permite comprimir bloques de píxeles de la imagen de manera independiente. Esta característica promueve, por un lado, la reducción de los datos que son manejados a la vez, además de los recursos *hardware* destinados a ello y, por otro lado, se establece como una solución muy competitiva para la mayoría de las aplicaciones basadas en sistemas del tipo *pushbroom/whiskbroom*. Todo ello ayuda a allanar el camino hacia el logro de un sistema de compresión que actúe en tiempo real.

   Existe un trabajo previo donde se introdujo por primera vez el algoritmo HyperLCA. Éste estableció el punto de partida hacia el uso de técnicas de proyección ortogonal y, en particular el método de Gram-Schmidt, para la compresión de datos hiperespectrales. Como novedad, la metodología descrita en ese trabajo previo se ha ampliado con el fin de ser adaptada y poder ejecutarse con el conjunto de operaciones propuesto en esta Tesis Doctoral. Además, también se ha definido un sistema de compresión de imágenes hiperespectrales completo que incluye la etapa

de codificación entrópica de los vectores que conforman los datos comprimidos. Adicionalmente, se ha extendido la definición algorítmica del método de transformada en el que se basa el algoritmo HyperLCA para incluir su ejecución en aritmética entera.

Finalmente, se ha demostrado que el algoritmo HyperLCA es también capaz de conservar los píxeles más diferentes o anómalos tras el proceso de compresión/descompresión. Esto es crucial para la correcta operación de muchas técnicas de procesamiento de imágenes hiperespectrales a realizar con posterioridad al proceso de compresión de los datos. De hecho, el uso de un método mátemático común en la descripción del compresor HyperLCA y el detector de agentes anómalos LbL-FAD anteriormente mencionado, garantiza que el proceso de compresión/descompresión no afecte seriamente al rendimiento del proceso de detección de anomalías utilizando los datos transformados y descomprimidos. Por lo tanto, este rasgo distintivo hace posible adaptar las metodologías propuestas en esta Tesis Doctoral a distintos escenarios en función de la aplicación objetivo, asegurando resultados similares en todas las situaciones.

4. La viabilidad de la ejecución conjunta de múltiples técnicas de análisis hiperespectral basadas en el mismo método matemático también ha sido verificada en esta Tesis Doctoral. En concreto, se ha analizado la adecuación del conjunto de operaciones propuesto para la ejecución simultánea de múltiples técnicas de procesamiento hiperespectral. Esto proporciona varios beneficios, sobre todo en el campo del procesamiento a bordo de imágenes hiperespectrales cuando varias aplicaciones deben coexistir en el mismo dispositivo y ser ejecutadas en tiempo real. En primer lugar, el uso de esta metodología implica invertir menos tiempo y esfuerzo durante la etapa de implementación y aceleración *hardware*, ya que el mismo producto puede ser reutilizado por varios algoritmos destinados a diferentes aplicaciones. En segundo lugar, permite la ejecución de diversas tareas al mismo tiempo con la ventaja de compartir las operaciones más costosas desde el punto de vista computacional, reduciéndose así la carga computacional y los recursos *hardware* requeridos para ello.

   En particular, se ha verificado la idoneidad de la metodología propuesta para la ejecución simultánea de la compresión con pérdidas de imágenes hiperespectrales y la detección de firmas espectrales anómalas. Para atender esta problemática se han

propuesto dos soluciones algorítmicas optimizadas. La primera, denominada *Optimized proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (ADeLoC), busca la mayor precisión en los resultados en términos de detección y compresión. En este sentido, el enfoque seguido por el algoritmo ADeLoC asegura los mismos resultados que los algoritmos LbL-FAD e HyperLCA pero lanzando entre un 39-41% menos número de operaciones. La segunda propuesta, denominada *Hardware-friendly proposal for the simultaneous detection of anomalous pixels and the lossy compression of HSIs* (HADeLoC), prioriza la optimización de los recursos *hardware* y la minimización de los tiempos de ejecución a expensas de una pérdida en la precisión y la calidad de los resultados en la etapa de compresión comparado con la versión original del compresor HyperLCA. Por el contrario, se ha verificado que esta propuesta ejecuta entre un 55-59% menos operaciones que si los algoritmos LbL-FAD e HyperLCA se implementaran de manera independiente en el mismo dispositivo de cómputo, y un 27-30% menos operaciones que la versión denominada ADeLoC. Por lo tanto, la conclusión principal que se puede extraer del análisis realizado es que siempre existe un balance entre la calidad de los resultados que se deseen obtener, los recursos computacionales utilizados y el tiempo de ejecución requerido.

5. En aras de confirmar los beneficios derivados de desarrollar soluciones algorítmicas basadas en el mismo método matemático; en términos de reducción en los tiempos de ejecución, los recursos *hardware* empleados y el esfuerzo humano implicado; y también verificar la idoneidad de las propuestas algorítmicas desarrolladas para aplicaciones en tiempo real, estas han sido implementadas en diversos dispositivos de cómputo paralelo, tales como *FPGAs* y *GPUs*. En concreto, los algoritmos LbL-FAD, HyperLCA y HADeLoC fueron acelerados en plataformas SoC, de sus siglas en inglés *System on Chip*, basados en FPGAs de Xilinx. Además, el compresor HyperLCA también fue adaptado para su ejecución en plataformas embebidas basadas en GPU de NVIDIA.

   La definición de los aceleradores *hardware* para FPGA (HWacc) que implementan los distintos algoritmos propuestos en esta Tesis Doctoral se ha llevado a cabo mediante la combinación de módulos descritos en lenguaje de síntesis de alto nivel (HLS), de sus siglas en inglés *High Level Synthesis*, con lógica de pega definida y personalizada con VHDL (*Very High Density Language*). En este sentido, los HWacc que implementan cada una de las operaciones propuestas en esta Tesis Doctoral fueron

definidos usando herramientas HLS. Estos módulos se han reutilizado en la implementación de los distintos HWacc que ejecutan los algoritmos anteriormente citados. En consecución, los mayores esfuerzos se han focalizado en las conexiones entre los citados módulos para ejecutar las etapas específicas de cada algoritmo en cuestión. Para ello, los *buffers* de memoria y la lógica de pega que integra y orquesta todos los componentes de interconexión entre módulos se instancian e implementan utilizando lenguaje VHDL. Por lo tanto, todo ello implica menos tiempo y esfuerzo durante la etapa de implementación *hardware* ya que un mismo producto puede ser reutilizado por varias soluciones algorítmicas destinadas a propósitos distintos. En relación a la tasa promedio de *frames* hiperespectrales capaces de ser procesados por los HWacc desarrollados, estos son capaces de manejar entre 778-1600 *frames* de 1024 píxeles hiperespectrales en un segundo (FPS), datos que claramente superan el mínimo establecido en 330 FPS por la aplicación en concreto definida. Por lo tanto, queda aseguro y verificado el comportamiento en tiempo real de las soluciones propuestas.

En cuanto a la implementación en GPU realizada del algoritmo de compresión HyperLCA, se han estudiado tres modelos diferentes de implementación del proceso completo de compresión, viéndolos como una evolución hacia una configuración óptima que cumple con las limitaciones impuestas por la aplicación en concreto definida. Esta solución optimizada apuesta por canalizar las transferencias de datos entre la CPU y la GPU y también la ejecución de los *kernerls* que implementan las operaciones propuestas en esta Tesis, a través de una estrategia basada en un *pipeline* de los datos. Para ello, dicha propuesta aprovecha los beneficios derivados de la ejecución concurrente de los *kernerls* definidos a través del recurso de los *streams* instanciados en el modelo de programación de CUDA. En relación a la tasa promedio de *frames* hiperespectrales capaces de ser procesados por segundo, la implementación sobre GPU propuesta es capaz de garantizar un rendimineto en tiempo real ya que se llegan a gestionar de manera eficiente entre 452-923 FPS empleando la placa de desarrollo Jetson TX2 y más de 1730 FPS con una de las más recientes placas embebidas comercializadas por NVIDIA, en concreto, la Jetson Xavier NX.

6. Finalmente, también se ha analizado brevemente la posibilidad de extender el uso de las proyecciones ortogonales y, en particular, el método de ortogonalización de Gram-Schmidit ejecutado por el conjunto de operaciones propuestas en esta Tesis Doctoral, a otras técnicas de procesamiento de imágenes hiperespectrales, como son la selección de bandas, el desmezclado, la detección de objetivos de interés y la clasificación. Aunque el análisis realizado está lejos de ser tan exhaustivo como

el llevado a cabo para los algoritmos LbL-FAD y el HyperLCA, sí que representa un punto de partida hacia futuros trabajos de investigación dentro de la misma temática.

Con objeto de resumir gráficamente las contribuciones aportadas por esta Tesis Doctoral al campo del procesamiento de las imágenes hiperespectrales en tiempo real a bordo de plataformas de observación terrestre basadas en sistemas del tipo *pushbroom*, la Figura B.1 muestra los principales objetivos y contribuciones derivados de ella.

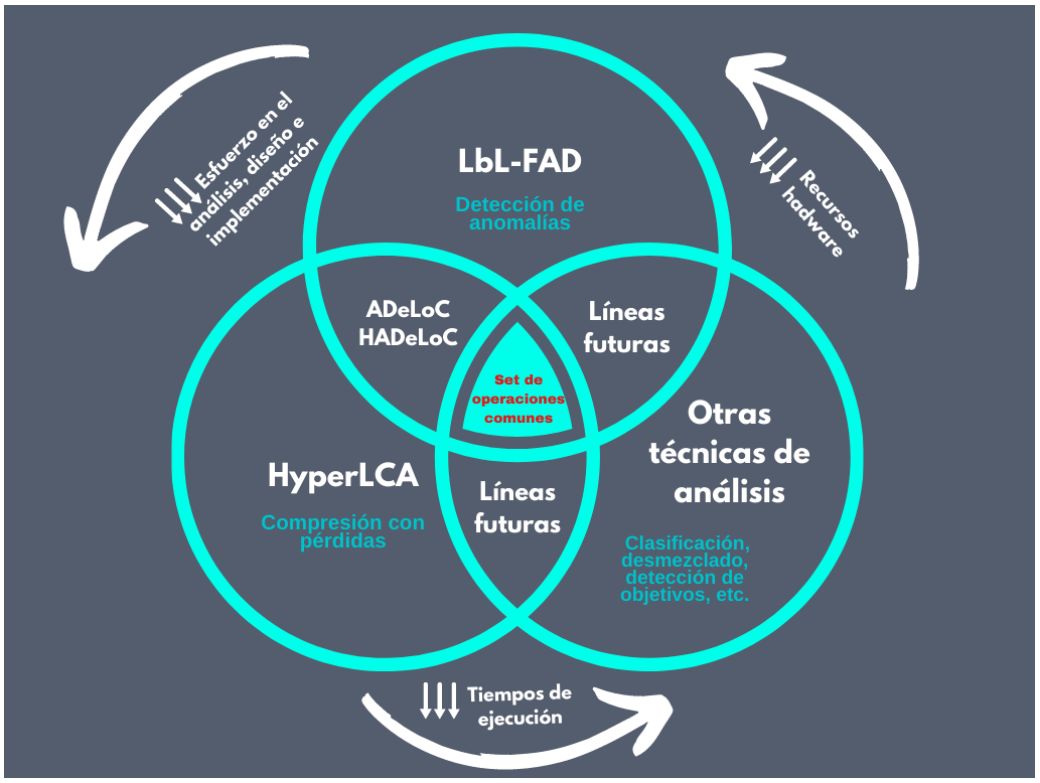


FIGURE B.1: Principales contribuciones de esta Tesis Doctoral.

Appendix C

Publications

C.1 Journals

The following publications are closely linked to the research goals defined for the realization of this Thesis:

[1] Caba, J., Díaz, M., Barba, J., Guerra, R., de la Torre, J.A., López, S. (2020). FPGA-Based On-Board Hyperspectral Imaging Compression: Benchmarking Performance and Energy Efficiency against GPU Implementations. 12 - 22, pp. 3741. MDPI Remote Sensing. https://doi.org/10.3390/rs12223741.

[2] Díaz, M., Guerra, R., Horstrand, P., López, S., López, J.F., Sarmiento, R. (2020). Towards the Concurrent Execution of Multiple Hyperspectral Imaging Applications by Means of Computationally Simple Operations. 12, pp. 1343. MDPI Remote Sensing. https://doi.org/10.3390/rs12081343.

[3] Díaz, M., Guerra, R., Horstrand, P., López, S. (2019). A Line-by-Line Fast Anomaly Detector for Hyperspectral Imagery. 57 - 11, pp. 8968 - 8982. IEEE Transactions on Geoscience and Remote Sensing. DOI: 10.1109/TGRS.2019.2923921.

[4] Guerra, R., Barrios, Y., Díaz, M., Báez, B., López, S., Sarmiento, R. (2019). A Hardware-Friendly Hyperspectral Lossy Compressor for Next-Generation Space-Grade Field Programmable Gate Arrays. 12 - 12, pp. 4813 - 4828. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. DOI: 10.1109/JSTARS.2019.2919791.

[5] Díaz, M., Guerra, R., Horstrand, P., Martel, E., López, S., López, J.F., Sarmiento, R. (2019). Real-Time Hyperspectral Image Compression Onto Embedded GPUs. 12 - 8, pp. 2792 - 2809. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. DOI: 10.1109/JSTARS.2019.2917088.

[6] Guerra, R., Barrios, Y., Díaz, M., Santos, L., López, S., Sarmiento, R. (2018). A new algorithm for the on-board compression of hyperspectral images. 10 - 3, pp. 428 - 469. MDPI Remote Sensing. DOI: 10.3390/rs10030428.

[7] Díaz, M., Guerra, R., López, S., Sarmiento, R. (2017). An Algorithm for an Accurate Detection of Anomalies in Hyperspectral Images With a Low Computational Complexity. 56 - 2, pp. 1159 - 1176. IEEE Transactions on Geoscience and Remote Sensing. DOI: 10.1109/TGRS.2017.2761019.

In addition, it has also collaborated in the following publications:

[8] Morales, A., Guerra, R., Horstrand, P., Diaz, M., Jimenez, A., Melian, J., López, S., López, J. F. (2020). A Multispectral Camera Development: From the Prototype Assembly until Its Use in a UAV System. Sensors. 20 - 21, pp. 6129. https://doi.org/10.3390/rs13050850.

[9] Melián, J.M., Jiménez, A., Díaz, M., Morales, A., Horstrand, P., Guerra, R., López, S., López, J.F. (2021). Real-time hyperspectral data transmission for UAV-based acquisition platforms. 13 - 5, pp. 850. MDPI Remote Sensing. https://doi.org/10.3390/rs13050850.

[10] Ortega, S., Guerra, R., Díaz, M., Fabelo, H., López, S, Callicó, G. (2019). Hyperspectral Push-Broom Microscope Development and Characterization. 7, pp. 122473 - 122491. IEEE Access. DOI: 10.1109/ACCESS.2019.2937729.

[11] Horstrand, P., Guerra, R., Rodríguez, A., Díaz, M., López, S., López, J.F. (2019). A UAV Platform based on a Hyperspectral Sensor for Image Capturing and On-Board Processing. 7 - 1, pp. 66919 - 66938. IEEE Access. DOI: 10.1109/ACCESS.2019.2913957.

[12] Horstrand, P., Díaz, M., Guerra, R., López, S., López, J.F. (2019). A Novel Hyperspectral Anomaly Detection Algorithm for Real-Time Applications With Push-Broom Sensors. 12 - 12, pp. 4787 - 4797. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. DOI: 10.1109/JSTARS.2019.2919911.

C.2 International Conferences

The following contributions to International Conferences are closely linked to the research goals defined for the realization of this Thesis:

[1] Díaz, M., Guerra, R., Horstrand, P., Martel, E., López, S., López, J.F., Roberto, S. (2019). Real-time hyperspectral image compression: a low consumption approach for UAV-based applications. In *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*.

[2] Díaz, M., López, S. (2019). Towards the efficient on-board processing of hyperspectral images. In *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*.

[3] Díaz, M., Guerra, R., López, S. (2019). A hardware-friendly anomaly detector for real-time applications with push-broom scanners. In *Workshop on Hyperspectral Image and Signal Processing Evolution in Remote Sensing (WHISPERS)*.

[4] Díaz, M., Guerra, R., López, S. (2019). A novel hyperspectral target detection algorithm for real-time applications with push-broom scanners. In *Workshop on Hyperspectral Image and Signal Processing Evolution in Remote Sensing (WHISPERS)*.

[5] Guerra, R., Díaz, M., Barrios, Y., López, S., Sarmiento, R. (2018). A Hardware-friendly algorithm for the on-board compression of hyperspectral images. In *Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*.

[6] Guerra, R., Díaz, M., Barrios, Y., López, S., Sarmiento, R. (2018). A hardware-friendly algorithm for compressing hyperspectral images. In *SPIE Remote Sensing*. DOI: https://doi.org/10.1117/12.2500493

[7] Díaz, M., López, S., Sarmiento, R. (2016). A new comparison of hyperspectral anomaly detection algorithms for real-time applications. In *SPIE Remote Sensing*. DOI: https://doi.org/10.1117/12.2244297

In addition, it has also collaborated in the works listed below:

[8] Horstrand, P., Guerra, R., Díaz, M., Morales, A., Jiménez, A., López, S., López, J.F. (2019). A spectral imaging system for precision agriculture: from its inception till a pre-commercial prototype. In *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*.

[9] Ortega, S., Guerra, R., Fabelo, H., Díaz, M., López, S., Callicó, G., Sarmiento, R. (2019). Low-Cost Hyperspectral Push-broom Microscope, targeting Smart Farming Applications. In *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*.

[10] Guerra, R., Horstrand, P., Díaz, M., López, S., López, J.F. (2019). Optimal UAV movement control for farming areas scanning using hyperspectral pushbrrom scanners. in *XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*.

[11] Díaz, M., Chanussot, J., Guerra, R., López, S., Sarmiento, R., Bertozzi, A.L. (2018). A novel highly parallel algorithm for the detection and tracking of chemical gas plumes using hyperspectral video sequences. In *Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*.

References

[1] Saba Daneshgar. *Remote sensing observations for monitoring coastal zones, Volturno River mouth case study.* PhD thesis, 04 2015.

[2] Raúl Celestino Guerra Hernández. *Towards the efficient processing of hyperspectral images: new hardware-friendly algorithms and OpenCL-based implementations.* PhD thesis, Universidad de las Palmas de Gran Canaria, 2017.

[3] María Lucana Santos Falcón. *Hyperspectral image compression onboard next-generation satellites: Implementation solutions on GPU and FPGAs.* PhD thesis, Universidad de Las Palmas de Gran Canaria, 2014.

[4] Raúl Guerra, Ernestina Martel, Jehandad Khan, Sebastián López, Peter Athanas, and Roberto Sarmiento. On the evaluation of different high-performance computing platforms for hyperspectral imaging: An OpenCL-based approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(11):4879–4897, 2017.

[5] María Díaz, Raúl Guerra, Pablo Horstrand, Ernestina Martel, Sebastián López, José F. López, and Sarmiento Roberto. Real-time hyperspectral image compression onto embedded GPUs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 1–18, 2019. ISSN 1939-1404. doi: 10.1109/JSTARS.2019.2917088.

[6] Julie Transon, Raphaël d'Andrimont, Alexandre Maugnard, and Pierre Defourny. Survey of hyperspectral earth observation applications from space in the Sentinel-2 context. *Remote Sensing*, 10(2):157, 2018.

[7] Megandhren Govender, K Chetty, and Hartley Bulcock. A review of hyperspectral remote sensing and its application in vegetation and water resource studies. *Water Sa*, 33(2), 2007.

[8] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, and A. Plaza. Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):37–78, Dec 2017. ISSN 2168-6831. doi: 10.1109/MGRS.2017.2762087.

[9] Alexander F.H. Goetz. Three decades of hyperspectral remote sensing of the Earth: A personal view. *Remote Sensing of Environment*, 113:S5–S16, sep 2009. ISSN 00344257. doi: 10.1016/j.rse.2007.12.014. URL https://linkinghub.elsevier.com/retrieve/pii/S003442570900073X.

[10] John A. Richards. *Remote Sensing Digital Image Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30061-5. doi: 10.1007/978-3-642-30062-2. URL http://link.springer.com/10.1007/978-3-642-30062-2.

[11] Andreas Birk, Burkhard Wiggerich, Heiko Bülow, Max Pfingsthorn, and Sören Schwertfeger. Safety, security, and rescue missions with an unmanned aerial vehicle (UAV). *Journal of Intelligent & Robotic Systems*, 64(1):57–76, 2011.

[12] Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of biomedical optics*, 19(1):010901, 2014.

[13] Muhammad Jaleed Khan, Hamid Saeed Khan, Adeel Yousaf, Khurram Khurshid, and Asad Abbas. Modern trends in hyperspectral image analysis: a review. *IEEE Access*, 6:14118–14129, 2018.

[14] Petra Tatzer, Markus Wolf, and Thomas Panner. Industrial application for inline material sorting using hyperspectral imaging in the nir range. *Real-Time Imaging*, 11(2):99–107, 2005.

[15] Silvia Serranti, Aldo Gargiulo, and Giuseppe Bonifazi. Hyperspectral imaging for process and quality control in recycling plants of polyolefin flakes. *Journal of Near Infrared Spectroscopy*, 20(5):573–581, 2012.

[16] Hans Grahn and Paul Geladi. *Techniques and applications of hyperspectral image analysis*. John Wiley & Sons, 2007.

[17] G Vane, M Chrisp, H Enmark, S Macenka, et al. Airborne visible/infrared imaging spectrometer (AVIRIS): an advanced tool for earth remote sensing. In *From Res. Towards Operational Use*, volume 2, 1984.

[18] SK Babey and CD Anger. A compact airborne spectrographic imager (CASI). In *Quantitative Remote Sensing: An Economic Tool for the Nineties, Volume 1*, pages 1028–1031, 1989.

[19] Peter A Mitchell. Hyperspectral digital imagery collection experiment (HYDICE). In *Geographic Information Systems, Photogrammetry, and Geological/Geophysical Remote Sensing*, volume 2587, pages 70–95. International Society for Optics and Photonics, 1995.

[20] P Strobl, R Richter, A Mueller, F Lehmann, D Oertel, S Tischler, and A Nielsen. Dais system performance, first results from the 1995 evaluation campaigns. In *Proceedings from the Second International Airborne Remote Sensing Conference and Exhibition*, volume 2, pages 325–334. Citeseer, 1996.

[21] B Kunkel, F Blechinger, D Viehmann, H VAN DER PIEPEN, and R Doerffer. ROSIS imaging spectrometer and its potential for ocean parameter measurements (airborne and space-borne). *International Journal of Remote Sensing*, 12(4):753–761, 1991.

[22] Kai Makisara, Marko Meinander, Markku Rantasuo, Jukko Okkonen, Mauri Aikio, and Kaarlo Sipola. Airborne imaging spectrometer for applications (AISA). In *Proceedings of IGARSS'93-IEEE International Geoscience and Remote Sensing Symposium*, pages 479–481. IEEE, 1993.

[23] T Cocks, R Jenssen, A Stewart, I Wilson, and T Shields. The HyMap airborne hyperspectral sensor: The system, calibration and performance. In *Proceedings of the 1st EARSeL workshop on Imaging Spectroscopy*, pages 37–42. EARSeL, 1998.

[24] Thomas A Ellis, Jeffrey Myers, Patrick Grant, Steven Platnick, Daniel C Guerin, John Fisher, Kai Song, Joseph Kimchi, Louis Kilmer, Daniel D LaPorte, et al. The NASA enhanced MODIS airborne simulator. In *Earth Observing Systems XVI*, volume 8153, page 81530N. International Society for Optics and Photonics, 2011.

[25] A Müller and A Hausold. The airborne imaging spectrometer data acquisition programme in 1998 1999 and 2000. In *The Digital Airborne Spectrometer Experiment (DAISEX)*, volume 499, page 7, 2001.

[26] Jay S Pearlman, Pamela S Barry, Carol C Segal, John Shepanski, Debra Beiso, and Stephen L Carman. Hyperion, a space-based imaging spectrometer. *IEEE Transactions on Geoscience and Remote Sensing*, 41(6):1160–1173, 2003.

[27] Mike A Cutter, Dan R Lobb, and Robert A Cockshott. Compact high resolution imaging spectrometer (CHRIS). *Acta Astronautica*, 46(2-6):263–268, 2000.

[28] Ettore Lopinto and Cristina Ananasso. The prisma hyperspectral mission. In *proceedings of the 33rd EARSeL Symposium Towards Horizon*, volume 2020, 2013.

[29] Hermann Kaufmann, Karl Segl, Luis Guanter, Stefan Hofer, K-P Foerster, Timo Stuffler, Andreas Mueller, Rudolf Richter, Heike Bach, Patrick Hostert, et al. Environmental mapping and analysis program (EnMAP)-recent advances and status. In *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 4, pages IV–109. IEEE, 2008.

[30] Jose AJ Berni, Pablo J Zarco-Tejada, Lola Suárez, and Elias Fereres. Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *IEEE Transactions on geoscience and Remote Sensing*, 47 (3):722–738, 2009.

[31] Telmo Adão, Jonáš Hruška, Luís Pádua, José Bessa, Emanuel Peres, Raul Morais, and Joaquim João Sousa. Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry. *Remote Sensing*, 9 (11):1110, 2017.

[32] P. Horstrand, R. Guerra, M. DÍaz, A. Morales, A. Jiménez, S. López, and J. F. López. A spectral imaging system for precision agriculture: From its inception till a pre-commercial prototype. In *2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6, 2019.

[33] Alberto Ortiz, Alfonso Rodríguez, Raúl Guerra, Sebastián López, Andrés Otero, Roberto Sarmiento, and Eduardo De la Torre. A runtime-scalable and hardware-accelerated approach to on-board linear unmixing of hyperspectral images. *Remote Sensing*, 10(11):1790, 2018.

[34] Alberto G Villafranca, Jordi Corbera, Francisco Martín, and Juan Fernando Marchán. Limitations of hyperspectral earth observation on small satellites. *Journal of Small Satellites*, 1(1):19–29, 2012.

[35] Rico Valentino, Woo-Sung Jung, and Young-Bae Ko. A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (UAV) clustering networks. *Sensors*, 18(11):3751, 2018.

[36] Antonio Plaza, Qian Du, Yang-Lang Chang, and Roger 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.

[37] Adrián Alcolea, Mercedes E Paoletti, Juan M Haut, Javier Resano, and Antonio Plaza. Inference in supervised spectral classifiers for on-board hyperspectral imaging: An overview. *Remote Sensing*, 12(3):534, 2020.

[38] Sebastian Lopez, Tanya Vladimirova, Carlos Gonzalez, Javier Resano, Daniel Mozos, and Antonio Plaza. The promise of reconfigurable computing for hyperspectral imaging onboard systems: A review and trends. *Proceedings of the IEEE*, 101(3): 698–722, 2013.

[39] Alan D George and Christopher M Wilson. Onboard processing with hybrid and reconfigurable computing on small satellites. *Proceedings of the IEEE*, 106(3):458–470, 2018.

[40] Qian Du and Reza Nekovei. Fast real-time onboard processing of hyperspectral imagery for detection and classification. *Journal of Real-Time Image Processing*, 4 (3):273–286, 2009.

[41] Ian Blanes, Joan Serra-Sagrista, Michael W Marcellin, and Joan Bartrina-Rapesta. Divide-and-conquer strategies for hyperspectral image processing: A review of their benefits and advantages. *IEEE Signal Processing Magazine*, 29(3):71–81, 2012.

[42] Mike Estlick, Miriam Leeser, James Theiler, and John J Szymanski. Algorithmic transformations in the implementation of k-means clustering on reconfigurable hardware. In *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, pages 103–110, 2001.

[43] Naoufal Amrani, Joan Serra-Sagristà, Valero Laparra, Michael W Marcellin, and Jesus Malo. Regression wavelet analysis for lossless coding of remote-sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 54(9):5616–5627, 2016.

[44] Chein-I Chang and Shao-Shan Chiang. Anomaly detection and classification for hyperspectral imagery. *IEEE transactions on geoscience and remote sensing*, 40(6): 1314–1325, 2002.

[45] Lifu Zhang, Bo Peng, Feizhou Zhang, Lizhe Wang, Hongming Zhang, Peng Zhang, and Qingxi Tong. Fast real-time causal linewise progressive hyperspectral anomaly

detection via cholesky decomposition. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(10):4614–4629, 2017.

[46] Chunhui Zhao and Xifeng Yao. Fast real-time kernel RX algorithm based on cholesky decomposition. *IEEE Geoscience and Remote Sensing Letters*, (99):1–5, 2018.

[47] Shih-Yu Chen, Yulei Wang, Chao-Cheng Wu, Chunhong Liu, and Chen-I Chang. Real-time causal processing of anomaly detection for hyperspectral imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 50(2):1511–1534, 2014.

[48] Pedram Ghamisi, Naoto Yokoya, Jun Li, Wenzhi Liao, Sicong Liu, Javier Plaza, Behnood Rasti, and Antonio Plaza. Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):37–78, 2017.

[49] Prasad S Thenkabail, John G Lyon, and Alfredo Huete. *Fundamentals, Sensor Systems, Spectral Libraries, and Data Mining for Vegetation*. CRC Press, 2018.

[50] Thomas Lillesand, Ralph W Kiefer, and Jonathan Chipman. *Remote sensing and image interpretation*. John Wiley & Sons, 2014.

[51] Chein-I Chang. *Hyperspectral data exploitation: theory and applications*. John Wiley & Sons, 2007.

[52] Teresa G Cervero, Julián Caba, Sebastián López, Julio Daniel Dondo, Roberto Sarmiento, Fernando Rincón, and Juan López. A scalable and dynamically reconfigurable FPGA-based embedded system for real-time hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8 (6):2894–2911, 2014.

[53] Alessandro Biondi and Giorgio Buttazzo. Timing-aware FPGA partitioning for real-time applications under dynamic partial reconfiguration. In *2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 172–179. IEEE, 2017.

[54] Gary A Shaw and Hsiaohua K Burke. Spectral imaging for remote sensing. *Lincoln laboratory journal*, 14(1):3–28, 2003.

[55] José M Bioucas-Dias, Antonio Plaza, Gustavo Camps-Valls, Paul Scheunders, Nasser Nasrabadi, and Jocelyn Chanussot. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geoscience and remote sensing magazine*, 1(2): 6–36, 2013.

[56] Peg Shippert. Introduction to hyperspectral image analysis. *Online Journal of Space Communication*, 3(2003):13, 2003.

[57] Dimitris Manolakis and Gary Shaw. Detection algorithms for hyperspectral imaging applications. *IEEE signal processing magazine*, 19(1):29–43, 2002.

[58] Nasser M Nasrabadi. Hyperspectral target detection: An overview of current and future challenges. *IEEE Signal Processing Magazine*, 31(1):34–44, 2014.

[59] Stefania Matteoli, Marco Diani, and Giovanni Corsini. A tutorial overview of anomaly detection in hyperspectral images. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):5–28, 2010.

[60] Irving S Reed and Xiaoli Yu. Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(10):1760–1770, 1990.

[61] Nasser M Nasrabadi. Regularization for spectral matched filter and RX anomaly detector. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XIV*, volume 6966, page 696604. International Society for Optics and Photonics, 2008.

[62] Yang Xu, Zebin Wu, Jun Li, Antonio Plaza, and Zhihui Wei. Anomaly detection in hyperspectral images based on low-rank and sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 54(4):1990–2000, 2016.

[63] Yuxiang Zhang, Bo Du, Liangpei Zhang, and Shugen Wang. A low-rank and sparse matrix decomposition-based mahalanobis distance method for hyperspectral anomaly detection. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3): 1376–1389, 2016.

[64] Wei Li and Qian Du. Collaborative representation for hyperspectral anomaly detection. *IEEE Transactions on geoscience and remote sensing*, 53(3):1463–1474, 2015.

[65] Consultative Committee for Space Data Systems (CCSDS). *Image Data Compression. CCSDS, Green Book 120.1-G-2.* Available Online: https://public.ccsds.org/Pubs/120x1g2.pdf, . (Accessed on 10 July 2020).

[66] Giovanni Motta, Francesco Rizzo, and James A Storer. *Hyperspectral data compression*. Springer Science & Business Media, 2006.

[67] Qian Du and James E Fowler. Hyperspectral image compression using JPEG2000 and principal component analysis. *IEEE Geoscience and Remote sensing letters*, 4 (2):201–205, 2007.

[68] Qian Du and James E Fowler. Low-complexity principal component analysis for hyperspectral image compression. *The International Journal of High Performance Computing Applications*, 22(4):438–448, 2008.

[69] Qian Du, Nam Ly, and James E Fowler. An operational approach to PCA + JPEG2000 compression of hyperspectral imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2237–2245, 2013.

[70] Miloš Radosavljević, Branko Brkljač, Predrag Lugonja, Vladimir Crnojević, Željen Trpovski, Zixiang Xiong, and Dejan Vukobratović. Lossy compression of multispectral satellite images with application to crop thematic mapping: A hevc comparative study. *Remote Sensing*, 12(10):1590, 2020.

[71] Ian Blanes, Enrico Magli, and Joan Serra-Sagrista. A tutorial on image compression for optical space imaging systems. *IEEE Geoscience and Remote Sensing Magazine*, 2(3):8–26, 2014.

[72] Marco Conoscenti, Riccardo Coppola, and Enrico Magli. Constant SNR, rate control, and entropy coding for predictive lossy hyperspectral image compression. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7431–7441, 2016.

[73] Mahesh Pal and Giles M Foody. Feature selection for classification of hyperspectral data by SVM. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5):2297–2307, 2010.

[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, April 2012. ISSN 2151-1535. doi: 10.1109/JSTARS.2012.2194696.

[75] Joseph W Boardman. Automating spectral unmixing of AVIRIS data using convex geometry concepts. 1993.

[76] José MP Nascimento and José MB Dias. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE transactions on Geoscience and Remote Sensing*, 43(4):898–910, 2005.

[77] Michael E Winter. N-findr: An algorithm for fast autonomous spectral end-member determination in hyperspectral data. In *Imaging Spectrometry V*, volume 3753, pages 266–275. International Society for Optics and Photonics, 1999.

[78] Daniel C Heinz et al. 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.

[79] José M Bioucas-Dias and Mário AT Figueiredo. Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing. In *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pages 1–4. IEEE, 2010.

[80] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[81] José M Bioucas-Dias and José MP Nascimento. Hyperspectral subspace identification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2435–2445, 2008.

[82] Gustavo Camps-Valls, Devis Tuia, Luis Gómez-Chova, Sandra Jiménez, and Jesús Malo. Remote sensing image processing. *Synthesis Lectures on Image, Video, and Multimedia Processing*, 5(1):1–192, 2011.

[83] Sicong Liu, Daniele Marinelli, Lorenzo Bruzzone, and Francesca Bovolo. A review of change detection in multitemporal hyperspectral images: Current techniques, applications, and challenges. *IEEE Geoscience and Remote Sensing Magazine*, 7 (2):140–158, 2019.

[84] Milica Orlandić, Johan Fjeldtvedt, and Tor Arne Johansen. A parallel FPGA implementation of the CCSDS-123 compression algorithm. *Remote Sensing*, 11(6):673, 2019.

[85] Luis Alberto Aranda, Antonio Sánchez, Francisco Garcia-Herrero, Yubal Barrios, Roberto Sarmiento, and Juan Antonio Maestro. Reliability analysis of the SHyLoC CCSDS123 IP core for lossless hyperspectral image compression using COTS FPGAs. *Electronics*, 9(10), 2020. ISSN 2079-9292. doi: 10.3390/electronics9101681. URL https://www.mdpi.com/2079-9292/9/10/1681.

[86] Luis Alberto Aranda, Pedro Reviriego, and Juan Antonio Maestro. Toward a fault-tolerant star tracker for small satellite applications. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5):3421–3431, 2020.

[87] Justin A Hogan, Raymond J Weber, and Brock J LaMeres. Reliability analysis of field-programmable gate-array-based space computer architectures. *Journal of Aerospace Information Systems*, 14(4):247–258, 2017.

[88] Solomon Banteywalu, Baseem Khan, Valentijn De Smedt, and Paul Leroux. A novel modular radiation hardening approach applied to a synchronous buck converter. *Electronics*, 8(5):513, 2019.

[89] Pablo Horstrand, Raul Guerra, Aythami Rodríguez, María Díaz, Sebastián López, and José Fco López. A UAV platform based on a hyperspectral sensor for image capturing and on-board processing. *IEEE Access*, 2019.

[90] Kejie Lu, Junfei Xie, Yan Wan, and Shengli Fu. Toward UAV-based airborne computing. *IEEE Wireless Communications*, 26(6):172–179, 2019.

[91] Interuniversity Microelectronics Centre (IMEC). *Hyperspectral drone camera system for application development.* Available Online: https://www.imec-int.com/drupal/sites/default/files/inline-files/UAV%20SNmosaic%20VIS%2BNIR%20hyperspectral%20imaging%20camera.pdf. (Accessed on 13 April 2020).

[92] Fredrik C Bruhn, Nandinbaatar Tsog, Fabian Kunkel, Oskar Flordal, and Ian Troxel. Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space. *CEAS Space Journal*, 12(4):551–564, 2020.

[93] Robert Wright, Miguel Nunes, Paul Lucey, Luke Flynn, Thomas George, Sarath Gunapala, David Ting, Alexander Soibel, Chiara Ferrari-Wong, Abigail Flom, et al. Hyti: thermal hyperspectral imaging from a cubesat platform. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 4982–4985. IEEE, 2019.

[94] Caleb Adams, Allen Spain, Jackson Parker, Matthew Hevert, James Roach, and David Cotten. Towards an integrated GPU accelerated SoC as a flight computer for small satellites. In *2019 IEEE Aerospace Conference*, pages 1–7. IEEE, 2019.

[95] George A Salazar and Glen F Steele. Commercial off-the-shelf (COTS) graphics processing board (gpb) radiation test evaluation report. 2013.

[96] Umar Ibrahim Minhas, Samuel Bayliss, and George A Constantinides. GPU vs FPGA: A comparative analysis for non-standard precision. In *International Symposium on Applied Reconfigurable Computing*, pages 298–305. Springer, 2014.

[97] Analog Devices. Fixed-point vs floating-point digital signal processing, 2010.

[98] DSP Berten. GPU vs FPGA performance comparison. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays-FPGA'17*, 2016.

[99] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar. Reducing power by optimizing the necessary precision/range of floating-point arithmetic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):273–286, 2000. doi: 10.1109/92.845894.

[100] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[101] Ambrose Finnerty and Hervé Ratigner. Reduce power and cost by converting from floating point to fixed point. *WP491 (v1. 0)*, 2017.

[102] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. *Remote sensing of environment*, 113:S110–S122, 2009.

[103] Nor Rizuan Mat Noor and Tanya Vladimirova. Integer KLT design space exploration for hyperspectral satellite image compression. In *International Conference on Hybrid Information Technology*, pages 661–668. Springer, 2011.

[104] Youcef Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992.

[105] Raúl Guerra, Lucana Santos, Sebastián López, and Roberto Sarmiento. A new fast algorithm for linearly unmixing hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 53(12):6752–6765, 2015.

[106] Barbara Penna, Tammam Tillo, Enrico Magli, and Gabriella Olmo. Transform coding techniques for lossy hyperspectral data compression. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1408–1421, 2007.

[107] Raúl Guerra, Yubal Barrios, María Díaz, Lucana Santos, Sebastián López, and Roberto Sarmiento. A new algorithm for the on-board compression of hyperspectral images. *Remote Sensing*, 10(3):428, 2018.

[108] Dimitris Manolakis, Christina Siracusa, and Gary Shaw. Hyperspectral subpixel target detection using the linear mixing model. *IEEE transactions on geoscience and remote sensing*, 39(7):1392–1409, 2001.

[109] Chein-I Chang. Orthogonal subspace projection (OSP) revisited: A comprehensive study and analysis. *IEEE transactions on geoscience and remote sensing*, 43(3): 502–518, 2005.

[110] Joseph C Harsanyi and C-I Chang. Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach. *IEEE Transactions on geoscience and remote sensing*, 32(4):779–785, 1994.

[111] Hamid Jafarzadeh and Mahdi Hasanlou. An unsupervised binary and multiple change detection approach for hyperspectral imagery based on spectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[112] H Jafarzadeh and M Hasanlou. Assessing and comparing the performance of endmember extraction methods in multiple change detection using hyperspectral data. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.

[113] Alp Ertürk and Antonio Plaza. Informative change detection by unmixing for hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 12(6):1252–1256, 2015.

[114] Sicong Liu, Lorenzo Bruzzone, Francesca Bovolo, and Peijun Du. Unsupervised multitemporal spectral unmixing for detecting multiple changes in hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(5):2733–2748, 2016.

[115] Alp Ertürk, Marian-Daniel Iordache, and Antonio Plaza. Sparse unmixing-based change detection for multitemporal hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(2):708–719, 2015.

[116] Hongjun Su, Peijun Du, and Qian Du. Semi-supervised dimensionality reduction using orthogonal projection divergence-based clustering for hyperspectral imagery. *Optical Engineering*, 51(11):111715, 2012.

[117] C. Chang, W. Xiong, H. Chen, and J. Chai. Maximum orthogonal subspace projection approach to estimating the number of spectral signal sources in hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):504–520, June 2011. ISSN 1941-0484. doi: 10.1109/JSTSP.2011.2134068.

[118] 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 *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pages 1010–1015, Dec 2011.

[119] H. Li and C. Chang. Linear spectral unmixing using least squares error, orthogonal projection and simplex volume for hyperspectral images. In *2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, June 2015. doi: 10.1109/WHISPERS.2015.8075424.

[120] Heesung Kwon and N. M. Nasrabadi. Kernel orthogonal subspace projection for hyperspectral signal classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(12):2952–2962, Dec 2005. ISSN 1558-0644. doi: 10.1109/TGRS.2005.857904.

[121] Hsuan Ren and Chein-I Chang. Automatic spectral target recognition in hyperspectral imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1232–1249, Oct 2003. ISSN 2371-9877.

[122] Ernestina Martel, Raúl Guerra, Sebastián López, and Roberto Sarmiento. A GPU-based processing chain for linearly unmixing hyperspectral images. *IEEE journal of selected topics in applied earth observations and remote sensing*, 10(3):818–834, 2017.

[123] Chuanmin Hu, Lian Feng, Zhongping Lee, Curtiss Davis, Antonio Mannino, Charles Mcclain, and Bryan Franz. Dynamic range and sensitivity requirements of satellite

ocean color sensors: Learning from the past. *Applied optics*, 51:6045–62, 09 2012. doi: 10.1364/AO.51.006045.

[124] Ashok Kumar, Sanjeev Mehta, Sandip Paul, R. Parmar, and R Samudraiah. Dynamic range enhancement of remote sensing electro-optical imaging systems. 12 2012.

[125] Dimitris Manolakis, David Marden, and Gary A Shaw. Hyperspectral image processing for automatic target detection applications. *Lincoln laboratory journal*, 14 (1):79–116, 2003.

[126] Dimitris Manolakis, Eric Truslow, Michael Pieper, Thomas Cooley, and Michael Brueggeman. Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms. *IEEE Signal Processing Magazine*, 31(1):24–33, 2014.

[127] María Díaz, Raúl Guerra, Sebastián López, and Roberto Sarmiento. An algorithm for an accurate detection of anomalies in hyperspectral images with a low computational complexity. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2): 1159–1176, 2017.

[128] Benyamin Hosseiny and Reza Shah-Hosseini. A hyperspectral anomaly detection framework based on segmentation and convolutional neural network algorithms. *International Journal of Remote Sensing*, 41(18):6946–6975, 2020.

[129] M. Díaz, R. Guerra, P. Horstrand, S. López, and R. Sarmiento. A line-by-line fast anomaly detector for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 57(11):8968–8982, 2019.

[130] Stefania Matteoli, Marco Diani, and Giovanni Corsini. Improved estimation of local background covariance matrix for anomaly detection in hyperspectral images. *Optical Engineering*, 49(4):046201, 2010.

[131] Qiandong Guo, Bing Zhang, Qiong Ran, Lianru Gao, Jun Li, and Antonio Plaza. Weighted-RXD and linear filter-based RXD: Improving background statistics estimation for anomaly detection in hyperspectral imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2351–2366, 2014.

[132] Heesung Kwon and Nasser M Nasrabadi. Kernel RX-algorithm: A nonlinear anomaly detector for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 43(2):388–397, 2005.

[133] A Schaum. Joint subspace detection of hyperspectral targets. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 3. IEEE, 2004.

[134] Chein-I Chang. Orthogonal subspace projection (OSP) revisited: A comprehensive study and analysis. *IEEE transactions on geoscience and remote sensing*, 43(3): 502–518, 2005.

[135] Wei Li and Qian Du. Unsupervised nearest regularized subspace for anomaly detection in hyperspectral imagery. In *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*, pages 1055–1058. IEEE, 2013.

[136] Yang Xu, Zebin Wu, Jun Li, Antonio Plaza, and Zhihui Wei. Anomaly detection in hyperspectral images based on low-rank and sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 54(4):1990–2000, 2016.

[137] Wei Li and Qian Du. Collaborative representation for hyperspectral anomaly detection. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1463–1474, 2015.

[138] James A Jablonski, Trevor J Bihl, and Kenneth W Bauer. Principal component reconstruction error for hyperspectral anomaly detection. *IEEE Geoscience and Remote Sensing Letters*, 12(8):1725–1729, 2015.

[139] Robert J Johnson, Jason P Williams, and Kenneth W Bauer. Autogad: An improved ICA-based hyperspectral anomaly detection algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 51(6):3492–3503, 2013.

[140] Yushi Chen, Zhouhan Lin, Xing Zhao, Gang Wang, and Yanfeng Gu. Deep learning-based classification of hyperspectral data. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, 7(6):2094–2107, 2014.

[141] Konstantinos Makantasis, Konstantinos Karantzalos, Anastasios Doulamis, and Nikolaos Doulamis. Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4959–4962. IEEE, 2015.

[142] Wenzhi Zhao and Shihong Du. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4544–4554, 2016.

[143] Yushi Chen, Xing Zhao, and Xiuping Jia. Spectral–spatial classification of hyperspectral data based on deep belief network. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2381–2392, 2015.

[144] Wei Li, Guodong Wu, and Qian Du. Transferred deep learning for anomaly detection in hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(5):597–601, 2017.

[145] Ning Ma, Yu Peng, Shaojun Wang, and Philip HW Leong. An unsupervised deep hyperspectral anomaly detector. *Sensors*, 18(3):693, 2018.

[146] Weiying Xie, Baozhu Liu, Yunsong Li, Jie Lei, Chein-I Chang, and Gang He. Spectral adversarial feature learning for anomaly detection in hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 58(4):2352–2365, 2019.

[147] Kai Jiang, Weiying Xie, Yunsong Li, Jie Lei, Gang He, and Qian Du. Semisupervised spectral learning with generative adversarial network for hyperspectral anomaly detection. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[148] Shangzhen Song, Huixin Zhou, Yixin Yang, and Jiangluqi Song. Hyperspectral anomaly detection via convolutional neural network and low rank with density-based clustering. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(9):3637–3649, 2019.

[149] Chunhui Zhao, Weiwei Deng, Yiming Yan, and Xifeng Yao. Progressive line processing of kernel RX anomaly detection algorithm for hyperspectral imagery. *Sensors*, 17(8):1815, 2017.

[150] M. Díaz, R. Guerra, and S. López. A hardware-friendly anomaly detector for real-time applications with push-broom scanners. In *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5, 2019.

[151] Raúl Guerra, Yúbal Barrios, María Díaz, Abelardo Baez, Sebastián López, and Sarmiento Roberto. A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 1–17, 2019. ISSN 1939-1404. doi: 10.1109/JSTARS.2019.2919791.

[152] USGS digital spectral library. URL http://speclab.cr.usgs.gov/spectral-lib.html.

[153] Grupo de Inteligencia Computacional, Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU), Spain. Hyperspectral imagery synthesis (EIAs) toolbox. URL http://www.ehu.es/ccwintco/index.php/Hyperspectral_Imagery_Synthesis_tools_for_MATLAB.

[154] Jared A Herweg, John P Kerekes, Oliver Weatherbee, David Messinger, Jan van Aardt, Emmett Ientilucci, Zoran Ninkov, Jason Faulring, Nina Raqueño, and Joseph Meola. SPECTIR hyperspectral airborne Rochester experiment data collection campaign. In *SPIE Defense, Security, and Sensing*, pages 839028–839028. International Society for Optics and Photonics, 2012.

[155] Antonio Plaza, Qian Du, Yang-Lang Chang, and Roger 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.

[156] Raúl Guerra, Pablo Horstrand, Aythami Rodríguez, María Díaz, Alejandro Morales, Adán Jiménez, Sebastián López, and José F López. Optimal UAV movement control for farming area scanning using hyperspectral pushbroom sensors. In *2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6. IEEE, 2019.

[157] P. Horstrand, M. Diaz, R. Guerra, S. Lopez, and J. F. Lopez. A novel hyperspectral anomaly detection algorithm for real-time applications with push-broom sensors. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 1–11, 2019. ISSN 1939-1404. doi: 10.1109/JSTARS.2019.2919911.

[158] Ming Gu and Stanley C Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM Journal on Matrix Analysis and Applications*, 16(1):79–92, 1995.

[159] Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Ichitaro Yamazaki. The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale. *SIAM Review*, 60(4):808–865, 2018.

[160] Raúl Guerra, Ernestina Martel, Jehandad Khan, Sebastián López, Peter Athanas, and Roberto Sarmiento. On the evaluation of different high-performance computing platforms for hyperspectral imaging: An OpenCL-based approach. *IEEE Journal*

of Selected Topics in Applied Earth Observations and Remote Sensing, 10(11):4879–4897, 2017.

[161] Yang Xu, Zebin Wu, Jocelyn Chanussot, and Zhihui Wei. Joint reconstruction and anomaly detection from compressive hyperspectral images using mahalanobis distance-regularized tensor RPCA. *IEEE Transactions on Geoscience and Remote Sensing*, 56(5):2919–2930, 2018.

[162] Dirk Borghys, Ingebjørg Kåsen, Véronique Achard, and Christiaan Perneel. Hyperspectral anomaly detection: Comparative evaluation in scenes with diverse complexity. *Journal of Electrical and Computer Engineering*, 2012:5, 2012.

[163] Hongjun Su, Zhaoyue Wu, Qian Du, and Peijun Du. Hyperspectral anomaly detection using collaborative representation with outlier removal. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2018.

[164] Victor Y Pan and Zhao Q Chen. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 507–516. ACM, 1999.

[165] Tarek Elgamal and Mohamed Hefeeda. Analysis of PCA algorithms in distributed environments. *arXiv preprint arXiv:1503.05214*, 2015.

[166] Angelika Bunse-Gerstner and William B Gragg. Singular value decompositions of complex symmetric matrices. *Journal of Computational and Applied Mathematics*, 21(1):41–54, 1988.

[167] Alan Kaylor Cline and Inderjit S Dhillon. Computation of the singular value decomposition. In Leslie Hogben, editor, *Handbook of linear algebra*, chapter 45. Chapman and Hall/CRC, 2006.

[168] James W Demmel. *Applied numerical linear algebra*, volume 56. Siam, 1997.

[169] S Fu, R Chang, S Couture, M Menarini, MA Escobar, M Kuteifan, M Lubarda, D Gabay, and V Lomakin. Micromagnetics on high-performance workstation and mobile computational platforms. *Journal of Applied Physics*, 117(17):17E517, 2015.

[170] Fred Ortenberg, PS Thenkabail, JG Lyon, and A Huete. Hyperspectral sensor characteristics: airborne, spaceborne, hand-held, and truck-mounted; integration of hyperspectral data with lidar. *Hyperspectral Remote sensing of vegetation*, pages 39–68, 2011.

[171] Naval Studies Board, National Research Council, et al. *Autonomous vehicles in support of naval operations*. National Academies Press, 2005.

[172] Cristina Gómez and David R Green. Small unmanned airborne systems to support oil and gas pipeline monitoring and mapping. *Arabian Journal of Geosciences*, 10 (9):202, 2017.

[173] Didier Keymeulen, Nazeeh Aranki, Ben Hopson, Aaron Kiely, Matthew Klimesh, and Khaled Benkrid. GPU lossless hyperspectral data compression system for space applications. In *2012 IEEE Aerospace Conference*, pages 1–9. IEEE, 2012.

[174] Michael W Marcellin and David S Taubman. JPEG2000: image compression fundamentals, standards, and practice. *Kluwer International Series in Engineering and Computer Science, Secs 642*, 2002.

[175] Lena Chang, Ching-Min Cheng, and Ting-Chung Chen. An efficient adaptive KLT for multispectral image compression. In *4th IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 252–255. IEEE, 2000.

[176] Pengwei Hao and Qingyun Shi. Reversible integer KLT for progressive-to-lossless compression of multiple component images. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, volume 1, pages I–633. IEEE, 2003.

[177] Andrea Abrardo, Mauro Barni, and Enrico Magli. Low-complexity predictive lossy compression of hyperspectral and ultraspectral images. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 797–800. IEEE, 2011.

[178] B. Penna, T. Tillo, E. Magli, and G.Olmo. Progressive 3-d coding of hyperspectral images based on JPEG 2000. *IEEE Geoscience and remote sensing letters*, 3(1): 125–129, 2006.

[179] Lucana Santos, Enrico Magli, Raffaele Vitulli, José F López, and Roberto Sarmiento. Highly-parallel GPU architecture for lossy hyperspectral image compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(2):670–681, 2013.

[180] Yubal Barrios, Antonio J Sánchez, Lucana Santos, and Roberto Sarmiento. Shyloc 2.0: A versatile hardware solution for on-board data and hyperspectral image compression on future space missions. *Ieee Access*, 8:54269–54287, 2020.

[181] L. Santos, J. F. López, R. Sarmiento, and R. Vitulli. FPGA implementation of a lossy compression algorithm for hyperspectral images with a high-level synthesis tool. In *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, pages 107–114, 2013.

[182] Aaron B Kiely, Matthew Klimesh, Ian Blanes, Jonathan Ligo, Enrico Magli, Nazeeh Aranki, Michael Burl, Roberto Camarero, Michael Cheng, Sam Dolinar, et al. The new CCSDS standard for low-complexity lossless and near-lossless multispectral and hyperspectral image compression. 2018.

[183] Estanislau Auge, Josep Santalo, Ian Blanes, Joan Serra-Sagrista, Aaron Kiely, et al. Review and implementation of the emerging CCSDS recommended standard for multispectral and hyperspectral lossless image coding. In *2011 First International Conference on Data Compression, Communications and Processing*, pages 222–228. IEEE, 2011.

[184] Estanislau Augé, Jose Enrique Sánchez, Aaron B Kiely, Ian Blanes, and Joan Serra-Sagrista. Performance impact of parameter tuning on the CCSDS-123 lossless multi- and hyperspectral image compression standard. *Journal of Applied Remote Sensing*, 7(1):074594, 2013.

[185] Bruno Aiazzi, Luciano Alparone, and Stefano Baronti. Quality issues for compression of hyperspectral imagery through spectrally adaptive dpcm. In *Satellite Data Compression*, pages 115–147. Springer, 2012.

[186] Chulhee Lee, Sangwook Lee, and Jonghwa Lee. Effects of lossy compression on hyperspectral classification. In *Satellite Data Compression*, pages 269–285. Springer, 2012.

[187] Fernando Garcia-Vilchez, Jordi Muñoz-Marí, Maciel Zortea, Ian Blanes, Vicente González-Ruiz, Gustavo Camps-Valls, Antonio Plaza, and Joan Serra-Sagristà. On the impact of lossy compression on hyperspectral image classification and unmixing. *IEEE Geoscience and remote sensing letters*, 8(2):253–257, 2010.

[188] Chein-I Chang. *Hyperspectral data processing: algorithm design and analysis*. John Wiley & Sons, 2013.

[189] Michael J Ryan and John F Arnold. The lossless compression of AVIRIS images by vector quantization. *IEEE transactions on geoscience and remote sensing*, 35(3): 546–550, 1997.

[190] Shen-En Qian, Allan B Hollinger, Dan Williams, and Davinder Manak. Vector quantization using spectral index-based multiple subcodebooks for hyperspectral data compression. *IEEE Transactions on Geoscience and Remote Sensing*, 38(3): 1183–1190, 2000.

[191] Mark R Pickering and Michael J Ryan. Efficient spatial-spectral compression of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 39(7): 1536–1539, 2001.

[192] Daniel Báscones, Carlos González, and Daniel Mozos. Hyperspectral image compression using vector quantization, PCA and JPEG2000. *Remote Sensing*, 10(6): 907, 2018.

[193] Filipe Magalhães, Francisco M Araújo, Miguel Correia, Mehrdad Abolbashari, and Faramarz Farahi. High-resolution hyperspectral single-pixel imaging system based on compressive sensing. *Optical Engineering*, 51(7):071406, 2012.

[194] KS Gunasheela and HS Prasantha. Compressive sensing approach to satellite hyperspectral image compression. In *Information and Communication Technology for Intelligent Systems*, pages 495–503. Springer, 2019.

[195] Saurabh Kumar, Subhasis Chaudhuri, Biplab Banerjee, and Feroz Ali. Onboard hyperspectral image compression using compressed sensing and deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[196] Giulio Coluccia, Cinzia Lastri, Donatella Guzzi, Enrico Magli, Vanni Nardino, Lorenzo Palombi, Ivan Pippi, Valentina Raimondi, Chiara Ravazzi, Florin Garoi, et al. Optical compressive imaging technologies for space big data. *IEEE Transactions on Big Data*, 2019.

[197] Azam Karami, Soosan Beheshti, and Mehran Yazdi. Hyperspectral image compression using 3d discrete cosine transform and support vector machine learning. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 809–812. IEEE, 2012.

[198] Jin Li and Zilong Liu. Multispectral transforms using convolution neural networks for remote sensing multispectral image compression. *Remote Sensing*, 11(7):759, 2019.

[199] Consultative Committee for Space Data Systems (CCSDS). *Lossless Multispectral and Hyperspectral Image Compression, Recommended Standard CCSDS 123.0-B-1.*, . May 2012. Blue Book.

[200] Matthew A Klimesh. Low-complexity lossless compression of hyperspectral imagery via adaptive filtering. 2005.

[201] Allen Gersho. Adaptive filtering with binary reinforcement. *IEEE Transactions on Information Theory*, 30(2):191–199, 1984.

[202] D Keymeulen, D Dolman, S Shin, J Riddley, M Klimesh, A Kiely, DR Thompson, M Cheng, S Dolinar, E Liggett, et al. High performance space data acquisition clouds screening and data compression with modified COTS embedded system-on-chip instrument avionics for space-based next generation imaging spectrometers (ngis). In *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, 2018.

[203] Consultative Committee for Space Data Systems (CCSDS). *Low-complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression, Recommended Standard CCSDS 123.0-B-2.*, . February 2019. Blue Book.

[204] Majid Rabbani. JPEG2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[205] Himanshu M Parmar and PG Scholar. Comparison of dct and wavelet based image compression techniques. *International journal of engineering development and research*, 2:664–669, 2014.

[206] Zixiang Xiong, Kannan Ramchandran, Michael T Orchard, and Ya-Qin Zhang. A comparative study of dct-and wavelet-based image coding. *IEEE Transactions on circuits and systems for video technology*, 9(5):692–695, 1999.

[207] Ian Blanes and Joan Serra-Sagristà. Cost and scalability improvements to the karhunen–loêve transform for remote-sensing image coding. *IEEE Transactions on Geoscience and Remote Sensing*, 48(7):2854–2863, 2010.

[208] Qian Du, Wei Zhu, and James E Fowler. Anomaly-based hyperspectral image compression. In *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages II–974. IEEE, 2008.

[209] Ian Blanes and Joan Serra-Sagristà. Pairwise orthogonal transform for spectral image coding. *IEEE Transactions on Geoscience and Remote Sensing*, 49(3):961–972, 2010.

[210] Consultative Committee for Space Data Systems (CCSDS). *Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression, Recommended Standard CCSDS 122.1-B-1.*, . September 2017. Blue Book.

[211] Consultative Committee for Space Data Systems (CCSDS). *Blue Books: Recommended Standards.* {[Online].Available:}https://public.ccsds.org/Publications/BlueBooks.aspx, . (Accessed March 2019).

[212] Paul G Howard and Jeffrey Scott Vitter. Fast and efficient lossless image compression. In *Data Compression Conference, 1993. DCC'93.*, pages 351–360. IEEE, 1993.

[213] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[214] Chang Li, Yong Ma, Xiaoguang Mei, Fan Fan, Jun Huang, and Jiayi Ma. Sparse unmixing of hyperspectral data with noise level estimation. *Remote Sensing*, 9(11): 1166, 2017.

[215] Raúl Guerra, Yubal Barrios, María Díaz, Abelardo Baez, Sebastián López, and Roberto Sarmiento. A hardware-friendly hyperspectral lossy compressor for next-generation space-grade field programmable gate arrays. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12):4813–4828, 2019.

[216] María Díaz, Raúl Guerra, Pablo Horstrand, Ernestina Martel, Sebastián López, José F López, and Roberto Sarmiento. Real-time hyperspectral image compression onto embedded GPUs. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(8):2792–2809, 2019.

[217] Julián Caba, María Díaz, Jesús Barba, Raúl Guerra, Jose A López, et al. FPGA-based on-board hyperspectral imaging compression: Benchmarking performance and energy efficiency against GPU implementations. *Remote Sensing*, 12(22):3741, 2020.

[218] Maria Diaz, Raúl Guerra, Pablo Horstrand, Sebastián López, José F López, and Roberto Sarmiento. Towards the concurrent execution of multiple hyperspectral imaging applications by means of computationally simple operations. *Remote Sensing*, 12(8):1343, 2020.

[219] Yaokai Liu, Tianxing Wang, Lingling Ma, and Ning Wang. Spectral calibration of hyperspectral data observed from a hyperspectrometer loaded on an unmanned aerial vehicle platform. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2630–2638, 2014.

[220] Paul Geladi, Jim Burger, and Torbjörn Lestander. Hyperspectral imaging: calibration problems and solutions. *Chemometrics and intelligent laboratory systems*, 72 (2):209–217, 2004.

[221] Helge Aasen, Eija Honkavaara, Arko Lucieer, and Pablo J Zarco-Tejada. Quantitative remote sensing at ultra-high resolution with UAV spectroscopy: a review of sensor technology, measurement procedures, and data correction workflows. *Remote Sensing*, 10(7):1091, 2018.

[222] Teemu Hakala, Lauri Markelin, Eija Honkavaara, Barry Scott, Theo Theocharous, Olli Nevalainen, Roope Näsi, Juha Suomalainen, Niko Viljanen, Claire Greenwell, et al. Direct reflectance measurements from drones: Sensor absolute radiometric calibration and system tests for forest reflectance characterization. *Sensors*, 18(5): 1417, 2018.

[223] L. Zhang, B. Peng, F. Zhang, L. Wang, H. Zhang, P. Zhang, and Q. Tong. Fast real-time causal linewise progressive hyperspectral anomaly detection via cholesky decomposition. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(10):4614–4629, 2017. doi: 10.1109/JSTARS.2017.2725382.

[224] Chunhui Zhao and Xifeng Yao. Progressive line processing of global and local real-time anomaly detection in hyperspectral images. *Journal of Real-Time Image Processing*, 16(6):2289–2303, 2019.

[225] Chein-I Chang, Yulei Wang, and Shih-Yu Chen. Anomaly detection using causal sliding windows. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(7):3260–3270, 2015.

[226] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang. High-Level Synthesis for FPGAs: From Prototyping to Deployment. *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, 30(4):473–491, April 2011. ISSN 0278-0070. doi: 10.1109/TCAD.2011.2110592.

[227] Xilinx Inc. Ultrafast vivado hls methodology guide. Technical report, Xilinx Inc., 2020.

[228] Donald G. Bailey. The advantages and limitations of high level synthesis for FPGA based image processing. In *Proceedings of the 9th International Conference on Distributed Smart Cameras*, ICDSC '15, page 134–139, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336819. doi: 10.1145/2789116.2789145. URL https://doi.org/10.1145/2789116.2789145.

[229] Johannes de Fine Licht, Simon Meierhans, and Torsten Hoefler. Transformations of high-level synthesis codes for high-performance computing. *CoRR*, abs/1805.08288, 2018. URL http://arxiv.org/abs/1805.08288.

[230] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. URL https://doi.org/10.1145/1327452.1327492.

[231] John Cheng, Max Grossman, and Ty McKercher. *Professional Cuda C Programming*. John Wiley & Sons, 2014.

[232] NVIDIA Developer. CUDA Toolkit Documentation. Features and Technical Specifications - Table 14. Technical Specifications per Compute Capability. [Online] .Available:https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#features-and-technical-specifications. (Accessed March 2019).

[233] NVIDIA Corporation. *NVIDIA Jetson Linux Developer Guide 32.4.3 Release. Power Management for Jetson Nano and Jetson TX1 Devices.* Available Online: https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_nano.html, . (Accessed on 7 September 2019).

[234] NVIDIA Corporation. *NVIDIA Jetson Linux Driver Package Software Features Release 32.3. Power Management for Jetson TX2 Series Devices.* Available Online: https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3231/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_tx2_32.html, . (Accessed on 7 September 2019).

[235] NVIDIA Corporation. *NVIDIA Jetson Linux Developer Guide 32.4.3 Release. Power Management for Jetson Xavier NX and Jetson AGX Xavier Series Devices.* Available Online: https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_jetson_xavier.html, . (Accessed on 7 September 2019).

[236] José M Melián, Adán Jiménez, María Díaz, Alejandro Morales, Pablo Horstrand, Raúl Guerra, Sebastián López, and José F López. Real-time hyperspectral data transmission for UAV-based acquisition platforms. *Remote Sensing*, 13(5):850, 2021.

[237] E. Ibarrola-Ulzurrun, L. Drumetz, J. Marcello, C. Gonzalo-Martín, and J. Chanussot. Hyperspectral classification through unmixing abundance maps addressing spectral variability. *IEEE Transactions on Geoscience and Remote Sensing*, 57 (7):4775–4788, 2019. doi: 10.1109/TGRS.2019.2892903.

[238] Sicong Liu, Qian Du, Xiaohua Tong, Alim Samat, Haiyan Pan, and Xiaolong Ma. Band selection-based dimensionality reduction for change detection in multitemporal hyperspectral images. *Remote Sensing*, 9(10):1008, 2017.

[239] Qian Du and He Yang. Similarity-based unsupervised band selection for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 5(4):564–568, 2008.

[240] Ahmad W Bitar, Loong-Fah Cheong, and Jean-Philippe Ovarlez. Sparse and low-rank decomposition for automatic target detection in hyperspectral imagery. *arXiv preprint arXiv:1711.08970*, 2017.

[241] Yi Chen, Nasser M Nasrabadi, and Trac D Tran. Sparse representation for target detection in hyperspectral imagery. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):629–640, 2011.

[242] Chein-I Chang, Hsuan Ren, and Shao-Shan Chiang. Real-time processing algorithms for target detection and classification in hyperspectral imagery. *IEEE transactions on geoscience and remote sensing*, 39(4):760–768, 2001.

[243] Chein-I Chang. Real-time recursive hyperspectral sample processing for active target detection: Constrained energy minimization. In *Real-Time Recursive Hyperspectral Sample and Band Processing*, pages 123–156. Springer, 2017.

[244] C. Chang, H. Li, M. Song, C. Liu, and L. Zhang. Real-time constrained energy minimization for subpixel detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2545–2559, 2015. doi: 10.1109/JSTARS.2015.2425417.

[245] Ernestina Martel, Raul Guerra, Sebastian Lopez, and Roberto Sarmiento. A GPU-based processing chain for linearly unmixing hyperspectral images. *IEEE journal of selected topics in applied earth observations and remote sensing*, 10(3):818–834, 2016.

[246] Hyperspectral Unmixing Datasets and Ground Truths. Available Online: http://lesun.weebly.com/hyperspectral-data-set.html. Accessed: 2021-02-24.

[247] Feiyun Zhu, Ying Wang, Shiming Xiang, Bin Fan, and Chunhong Pan. Structured sparse method for hyperspectral unmixing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 88:101–118, 2014.

[248] Cheng-I Chang, Xiao-Li Zhao, Mark LG Althouse, and Jeng Jong Pan. Least squares subspace projection approach to mixed pixel classification for hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 36(3):898–912, 1998.

[249] Inmaculada García Dópido. *New techniques for hyperspectral image classification*. PhD thesis, PhD thesis, Universidad de Extremadura, 2013.

[250] Jakob Sigurdsson, Magnus O Ulfarsson, and Johannes R Sveinsson. Total variation and ℓq based hyperspectral unmixing for feature extraction and classification. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 437–440. IEEE, 2015.

[251] Anabella Medina Machín, Javier Marcello, Antonio I Hernández-Cordero, Javier Martín Abasolo, and Francisco Eugenio. Vegetation species mapping in a coastal-dune ecosystem using high resolution satellite imagery. *GIScience & Remote Sensing*, 56(2):210–232, 2019.

[252] AP Carleer and Eléonore Wolff. Urban land cover multi-level region-based classification of vhr data by selecting relevant features. *International Journal of Remote Sensing*, 27(6):1035–1051, 2006.

[253] Mathias Bochow, Birgit Heim, Theres Küster, Christian Rogaß, Inka Bartsch, Karl Segl, Sandra Reigber, and Hermann Kaufmann. On the use of airborne imaging

spectroscopy data for the automatic detection and delineation of surface water bodies. In *Remote sensing of planet earth*, pages 1–22. InTech, 2012.