CENTRO INTERNACIONAL DE ESTUDOS
DE DOUTORAMENTO E AVANZADOS
DA USC (CIEDUS)

TESIS DE DOCTORADO

EFFICIENT MULTITEMPORAL CHANGE DETECTION TECHNIQUES FOR HYPERSPECTRAL IMAGES ON GPU

Javier López Fandiño

ESCUELA DE DOCTORADO INTERNACIONAL

PROGRAMA DE DOCTORADO EN INVESTIGACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

SANTIAGO DE COMPOSTELA
2018

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

DECLARACIÓN DEL AUTOR DE LA TESIS

Efficient Multitemporal Change Detection Techniques for Hyperspectral Images on GPU

Don Javier López Fandiño

Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:

1. *La tesis abarca los resultados de la elaboración de mi trabajo.*
2. *En su caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
3. *La tesis es la versión definitiva presentada para su defensa y coincide con la versión enviada en formato electrónico.*
4. *Confirmo que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*

En Santiago de Compostela, 28 de Mayo de 2018

Fdo. Javier López Fandiño

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DE LOS DIRECTORES DE LA TESIS

Efficient Multitemporal Change Detection Techniques for Hyperspectral Images on GPU

Dra. Dora Blanco Heras

Dr. Francisco Argüello Pedreira

INFORMAN:

Que la presente tesis, corresponde con el trabajo realizado por ***Don Javier López Fandiño*** *bajo nuestra dirección, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores de ésta no incurre en las causas de abstención establecidas en la Ley 40/2015.*

En Santiago de Compostela, 28 de Mayo de 2018

Fdo. Dra. Dora Blanco Heras
Directora tesis

Fdo. Dr. Francisco Argüello Pedreira
Director tesis

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

-You're going to make a difference. A lot of times it won't be huge, it won't be visible even. But it will matter just the same.

Commissioner James Gordon

-Why do we fall? -So we can learn to pick ourselves back up.

Thomas Wayne

-It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.

John Von Neumann

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acknowledgments

First and foremost, my deepest gratitude to Dora and Francisco, my Ph.D. advisors, for their guidance during these years. This thesis would not have been possible without their supervision.

I also want to thank Prof. Jocelyn Chanussot and his group at GIPSA-lab, Grenoble, for their kind assistance during my research stay in their group. My gratitude also goes to Sicong Liu, Tongji University, and the EHU group from Universidad del Pais Vasco, for providing the datasets used to validate the proposals of this thesis.

I want to express my deep appreciation to the Departamento de Electrónica e Computación and to the Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS) for all the help and resources provided. I can only extend my thanks to my colleagues and friends from the CiTIUS, who are responsible for making these years a great experience.

Special thanks go to my family, for helping me get here, and to my partner, for always being my biggest support.

Finally, I would like to thank the institutions that provided funding for the development of this thesis: Xunta the Galicia, under a predoctoral grant. Consellería de Cultura, Educación e Ordenación Universitaria [grant numbers GRC2014/008 and ED431G/08], and Ministry of Education, Culture and Sport, Government of Spain [grant numbers TIN2013-41129-P and TIN2016-76373-P], both co-funded by the European Regional Development Fund (ERDF).

In Santiago de Compostela, May 2018

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

Acronyms

1D 1-Dimensional.

2D 2-Dimensional.

3D 3-Dimensional.

AA Average Accuracy.

AE Autoencoder.

API Application Program Interface.

AWGN Additive White Gaussian Noise.

CA Cellular Automata.

CD Change Detection.

CMG Color Morphological Gradient.

CPU Central Processing Unit.

CS Class-Specific Accuracy.

CUDA Compute Unified Device Architecture.

CVA Change Vector Analysis.

DAFE Discriminant Analysis Feature Extraction.

ECAS-II Evolutionary Cellular Automata Segmentation.

ED Euclidean Distance.

ELM Extreme Learning Machine.

EM Expectation-Maximization.

EMLS Expectation-Maximization-based Level Set.

EMP Extended Morphological Profile.

EVD Eigenvalue Decomposition.

FA False Alarm.

FE Feature Extraction.

FPGA Field Programmable Gate Arrays.

GMM Gaussian Mixture Model.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HPC High Performance Computing.

HRW Hyperspectral Raw.

HSeg Hierarchical Segmentation.

HYFM Hyperspectral Fourier-Merlin.

ICA Independent Component Analysis.

ISODATA Iterative Self-Organizing Data Analysis Technique.

k Kappa coefficient.

K-PCA Kernel Principal Component Analysis.

MA Missed Alarm.

MLFFT Multilayer Fractional Fourier Transform.

MM Mathematical Morphology.

MP Morphological Profile.

MRF Markov Random Field.

MSF Minimum Spanning Forest.

MV Majority Vote.

NDVI Normalized Difference Vegetation Index.

NPP NVIDIA Performance Primitives.

NWFE Non-parametric Weighted Feature Extraction.

OA Overall Accuracy.

OAA One-Against-All.

OAO One-Against-One.

PC Principal Component.

PCA Principal Component Analysis.

PSNR Peak Signal-to-Noise Ratio.

PV Plurality Vote.

RAM Random Access Memory.

RBF Radial Basis Function.

RCMG Robust Color Morphological Gradient.

RD-MSF Stochastic Minimum Spanning Forest.

RGB Red, Green, Blue.

RQS Really Quick Shift.

SAE Stacked Autoencoder.

SAM Spectral Angle Mapper.

SAR Satellite Aperture Radar.

SE Structuring Element.

SGD Stochastic Gradient Descent.

SID Spectral Information Divergence.

SIFT scale-Invariant Feature Transform.

SIMD Single Instruction Multiple Data.

SLFN Single hidden Layer Feed-forward neural Network.

SLI Scalable Link Interface.

SM Streaming Multiprocessor.

SURF Speeded-Up Robust Features.

SVD Single Value Decomposition.

SVM Support Vector Machine.

TCA Transfer Component Analysis.

USGS United States Geological Survey.

V-ELM Voting-based Extreme Learning Machine.

Resumen

Esta tesis tiene como principal objetivo el desarrollo de técnicas eficientes para la detección de cambios en conjuntos de datos hiperespectrales multitemporales provenientes de sensado remoto de la cobertura de la corteza terrestre. Para alcanzar este objetivo, es necesario desarrollar también técnicas eficientes para el registrado, extracción de características, segmentación y clasificación de imágenes hiperespectrales. En esta tesis se desarrollan dos líneas principales de investigación: la propuesta de esquemas espectrales-espaciales para la detección de cambios, mejorando la calidad de detección de los esquemas existentes, y la propuesta de técnicas computacionalmente eficientes para arquitecturas multinúcleo y GPUs (*Graphics Processing Units*, en inglés) con el fin último de alcanzar la ejecución de dichas técnicas en tiempo real en hardware de consumo.

Las imágenes hiperespectrales incluyen cientos de valores de reflectancia a distintas longitudes de onda, dando lugar a píxeles con múltiples componentes que se conocen como píxel-vectores [1, 2]. En función del sensor utilizado, la información contenida en un único píxel-vector puede cubrir espectros desde el infrarrojo cercano hasta el ultravioleta, incluyendo el espectro visible [3]. Por lo tanto, las imágenes hiperespectrales pueden considerarse como un cubo tridimensional de información (incluyendo las dos dimensiones espaciales y la dimensión espectral) cuyo correcto procesado requiere del uso de técnicas especialmente adaptadas para tal fin [4]. Un procesado adecuado de la información almacenada en imágenes hiperespectrales permite la identificación de materiales y objetos. Esto hace posible llevar a cabo tareas como la clasificación, segmentación o detección de objetivos en imágenes individuales, o incluso la detección de cambios entre pares de imágenes o secuencias de vídeo [5, 6]. La disponibilidad de imágenes hiperespectrales de sensado remoto está creciendo de manera exponencial en los últimos años, gracias al uso de escáneres de sensado remoto [7, 8] que pueden colocarse en satélites como el espectrómetro Hyperion [9], o en plataformas aero-

transportadas como los sensores AVIRIS [10] o ROSIS [11]. En concreto, estos sensores cubren un rango espectral desde los 0.4 µm (el límite entre la luz ultravioleta y la visible) hasta los 2.5 µm (el final del infrarrojo cercano).

La detección de cambios es la tarea que permite detectar de forma automática las diferencias significativas en conjuntos de datos multitemporales que incluyen dos o más imágenes correspondientes a la misma área geográfica en distintos instantes de tiempo [12, 13, 14]. La detección de cambios tiene varias posibles aplicaciones en entornos de sensado remoto, tales como la monitorización de la utilización de terrenos, la monitorización de catástrofes o la supervisión de la evolución de cultivos, entre otras [15]. El planeamiento urbano es el proceso encargado de gestionar la evolución de estructuras hechas por humanos y la protección del medio ambiente [16, 17]. La monitorización de catástrofes se encarga de ayudar a gestionar situaciones como los vertidos de petróleo en el mar o el control de incendios forestales [18, 19, 20]. Por último, la supervisión de la evolución de cultivos tiene como fin último la mejora de los procesos relacionados con técnicas de agricultura de precisión, con el objetivo de optimizar su eficiencia y reducir los costes [21, 22, 23].

El procesado de detección de cambios puede tratarse a nivel de píxel [24] o a nivel de objeto [25], y puede estar centrado en detección binaria [26, 27, 28] o [29, 30, 31] multiclase. Para la primera de estas divisiones, es necesario añadir técnicas de procesado espacial al procesado a nivel de píxel para considerar la información a nivel de objeto. En el caso de la segunda división, la detección de cambios binaria se centra solamente en decidir si un píxel (u objeto) ha cambiado o no entre las imágenes tomadas en consideración, mientras que la detección de cambios multiclase proporciona una clasificación de los distintos tipos de cambio que se han encontrado en el conjunto de datos multitemporal.

Otra posible categorización de las técnicas de detección de cambio está basada en el tipo de fusión de la información multitemporal, que puede llevarse a cabo a nivel de características o a nivel de decisión. Dentro de la primera aproximación, algunas técnicas se basan en el uso de operaciones algebraicas como la diferencia entre imágenes, el ratio entre imágenes, el índice de diferencia de vegetación [32] o el análisis del vector de cambio (CVA, por sus siglas en inglés) [33]. También hay técnicas basadas en métodos borrosos, asumiendo la existencia de algún solapamiento entre las clases de cambio y no cambio [34]. Las aproximaciones basadas en información contextual como los campos aleatorios de Markov (abreviado como MRF, en inglés) [35] o los conjuntos de nivel con maximización de la esperanza (EMLS, en

inglés) también son comunes [36]. Por último, algunos métodos se basan en algoritmos de detección multiescala o jerárquicos [37].

En el caso de los métodos en los cuales la fusión se hace a nivel de decisión, existen técnicas basadas en post-clasificación, que calculan los cambios como la diferencia entre las clasificaciones individuales de las dos imágenes [38, 39]. Otras técnicas realizan una clasificación multifecha directa, considerando toda la información multitemporal en conjunto [40, 41, 42].

Como se explicó anteriormente, los esquemas de detección de cambio multiclase basados en fusión de la información a nivel de decisión, necesitan técnicas de clasificación eficientes para imágenes hiperespectrales como una etapa crucial de su procesado. En el pasado, las máquinas de soporte vectorial o *Support Vector Machines* (SVMs) [43, 44, 45, 46] han sido consideradas como el algoritmo estándar para llevar a cabo las tareas de clasificación en imágenes multidimensionales de sensado remoto. Sin embargo, es necesario utilizar aproximaciones más rápidas para conseguir la ejecución de estos algoritmos en tiempo real. Las máquinas de aprendizaje extremo, o ELMs, por sus siglas en inglés [47, 48, 49, 50], han demostrado ser una alternativa apropiada a las SVM en términos de precisión [51, 52, 53, 54, 55], y resultan especialmente adecuadas para su proyección en arquitecturas de procesado paralelo debido a que están basadas en operaciones matriciales. En esta tesis se desarrolla un clasificador ELM eficiente en GPU y se aplica a imágenes de sensado remoto obteniendo tiempos de ejecución mucho más rápidos que los obtenidos mediante SVM y mejorando también las precisiones de clasificación. También se estudian diferentes alternativas de aplicación de ELM. Por ejemplo, se explora el uso de agrupamientos. Se denomina agrupamiento a un conjunto de clasificadores independientes cuyos resultados son posteriormente combinados para obtener una clasificación final conjunta. Publicaciones anteriores han demostrado que esta técnica mejora la precisión de los resultados de clasificación [50, 56].

El uso de información espacial, en conjunto con la información espectral de las imágenes hiperespectrales, permite mejorar la calidad de los resultados obtenidos en términos de precisión en diversas tareas como la eliminación de ruido [57, 58] o la clasificación [5, 59, 60, 61]. Por este motivo, diversas aproximaciones basadas en técnicas de segmentación y morfología matemática MM [62] han sido estudiadas en esta tesis. Se proponen esquemas de clasificación espectral-espacial basados en la combinación del ELM desarrollado con algoritmos de segmentación como *watershed* [63], el algoritmo RQS (del inglés, *Really Quick Shift*) [64] y *Evolutionary Cellular Automata Segmentation* (ECAS-II) [65]. Algunas técnicas de

segmentación, como por ejemplo, la transformada *watershed*, necesitan de un pre-procesado durante el cual la dimensionalidad del conjunto de datos original se reduce. Otras técnicas, como es el caso de ECAS-II operan directamente sobre la dimensionalidad completa de los datos, manteniendo toda la información espectral a costa de un procesado más costoso computacionalmente. Esta tesis presenta una proyección sobre GPU de los segmentadores producidos por el algoritmo ECAS-II, permitiendo así que sean ejecutados con una alta eficiencia computacional, lo que los convierte en una opción adecuada para ser considerada en esquemas espectrales-espaciales eficientes. Como ejemplo, se propone un esquema combinando ECAS-II con ELM (ECAS-II+ELM) que es evaluado sobre imágenes hiperespectrales. Una regularización espacial basada en vecindades se añade después de la clasificación espectral [53] o al final del procesado espectral-espacial [55] para mejorar todavía más la calidad de la clasificación. El proceso consiste en explorar iterativamente los píxeles de la imagen, actualizando la etiqueta de un píxel si más de la mitad de los píxeles vecinos comparten una etiqueta diferente.

Los perfiles morfológicos extendidos (EMP, por sus siglas en inglés), resultan particularmente útiles para detectar la presencia de objetos a diferentes escalas, en lo que a tamaño se refiere. [66, 67, 68, 69]. Los EMP funcionan mediante la aplicación de operadores morfológicos [70, 71] en las bandas de una imagen hiperespectral, usando elementos estructurales de distintos tamaños. De esta forma, se obtiene una representación multinivel de los objetos incluidos en la imagen para su posterior análisis. Los EMP se usan en esta tesis para extraer información espacial a nivel de objeto en un esquema multi-etapa para la detección de cambios multiclase en imágenes hiperespectrales multitemporales.

Debido a la gran cantidad de información que contiene las imágenes hiperespectrales, es necesario, habitualmente, aplicar un pre-procesado que permita reducir la dimensionalidad de los datos originales, tanto para reducir el coste computacional de los algoritmos como para tratar de eliminar la información redundante o innecesaria. Este procesado puede llevarse a cabo mediante técnicas de extracción de características [72, 73] como (por sus nombres en inglés) *Principal Component Analysis* (PCA) [74], *Kernel Principal Component Analysis* (K-PCA) [75], *Transfer Component Analysis* (TCA) [76] (un método diseñado especialmente para problemas de adaptación de dominio), *Independent Component Analysis* (ICA) [77, 78], *Non-parametric Weighted Feature Extraction* (NWFE) [79], mediante el uso de gradientes vectoriales como el *Robust Color Morphological Gradient* (RCMG) [63, 80], por ejemplo, o mediante el uso de alguna técnica de *deep learning* [81, 82]. La aplicación de PCA [83]

o NWFE permite obtener una proyección de los datos originales en un espacio de características de dimensionalidad menor, manteniendo la mayor parte de la información relevante. Esto resulta especialmente útil como pre-procesado para aquellas técnicas que producen un incremento de dimensionalidad de los datos de entrada, como es el caso de los EMP, ya que el pre-procesado mediante el uso de PCAs o NWFE permitirá reducir el tamaño de los datos que deben ser procesados.

Los gradientes vectoriales son especialmente útiles como pre-procesado para técnicas de segmentación tales como, por ejemplo, la transformada *watershed* o el algoritmo RQS, ya que generan imágenes de una única banda destacando aquellas áreas donde la diferencia entre los niveles de intensidad de píxeles vecinos es mayor, es decir, las áreas que con más probabilidad representan el borde entre regiones de segmentación. PCA, NWFE, y gradientes vectoriales se han aplicado como parte de distintos esquemas espectrales-espaciales propuestos en esta tesis.

Estudios recientes han mostrado que diferentes técnicas de *deep learning* resultan adecuadas para procesar la información incluida en conjuntos de datos hiperespectrales o SAR (*Single Aperture Radar*, en inglés) [81, 82]. Las redes convolucionales y los *Stacked Autoencoders* (SAEs), entre otros métodos, han sido usados en tareas de sensado remoto como la reducción de dimensionalidad y extracción de características [84, 85, 86], registrado [87], segmentación [88, 89, 90, 91], eliminación de ruido [92], clasificación [93, 94, 95, 96, 86] y detección de cambios [97, 91, 98, 97, 99, 100]. Los SAEs se usan en esta tesis como alternativa a PCA para la extracción de características en un esquema de detección de cambios multiclase, permitiendo reducir la dimensionalidad de los datos originales a la vez que se destacan las características asociadas a los cambios en la imagen multitemporal.

La mayoría de las técnicas de detección de cambios existentes se centran en el problema de detección binaria (detectar, para cada píxel en la imagen, si ha cambiado o no), o realizan un agrupamiento de las diferentes clases de cambio detectadas mediante aproximaciones multiclase que no identifican las diferentes clases de cambio existentes. Entre las aproximaciones binarias para la detección de cambios, CVA [33, 101] es una técnica que compara los espectros de pares de píxeles mediante el uso de distancia euclídea entre dos imágenes multitemporales, con el objetivo de detectar si un cambio ha sucedido. Para determinar si una variación en el espectro es relevante o no, se deben utilizar técnicas de umbralización sobre el resultado del CVA. Habitualmente, esta umbralización se lleva a cabo mediante la técnica de maximización de la esperanza (conocida como EM, por sus siglas en inglés) [102]. La

principal desventaja de esta aproximación es que requiere un proceso iterativo que no resulta la mejor opción para obtener esquemas de detección de cambios eficientes, en lo que se refiere a tiempos de ejecución. En su lugar, el método de Otsu [103] para umbralización basado en histogramas puede alcanzar resultados comparables a EM en cuanto a precisión mediante un método algorítmico mucho más rápido. La distancia SAM (*Spectral Angle Mapper*, en inglés) [104, 105] ha sido usada como una buena alternativa a la distancia Euclídea para comparar espectros en imágenes hiperespectrales en tareas como la segmentación. Es también un buen candidato para comparar los espectros en conjuntos de datos multitemporales, ya que es invariante a cambios de escala, lo que la hace insensible a cambios en las condiciones de iluminación [105, 104, 106]. En lo referente a detección de cambios multiclase, las aproximaciones clásicas se basan en la fusión de la información multitemporal y aplicación directa de técnicas de agrupamiento como EM o *K-means*. En esta tesis, se presentan esquemas de detección de cambios entre pares de imágenes, tanto binarios como multiclase, para alcanzar una aproximación efectiva y computacionalmente eficiente.

Es común la aparición de ruido en las firmas espectrales capturadas por satélites de sensado remoto [107]. Este efecto puede ser especialmente dañiño para la detección de cambios ya que la aparición de distintos ruidos en las imágenes a procesar puede dar lugar a la detección de cambios inexistentes. Por este motivo, el esquema multiclase para detección de cambios propuesto en esta tesis se evalúa bajo distintas condiciones de ruido para medir su robustez ante este efecto.

En cuanto a la eficiencia temporal de los esquemas propuestos, las aplicaciones hiperespectrales de sensado remoto tienen altos requisitos computacionales, lo que las convierte en candidatos apropiados para ser proyectados en infraestructuras de computación de alto rendimiento como clusters o dispositivos hardware especializados [108, 109, 110]. En el pasado, algunas técnicas de sensado remoto han sido ejecutadas en *Field Programmable Gate Array* (FPGA) [111, 112, 113, 114] o GPUs [108, 80, 115, 116, 117] para reducir sus tiempos de ejecución. Sin embargo, la mayoría de métodos de detección de cambios en la literatura se diseñaron sin considerar el tiempo computacional. En muchos casos, el coste computacional no es ni siquiera mencionado.

Las GPUs resultan particularmente adecuadas para el procesado de imágenes hiperespectrales debido a su arquitectura masivamente paralela, que incluye cientos de núcleos de procesamiento. Sin embargo, las GPUs son un hardware que, a pesar de encontrarse en un estado de madurez, está en constante evolución en los últimos años, lo que hace necesario

desarrollar técnicas especialmente adaptadas para aprovechar completamente su particular arquitectura. Las GPUs de consumo proporcionadas por NVIDIA han sido seleccionadas entre todas las arquitecturas de computación de altas prestaciones disponibles como plataforma para obtener implementaciones eficientes de los esquemas propuestos en esta tesis. En esta tesis, se utilizan distintas técnicas para un aprovechamiento eficiente de la arquitectura en las implementaciones CUDA (del inglés *Compute Unified Device Architecture*) de los esquemas propuestos, dando lugar a esquemas más eficientes, en cuanto a tiempo computacional, que la versión correspondiente para arquitecturas multinúcleo utilizando OpenMP.

En los últimos años, el uso de múltiples GPUs para distribuir la carga computacional, consiguiendo mayores aceleraciones, es una tendencia en auge [118, 119, 120]. Por lo tanto, es un tema también tratado en esta tesis. En particular, se presenta un esquema para detección binaria de cambios adaptado a multi-GPU.

Teniendo en cuenta el estado del arte en detección de cambios para imágenes hiperespectrales multitemporales de sensado remoto, esta tesis responde a la siguiente cuestión:

> *¿Es posible diseñar un método de detección de cambios multiclase, para imágenes hiperespectrales de sensado remoto, que mejore los resultados de la literatura en términos de precisión y también en cuanto a coste computacional, utilizando arquitecturas GPU de consumo?*

Contribuciones principales

Como consecuencia del proceso de investigación llevado a cabo para el desarrollo de esta tesis, se han alcanzado diversas contribuciones para los campos de sensado remoto y computación de altas prestaciones:

1. **Propuesta de clasificadores eficientes basados ELM para imágenes hiperespectrales de sensado remoto**

    a) Se ha desarrollado un algoritmo basado en ELM para la clasificación de imágenes hiperespectrales [121]. Este algoritmo alcanza resultados de clasificación competitivos con los alcanzados por SVM en conjuntos de datos hiperespectrales de sensado remoto, pero con un menor tiempo de ejecución. Los resultados muestran que ELM es una mejor alternativa para conseguir eficiencia en cuanto a tiempos de ejecución que clasificadores tradicionales como SVM.

b) También se explora el uso de agrupamientos para mejorar la precisión de un clasificador ELM individual, comprobando que es una aproximación adecuada para obtener resultados de clasificación más robustos [121].

c) El clasificador basado en ELM se introduce en esquemas de clasificación espectral-espacial basados en segmentación, similares al propuesto en [80] basado en segmentación mediante *watershed* y clasificación mediante SVM. Para combinar los procesados espectrales y espaciales incluidos en el esquema, se utiliza una etapa final de voto de la mayoría [122].

d) En esta tesis se presenta la primera implementación eficiente en GPU en la literatura de un clasificador basado en ELM para imágenes hiperespectrales de sensado remoto. También se presentan implementaciones en GPU para los esquemas espectrales-espaciales propuestos.

2. **Proyección en GPU de un segmentador multidimensional (ECAS-II)**

a) En esta tesis se propone también la primera proyección sobre GPU del algoritmo ECAS-II [65] para segmentación de imágenes multidimensionales sin reducción de la dimensionalidad original de las imágenes [123]. Esta proyección permite reducir considerablemente los tiempos de ejecución de la segmentación, haciendo que esta estrategia sea adecuada para incluir en esquemas espectrales-espaciales eficientes.

b) ECAS-II se combina con ELM dando lugar a un nuevo esquema espectral-espacial para la clasificación eficiente de imágenes hiperespectrales. El esquema propuesto muestra que mantener la dimensionalidad de los datos durante todo el proceso de clasificación permite mejorar los resultados de clasificación obtenidos.

3. **Propuesta de esquemas de detección de cambios a nivel de objeto eficientes**

En esta tesis se proponen diversos esquemas para detección de cambios en imágenes hiperespectrales. Los esquemas propuestos combinan detección binaria y multiclase.

a) **Detección de cambios binaria**

La propuesta para detección de cambios binaria presentada en esta tesis [124, 125] se basa en CVA [101]. Se necesita una técnica de procesado de la información espacial rápida para incluir dicha información sin incrementar el reducido tiempo computacional del CVA. Las etapas del esquema propuesto son las siguientes:

i. Se utiliza una técnica de segmentación basada en el uso de un gradiente en combinación con *watershed*, ECAS-II o RQS para combinar la información espectral mediante un proceso de promediado de regiones. Este proceso modifica las imágenes de entrada originales obteniendo unas donde todos los píxeles en la misma región espacial comparten el mismo espectro promediado.

ii. El esquema propuesto incluye el uso de la distancia SAM en lugar de la distancia Euclídea para el cálculo del CVA [104].

iii. Para realizar una umbralización que permita que permita la separación inicial de cambios y no cambios, se ha seleccionado el método de Otsu [103].

iv. El mapa de detección de cambios obtenido tras la umbralización se regulariza espacialmente mediante un proceso que tiene en cuenta la vecindad próxima de cada píxel, con el objetivo de eliminar píxeles desconectados mal clasificados del mapa de detección de cambios definitivo.

Los resultados muestran la efectividad de este esquema, mejorando las precisiones obtenidas en conjuntos de datos hiperespectrales multitemporales con distintas combinaciones de técnicas alternativas para cada etapa de procesado.

b) **Detección de cambios multiclase**

La detección de cambios multiclase requiere del uso de un clasificador para poder saber con precisión las transiciones entre clases que se producen en el conjunto de datos multitemporal. Por este motivo, el algoritmo ELM presentado anteriormente [122] se usa para llevar a cabo esta tarea de forma eficiente, siguiendo las etapas que se detallan a continuación:

i. El esquema propuesto para detección de cambios multiclase [126] comienza con la fusión de la información multitemporal. En esta tesis se consideran dos aproximaciones diferentes para llevarla a cabo. Estas son la diferencia píxel a píxel y el apilamiento de toda la información en una única imagen.

ii. A continuación, se utilizan técnicas de extracción de características como PCA, NWFE o SAEs para reducir la dimensionalidad de los datos a la vez que se destacan los cambios a nivel espectral.

iii. Se utiliza un EMP [68] para extraer la información espacial a nivel de objeto.

iv. En este punto, los píxeles que no fueron seleccionados como cambio se eliminan mediante una máscara de detección de cambio binaria, antes de llevar a cabo la clasificación de los cambios. Esto permite reducir de forma considerable la carga computacional del clasificador, obteniendo así un esquema más eficiente.

v. Por último, se utiliza ELM [127, 122] para llevar a cabo la clasificación de los píxeles seleccionados como cambio, obteniendo así el mapa de clasificación de cambios definitivo.

Los resultados obtenidos muestran que el esquema propuesto permite obtener un mapa de clasificación de cambios preciso para conjuntos de datos hiperespectrales multitemporales. Este mapa de cambios identifica las transiciones entre clases para cada cambio detectado, al contrario que un gran número de esquemas en la literatura que sólo agrupan los diferentes tipos de cambio sin llegar a identificarlos.

Abstract

The main goal of this thesis is the development of efficient techniques for Change Detection (CD) between multitemporal remote sensing hyperspectral datasets for land cover applications. In order to achieve this goal, the development of efficient registration, Feature Extraction (FE), segmentation and classification techniques for hyperspectral images is also required.

Hyperspectral images contain hundreds of reflectance values at different wavelengths for each pixel, resulting in multidimensional pixels known as pixel-vectors. Hyperspectral images can be considered as a 3-Dimensional (the two spatial dimensions and the spectral one) cube of information requiring specially adapted techniques in order to be correctly processed.

Detecting regions of change in multiple hyperspectral images of the same scene taken at different times is of widespread interest for a large number of applications. For remote sensing, in particular, a very common application is land-cover analysis, which includes applications such as land use monitoring, catastrophe monitoring, or crop evolution supervision, among others. CD can be tackled at pixel or object level, and it can be focused on binary or multiclass CD. This thesis introduces CD schemes at object level for both binary and multiclass scenarios.

The multiclass CD scheme proposed is based on supervised direct multidate classification. Therefore, efficient classification techniques for hyperspectral images are needed. An efficient Extreme Learning Machine (ELM) classifier for hyperspectral images is introduced in this thesis, as an alternative for the traditional Support Vector Machine (SVM).

Both spatial and spectral information from the images is considered, to improve the quality of the proposed schemes for classification and CD. This is done through efficient segmentation techniques, such as watershed transform, Really Quick Shift (RQS) or Evolutionary Cellular

Automata Segmentation (ECAS-II). Extended Morphological Profiles (EMPs) are also used to extract object-based spatial information.

Different FE methods are applied in the spectral-spatial schemes proposed herein to reduce the dimensionality of the original data while maintaining the relevant information. Principal Component Analysis (PCA), Non-parametric Weighted Feature Extraction (NWFE), Stacked Autoencoders (SAEs), and vectorial gradients, such as Robust Color Morphological Gradient (RCMG), are the FE methods considered in this thesis.

The high dimensionality of the hyperspectral images makes the development of computationally efficient processing schemes critical. Therefore, these tasks are good candidates to be projected onto High Performance Computing (HPC) infrastructures. Graphics Processing Units (GPUs) are particularly adequate for the processing of hyperspectral images due to their massively parallel architecture. Several techniques for efficient exploitation of the GPU architecture are used in the implementations of the schemes proposed in this thesis, producing schemes that are far more time-efficient than the corresponding OpenMP schemes for multicore architectures.

Several experiments were carried out in different hyperspectral and multispectral datasets allowing us to conclude that the proposed schemes improve the accuracy of current state-of-the-art algorithms for remote sensing hyperspectral image processing. Furthermore, the projection of the schemes proposed in this thesis onto GPU allows their execution in real-time scenarios.

Keywords: change detection, remote sensing, hyperspectral imaging, graphics processing unit

Thesis overview

The main goal of this thesis is the development of efficient techniques for Change Detection (CD) between multitemporal remote sensing hyperspectral datasets for land cover applications. In order to achieve this goal, the development of efficient registration, Feature Extraction (FE), segmentation and classification techniques for hyperspectral images is also required. Two main research lines are involved: the proposal of spectral-spatial CD schemes that increase the detection accuracy of the existing ones, and the proposal of computationally efficient techniques for multicore and Graphics Processing Unit (GPU) architectures with the ultimate goal of achieving real-time execution in commodity hardware.

Hyperspectral images include hundreds of reflectance values at different wavelengths resulting in pixels with multiple components, known as pixel-vectors [1, 2]. Depending on the sensor, the information comprised in a single pixel-vector covers spectra from the near infrared to the ultraviolet, including the visible spectrum [3]. Hyperspectral images can be considered as a 3-Dimensional (the two spatial dimensions and the spectral one) cube of information requiring techniques specially adapted in order to be correctly processed [4]. An adequate processing of the information contained in the hyperspectral images allows us to distinguish between different physical materials and objects. This makes it possible to perform tasks as the classification, segmentation or target detection in a single image or even the CD between corresponding pairs of images or video sequences [5, 6]. The availability of remote sensing hyperspectral images is increasing exponentially in the recent years, thanks to the use of multispectral scanners for remote sensing purposes [7, 8] that can be placed in satellites, such as the Hyperion spectometer [9], or in airborne platforms as the AVIRIS [10] or ROSIS [11] sensors. In particular, these sensors cover a spectral range from 0.4 μm (the limit between ultraviolet and visible light) to 2.5 μm (the end of the near-infrared).

CD is the task that allows the automatic recognition of significant differences between multitemporal datasets including two or more images over the same geographical area at different time-frames [12, 13, 14]. The CD processing has different possible applications in remote sensing scenarios such as land use monitoring, catastrophe monitoring, or crop evolution supervision, among others [15]. Urban planning is a process related with the management of the evolution of human made infrastructures and the protection of the environment [16, 17]. Catastrophe monitoring aims to help dealing with situations such as oil slicks in the sea or forest fires control [18, 19, 20]. Finally, crop evolution supervision is devoted to improve the processes involved in precision farming techniques, aiming to optimize the efficiency, thus reducing costs [21, 22, 23].

CD processing can be tackled at pixel [24] or at object [25] level and can be focused on binary [26, 27, 28] or multiclass [29, 30, 31] CD. For the former division, spatial processing techniques must be added to the pixel level processing to consider the information at object level. Regarding the later division, a binary CD processing only aims to designate whether a pixel (or object) has changed or not between the considered images, whereas a multiclass CD processing performs a classification of the different types of changes that can be found in the multitemporal dataset.

Another categorization is based on the type of fusion of the multitemporal data, which can be performed at feature or at decision level [12]. In the former approach, some techniques are based on algebra operations, such as image differencing, image ratioing, vegetation index differencing [32], or Change Vector Analysis (CVA) [33]. Other techniques are based on fuzzy methods and assume some overlapping between the changed/non-changed classes [34]. Context-based approaches, such as Markov Random Fields (MRFs) [35] or Expectation-Maximization-based Level Set (EMLS), are also common [36]. Finally, some alternative methods are based on multiscale or hierarchical detection algorithms [37].

In the case of methods in which the fusion is made at decision level, there are techniques based on post-classification that compute the changes as the differences in the separate classification of the two images [38, 39]. Other techniques perform a direct multidate classification considering all the multitemporal information together [40, 41, 42].

As explained above, for multiclass CD schemes based on fusion at decision level, efficient classification techniques for hyperspectral images are needed as a crucial stage. In the past, Support Vector Machines (SVMs) [43, 44, 45, 46] have been considered as a standard algorithm for performing the classification task in remote sensing multidimensional images.

Nevertheless, faster approaches are needed in order to achieve real-time execution. Extreme Learning Machines (ELMs) [47, 48, 49, 50] have proven to be a good alternative to SVMs in terms of accuracy [51, 52, 53, 54, 55] and are particularly adequate to be projected on parallel architectures as they rely on matrix operations. An efficient GPU ELM-based classifier is developed in this thesis and applied to remote sensing images achieving execution times much faster than SVM and better classification accuracies. Different possibilities of applying ELM are also studied. For instance, the use of ensembles is explored. An ensemble is a group of independent classifiers whose results are later combined in order to obtain a final joint classification. As it has been proven in other works, this technique improves the accuracy of the classification results [50, 56].

The use of the spatial information together with the spectral information of the hyperspectral images increases the quality of the results in terms of accuracy for different tasks such as denoising [57, 58] or classification [5, 59, 60, 61]. For this reason, several approaches based on segmentation techniques and Mathematical Morphology (MM) [62] have been studied in this thesis. Spectral-spatial classification schemes based on the combination of the developed ELM with segmentation algorithms as watershed [63], Really Quick Shift (RQS) algorithm [64], and Evolutionary Cellular Automata Segmentation (ECAS-II) [65] are proposed. Some segmentation techniques, such as the watershed transform, need a pre-processing step where the dimensionality of the original data is reduced in order to be applied. Other techniques, as it is the case of ECAS-II, operate over the full dimensionality of the data, maintaining all the spectral information at the cost of a more computationally costly processing. This thesis presents a projection onto GPU of the segmenters produced by the ECAS-II algorithm, allowing them to be executed with high computational efficiency, which makes them a suitable option to be considered in real world spectral-spatial schemes. By way of example, a scheme combining ECAS-II with ELM (ECAS-II+ELM) is proposed and evaluated over hyperspectral images. Moreover, a spatial regularization based on the neighborhood can be added after the spectral classification [53] or at the end of the spectral-spatial processing [55] to further improve the quality of the classification. The process consists in iterative exploring the pixels of the image updating the label of a pixel if more than half of the neighbors share a different label.

Extended Morphological Profiles (EMPs) have proven to be particularly useful for detecting the presence of objects at different size levels [66, 67, 68, 69]. EMPs work by applying morphological operators [70, 71] with different Structuring Elements (SEs) over the bands of

a hyperspectral image. In this way, a multilevel representation of the objects present in the image is obtained for further analysis. EMPs are used in this thesis to extract the object-based spatial information in a multi-stage scheme proposed to perform multiclass CD of multitemporal hyperspectral images.

As stated previously, due to the high amount of information contained by a hyperspectral image, a pre-processing for dimensionality reduction of the original data is usually necessary. This can be tackled through FE techniques [72, 73], such as Principal Component Analysis (PCA) [74], Kernel Principal Component Analysis (K-PCA) [75], Transfer Component Analysis (TCA) [76] (an FE method specifically designed to domain adaptation problems), Independent Component Analysis (ICA) [77, 78], Non-parametric Weighted Feature Extraction (NWFE) [79], by means of vectorial gradients as the Robust Color Morphological Gradient (RCMG) [63, 80], for example, or by using different deep learning techniques [81, 82]. The application of PCA [83] or NWFE allows us to obtain a projection of the original data to a lower dimensional space of features maintaining all the relevant information. This is particularly useful as a pre-processing for those techniques that will increment the dimensionality of the input data, as is the case of EMPs, as the pre-processing by means of PCAs or NWFE will enable us to reduce the size of data to be processed.

The vectorial gradients are particularly helpful as a pre-processing for some segmentation techniques, such as, for instance, the watershed transform or the RQS algorithm, as they generate single-band images highlighting the areas where the difference between the intensity level of neighborhood pixels is bigger; i.e., the areas that are more likely to be the border between segmentation regions. PCA, NWFE, and vectorial gradients have been applied as part of different spectral-spatial schemes proposed in this thesis.

Recent works have studied the suitability of different deep learning techniques to deal with the information contained in hyperspectral or Satellite Aperture Radar (SAR) datasets [81, 82]. Deep convolutional networks and Stacked Autoencoders (SAEs), among others, have been used in remote sensing tasks, such as dimensionality reduction and FE [84, 85, 86], registration [87], segmentation [88, 89, 90, 91], image denoising [92], classification [93, 94, 95, 96, 86] and change detection [97, 91, 98, 97, 99, 100]. In this thesis, SAEs are used as an alternative to PCA to perform FE in a multiclass CD scheme, allowing us to reduce the dimensionality of the original data while enhancing the features associated to changes in the multitemporal image.

Most of the existent CD techniques focus on the binary CD problem (detection of changed or non-changed status for each pixel) or merely perform a clustering of the different classes of changes detected in a multiclass approximation but without identifying the classes of changes. Among the binary CD approximations, CVA [33, 101] is a technique that compares the spectra of corresponding pairs of pixels through the use of the Euclidean Distance (ED) in two multi-temporal images in order to detect whether a change has occurred. To determine if a variation in the spectra is relevant or irrelevant, thresholding techniques must be applied to the CVA result. This thresholding is commonly tackled by the Expectation-Maximization (EM) technique [102]. The main disadvantage of this approach is that it requires an iterative process, which is not the best option to achieve time-efficient CD schemes. Instead, the Otsu method [103] for thresholding based on histograms can achieve discrimination results comparable to the EM ones in terms of accuracy through a much faster algorithmic method. The Spectral Angle Mapper (SAM) distance [104, 105] has been used as a good alternative to ED for the comparison of spectra in hyperspectral images in tasks such as segmentation. It is a good candidate for comparing the spectra in multitemporal datasets as it is invariant to scale changes, rendering it insensitive to changes in the illumination conditions [105, 104, 106]. Regarding multiclass CD, the classical approaches rely on the fusion of the multitemporal information and the direct application of clustering techniques such as EM or K-means. In this thesis, both a binary and a multiclass CD scheme are proposed in order to achieve an effective and computationally efficient approach for CD.

The presence of noise in the spectral signatures captured by the remote sensing sensors is common [107]. This effect can be particularly harmful in CD scenarios, as the appearance of different noises in the images to be processed for the CD can result in the detection of non-existent changes. For this reason, the multiclass CD scheme proposed in this thesis is evaluated under different noisy conditions in order to measure its robustness against noise.

Regarding the time-efficiency of the proposed schemes, remote sensing hyperspectral applications are computationally demanding, making them good candidates to be projected onto High Performance Computing (HPC) infrastructures, such as clusters or specialized hardware devices [108, 109, 110]. In the past, some remote sensing techniques were executed in Field Programmable Gate Arrays (FPGA) [111, 112, 113, 114] or GPUs [108, 80, 115, 116, 117] to reduce their execution times. Nevertheless, a vast majority of CD methods in the literature are designed without considering the computational cost. In most of the cases the computational cost of the proposed methods is not even mentioned.

GPUs are particularly adequate for the processing of hyperspectral images due to their massively parallel architecture, including hundreds of cores. Nevertheless, GPUs are a hardware that, despite being mature, has been constantly evolving in recent years, requiring the development of techniques specially adapted to fully exploit their particular architecture. Commodity NVIDIA GPUs have been selected among all the HPC available architectures as the target architecture to obtain efficient implementations of the schemes proposed in this thesis. Several techniques for a more efficient exploitation of the architecture are used in the Compute Unified Device Architecture (CUDA) implementations of the schemes proposed, producing schemes far more time-efficient than the corresponding OpenMP schemes for multicore architectures.

In recent years, the use of multiple GPUs for partitioning the computational load and consequently achieving further speedups has also been a trend [118, 119, 120]. Therefore, it is a topic also explored in this thesis. In particular, a scheme for binary CD is proposed.

Given the state of the art in CD for multitemporal hyperspectral remote sensing images, this thesis answers the following question:

> *Is it possible to design a multiclass CD method for remote sensing hyperspectral images that improves the results in the literature in terms of accuracy and also in computational cost on commodity GPU architectures?*

Main contributions

Several contributions to the remote sensing and HPC fields have been achieved as a consequence of the research process carried out for the development of this thesis:

1. **Proposal of efficient ELM-based classifiers for remote sensing hyperspectral images**

    a) An ELM-based algorithm for the classification of hyperspectral images is developed [121]. This algorithm achieves classification results competitive with those achieved by SVM in remote sensing hyperspectral datasets, but with lower execution times. The results show that the use of ELM is a better alternative achieving time-efficiency than other traditionally used classifiers, such as SVM.

b) The use of ensembles to outperform the accuracy of a single ELM classifier is also explored and proven to be a suitable approach for obtaining robustness in the classification results [121].

c) The ELM-based classifier is introduced in spectral-spatial classification schemes based on segmentation, such as the one proposed in [80] based on watershed segmentation and SVM classification. A final Majority Vote (MV) step is used to combine the spectral and spatial processing involved in the scheme [122].

d) The first efficient GPU implementation in the literature of an ELM-based classifier for remote sensing hyperspectral images is presented in this thesis. GPU implementations are also developed for the spectral-spatial schemes proposed.

2. **GPU projection of a multidimensional segmenter (ECAS-II)**

a) The first GPU projection of the ECAS-II [65] algorithm for segmentation of multidimensional images without reduction of the dimensionality of the images is presented herein [123]. This projection allows us to considerably reduce the execution time of the segmentation, making this strategy eligible to be included in efficient spectral-spatial schemes.

b) ECAS-II is combined with ELM in a new spectral-spatial scheme for the efficient classification of hyperspectral images. The proposed scheme shows that maintaining the data dimensionality during the whole classification process improves the classification results.

3. **Proposal of efficient object-based change detection schemes**

Different schemes for CD in hyperspectral images are proposed herein. The proposed schemes combine binary and multiclass CD.

a) **Binary change detection**

The binary approach presented in this thesis for CD [124, 125] is based on CVA [101]. In order to include the spatial information in this scheme, a fast technique is needed to avoid increasing the low computation time of the CVA. The stages of the scheme are the following:

i. A segmentation technique based on the use of a gradient in combination with a segmentation stage by watershed, ECAS-II, or RQS is used and combined

with the spectral information through the use of an averaging process. This process modifies the original input images, obtaining new ones where all the pixels in the same spatial region share the same averaged spectra.

ii. The proposed scheme includes the use of the SAM distance instead of the ED in the computation of the CVA [104].

iii. The Otsu method is selected as thresholding technique, aiming for the efficient computation of the scheme [103].

iv. The CD map obtained is spatially regularized regarding the close neighborhood values in order to remove disconnected misclassified pixels in the final change detection map.

The results show the effectiveness of this scheme, improving the accuracies obtained in multitemporal hyperspectral datasets with different combinations of alternative techniques for each stage.

b) **Multiclass change detection**

A multiclass CD requires the use of a classifier in order to verify not only whether there is a change, but also to ascertain the particular transitions in the multitemporal dataset. For this reason, the previously presented ELM algorithm [122] is used to perform this task efficiently, following the steps detailed below:

i. The proposed scheme for multiclass CD [126] begins with the fusion of the multitemporal data. Two different approaches for combining the multitemporal information are used in this thesis. The pixel by pixel difference and the stack of all the information in a single image.

ii. Then, FE techniques, as PCA, NWFE, or SAEs, are applied to reduce the dimensionality of these data while highlighting the spectral changes.

iii. The spatial information at object level is then extracted through the use of an EMP [68].

iv. At this point, the pixels that have not been selected as changes in the binary processing are removed by a binary CD mask before performing the classification. This considerably reduces the computational workload of the classifier thus achieving a more efficient scheme.

v. Finally, ELM [127, 122] performs the classification of the pixels selected as changes, obtaining the final CD classification map.

The achieved results show that the proposed scheme allows us to efficiently obtain an accurate change classification map of multitemporal hyperspectral datasets. This change map identifies the transitions between classes for each detected change, as opposed to a high number of the schemes in the literature, which only cluster the different types of change detected without identifying them.

Publications

The publications listed below have been accomplished as a result of the research carried out for the development of this thesis.

International journals

[125] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Mauro Dalla Mura. GPU framework for change detection in multitemporal hyperspectral images. *International Journal of Parallel Programming (IJPP)*, pages 1–21, 2017. DOI: 10.1007/s10766-017-0547-5.

[123] Javier López-Fandiño, Blanca Priego, Dora B. Heras, Francisco Argüello, and Richard J. Duro. GPU projection of ECAS-II segmenter for hyperspectral images based on cellular automata. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):20–28, 2017. DOI:10.1109/JSTARS.2016.2588530.

[122] Javier López-Fandiño, Pablo Quesada-Barriuso, Dora B. Heras, and Francisco Argüello. Efficient ELM-based techniques for the classification of hyperspectral remote sensing images on commodity GPUs. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, 8(6):2884 – 2893, 2015. DOI:10.1109/JSTARS.2014.2384133.

International conferences

[128] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Efficient GPU multiclass change detection for multidimensional images in the presence of noise. In *SPIE Remote Sensing*. International Society for Optics and Photonics, 2018. Accepted.

[129] Javier López-Fandiño, Alberto S. Garea, Dora B. Heras, and Francisco Argüello. Stacked autoencoders for multiclass change detection in hyperspectral images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2018 IEEE International*. IEEE, 2018. Accepted.

[126] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Richard J. Duro. CUDA

multiclass change detection for remote sensing hyperspectral images using extended morphological profiles. In *2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, pages 404 – 409. IEEE, 2017. DOI: 10.1109/IDAACS.2017.8095113.

[124] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Mauro Dalla Mura. GPU framework for change detection in multitemporal hyperspectral images. In *10th International Symposium on High-Level Parallel Programming and Applications (HLPP17)*, pages 115 – 132, 2017.

[130] Blanca Priego, Richard J. Duro, Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Evolutionary cellular automata based approach to high-dimensional image segmentation for GPU projection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1593–1600. IEEE, 2016. DOI: 10.1109/IJCNN.2016.7727388.

[121] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Efficient classification of hyperspectral images on commodity GPUs using ELM-based techniques. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 1, 2014. ISBN:1-60132-282-8.

National conferences

[131] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Clasificación supervisada de imágenes de sensado remoto en GPU. In *XXV Jornadas de Paralelismo*, pages 319–325, 2014. ISBN: 978-84-697-0329-3.

Thesis organization

The remaining of this thesis is organized in seven chapters. Chapter 1 is devoted to the introduction of the hyperspectral imagery and the main processing techniques concerning this kind of images. There are several processing tasks related to the hyperspectral imagery, including FE, segmentation, MM, classification, registration, and CD. This last one is the main topic of this thesis but other processing tasks were also studied as parts of the CD schemes proposed.

Chapter 2 deals with the efficient computation of hyperspectral images on GPU. Two different parallel programming models are introduced to this end: OpenMP and CUDA. OpenMP is a programming model designed to work in multicore architectures while CUDA is a model

to efficiently perform general purpose computations on NVIDIA GPUs. The different techniques for achieving efficient GPU implementations used in this thesis are also explained in this chapter. Finally, the experimental setup used during the development of this thesis is introduced in detail, including the hardware used, the different performance metrics considered to evaluate the processing techniques developed, and the datasets considered for the experiments.

Chapter 3 presents the GPU implementation of the ELM classification algorithm for hyperspectral images. Different spectral-spatial classification schemes on GPU based on the use of ELM are also introduced. In particular, a scheme combines a watershed segmentation over a gradient of the original image with the ELM classification by means of a MV processing. The other scheme replaces the watershed segmentation processing by RQS-based segmentation. The experimental results achieved by applying all the introduced schemes are also discussed.

A GPU implementation of the segmenters produced by the ECAS-II algorithm is presented in Chapter 4. ECAS-II is an algorithm based on Cellular Automata (CA) that considers all the spectral information of the hyperspectral dataset during the segmentation process, allowing to obtain more accurate segmentation maps than other alternatives that perform dimensionality reduction in some steps of the algorithm. A spectral-spatial scheme combining the ECAS-II with ELM is also introduced, including accuracy and execution time results of experiments applied to hyperspectral datasets.

Chapter 5 introduces an efficient GPU scheme for the binary CD on multitemporal hyperspectral datasets. The scheme targets the object level CD. It considers regions of the image by using a region averaging process based on segmentation of the dataset. The multitemporal information is then merged by using distance computation techniques, and the final binary CD map is obtained through the application of thresholding. The GPU implementation of the techniques proposed for each stage of the algorithm are detailed and the scheme is evaluated over different multitemporal hyperspectral datasets.

Chapter 6 is devoted to the introduction of a novel multiclass CD scheme that combines binary and multiclass CD techniques in order to efficiently achieve an accurate *from-to* change classification map. The scheme is based on direct multidate classification performed by ELM. The multitemporal information of the dataset is merged and the change features are then extracted by FE techniques such as PCA, or SAE. The spatial information is taken into account by computing an EMP. The binary and multiclass CD are combined by filtering the result of

the multiclass branch of the scheme with a mask obtained from the binary CD branch, before performing the final change classification. The techniques needed for the GPU projection of the complete scheme are introduced. The scheme is evaluated in multitemporal hyperspectral and multispectral datasets and its tolerance to noisy conditions is evaluated considering two different types of noise.

The last chapter of this thesis retrieves the main contributions, presents the conclusions and proposes some possible future research lines.

CHAPTER 1

HYPERSPECTRAL IMAGE PROCESSING FUNDAMENTALS

This first chapter is devoted to the introduction of the fundamental concepts and techniques used in the schemes proposed in this thesis. First, the main characteristics of the hyperspectral images are detailed. Images of this kind are the target for the different schemes developed during this thesis. Then, the main processing tasks associated to hyperspectral images are outlined.

The central sections of this chapter introduce the fundamental concepts related to the techniques included in the proposed schemes for Change Detection (CD) of hyperspectral images. Feature Extraction (FE), segmentation, mathematical morphology, supervised classification, similarity measures, registration, and CD are the topics reviewed in these sections, with the application of the techniques to schemes devoted to the CD in hyperspectral datasets being the final goal of this thesis.

The last section of this chapter introduces *HypeRvieW* [132], a desktop tool developed by the research group for the analysis and processing of hyperspectral images, which provides easy access to some of the schemes proposed in this thesis.

1.1 Hyperspectral images

Hyperspectral images are datasets comprising reflectance values over a wide wavelength range for each pixel. In this kind of image, each pixel is characterized as a vector of val-

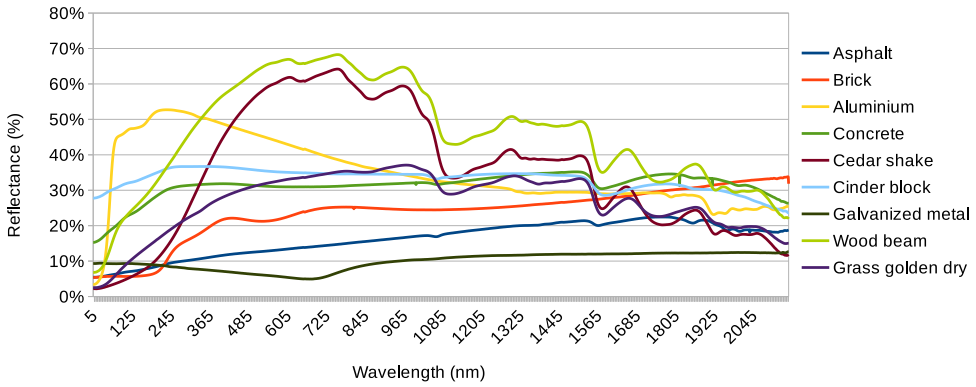


Figure 1.1: Spectral signatures of different materials provided by the United States Geological Survey [133] spectral library *splib06a* [134].

ues for the corresponding wavelength, and therefore, is called *pixel-vector* [2]. For the case of the hyperspectral images, these pixel-vectors are composed of hundreds of values, while the term multispectral image is used when the pixel-vector contains from three to ten values [3]. The availability of images of this kind represents an improvement over the panchromatic (single grayscale band) or Red, Green, Blue (RGB) images (composition of three channels that creates the illusion of color) regarding the amount of information that can be extracted from the data. The values contained in each pixel-vector of a hyperspectral image provide a pattern known as *spectral signature*. Each material has a particular spectral signature, allowing us to perform different processings to distinguish among them. Figure 1.1 shows the spectral signatures of different kind of materials according to the data obtained from the United States Geological Survey [133] spectral library *splib06a* [134].

Hyperspectral images are captured by optical sensors, usually placed in satellites, airplanes or drones, allowing us to obtain information of the Earth's surface in both the spatial and spectral domains. Images of this kind are called remote sensing images. The spatial domain represents the physical space captured by the sensor, whereas the spectral domain represents the electromagnetic wavelength range covered. In this way, the hyperspectral images can be seen as 3-Dimensional (3D) cubes that can be studied in both the spectral and spatial domains in order to extract relevant knowledge. This information can be used, for instance, to obtain a classification of the different materials present in the image or to obtain a spatial segmentation of the different regions captured.

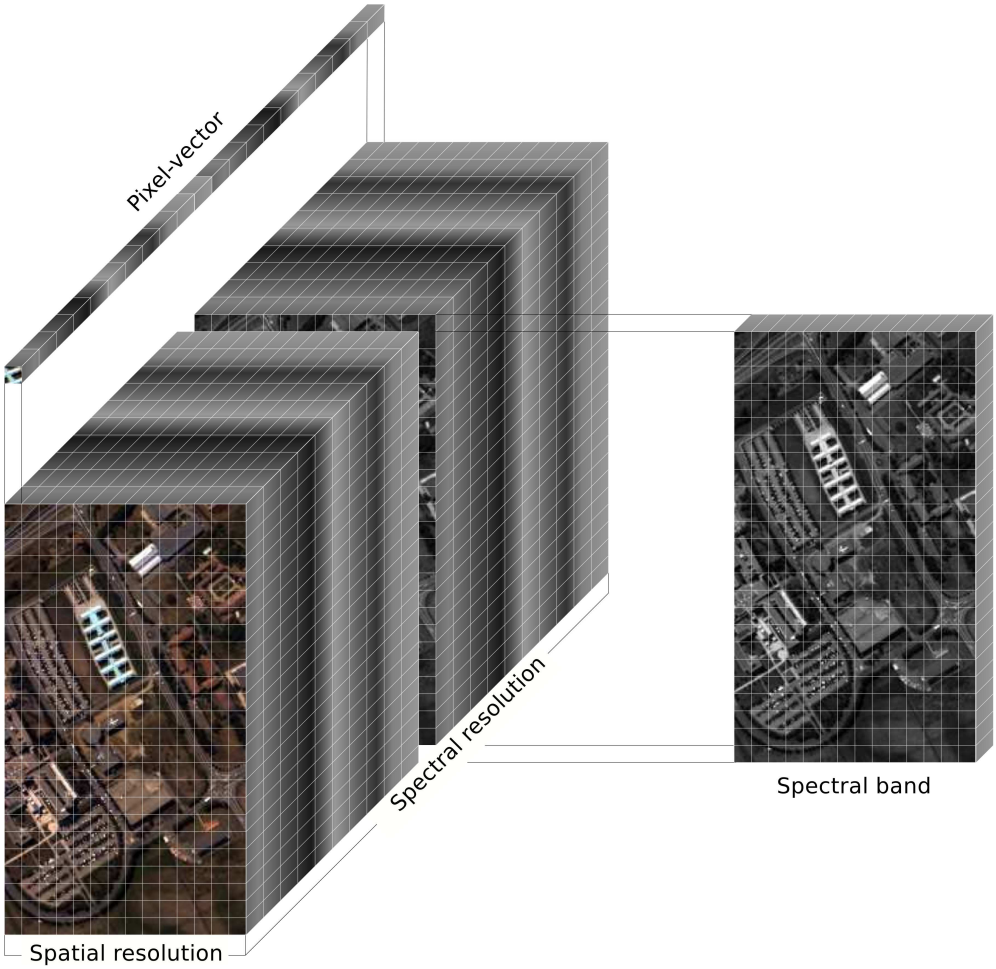


Figure 1.2: Hyperspectral image data cube.

Nowadays, there exist numerous satellite and airborne hyperspectral sensors, such as Hyperion (EO-1), AVIRIS, or ROSIS-03. Each sensor has particular capabilities regarding the spectral resolution (spectral distance between two contiguous wavelengths), spectral range (the wavelengths comprised), number of acquired bands (directly dependent on the two previous parameters), or pixel size (number of meters of Earth's surface covered by a single pixel). The spectral range acquired for these sensors usually contains not only the spectrum related

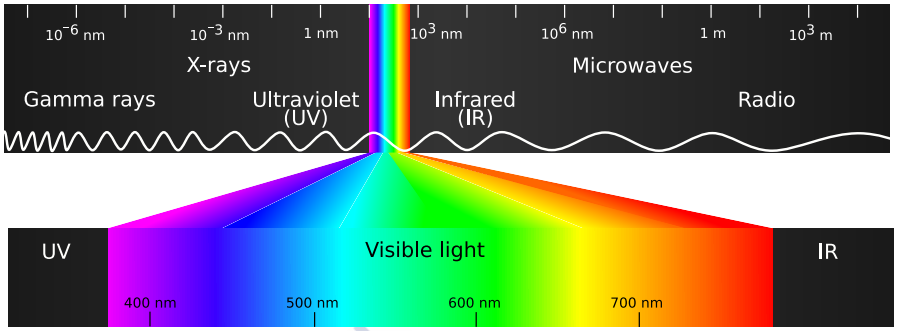


Figure 1.3: Wavelength comprised by hyperspectral images [135].

to the frequencies visible for the human eye but also part of the infrared or ultraviolet frequencies. For this reason, hyperspectral images allow us to extract more knowledge than could be extracted from lower dimensionality images.

The term *multitemporal dataset* is used to refer to groups of two or more images corresponding to different time-frames over the same physical space. Each image may correspond to a different time of day, to different seasons or even to different years.

1.1.1 Hyperspectral image processing

The information contained in the hyperspectral datasets needs to be processed for different tasks. Some of the possible processing stages required by the tasks are the following [136, 5, 3]:

- *Registration [137, 138].* The registration process consists of the estimation of geometric transformations needed to match at pixel level two or more images acquired over the same area at different times. It is an indispensable pre-processing stage in order to perform subsequent computations over multitemporal datasets.

- *Feature extraction [139, 86, 72].* This process aims to transform the hyperspectral cube into a reduced version, containing as much of the relevant original information as possible. It is a process that usually allows the dimensionality of the data to be reduced before performing a classification task, in order to avoid the problem known as *the course of dimensionality* [140].

- *Image encoding and compression [141, 142].* The size of the hyperspectral datasets can be huge. For this reason, the adequate encoding and compression of the data is relevant in order to achieve the efficient storage and transmission thereof.

- *Target and anomaly detection [143, 144, 114].* This process aims to detect pixel-vectors in an image that does not match with the general pattern of the remain pixel-vectors in the image. These pixel-vectors could correspond to a specific target or, on the contrary, to an unknown target, in which case the processing is known as anomaly detection.

- *Image segmentation [145, 146, 55].* The partitioning of the image in discrete non-overlapping regions. Each region must comply with a specific criterion of homogeneity, such as the similarity of the spectral signatures of the pixel-vectors contained or the intensity of texture. This process helps in the subsequent detection of the different objects present in an image, and is usually used as a spatial branch in spectral-spatial schemes.

- *Classification [45, 67, 46, 61, 147, 68].* This is the process devoted to the assignation of a label to each pixel in the image. In this way, the different classes or materials detected in the image are identified. There exist both unsupervised and supervised classification methods. Usually, a supervised or semi-supervised method is required to label all the pixels in the image according to the desired criteria. The supervised classification requires an initial stage called *training* in which the classifier algorithm will learn the characteristics of the different labels that must be used to perform the classification of all the pixels in the image.

- *Domain adaptation [148, 149].* Domain adaptation, also known as transfer learning, is a process that consists in learning data from a source domain, exploiting the available knowledge, and then extrapolating what was learned to a different target domain. The motivation is that it is not possible to perform the learning task on the target domain due to the lack of available prior knowledge on it.

- *Spectral unmixing [118, 150, 116].* Owing to the spatial resolution of the sensors used to acquire the data, is usual to face images where each pixel contains mixed spectral signatures of several materials. Spectral unmixing is the technique dedicated to find the pure spectral signatures, known as *endmembers*, of each pixel on the images along with the abundance of each endmember in the raw pixel acquired by the sensor.

- *Multitemporal CD [151, 152, 39, 153, 42, 36].* This process aims to detect and/or classify the significant changes existing between a set of multitemporal images covering the same space at different time-frames. This process may aim the discovery of changes at the pixel or at the object level, regarding the scope where it is being applied.

Most of the stages introduced before are addressed in this thesis. In particular, efficient schemes for solving the FE and dimensionality reduction, image segmentation and classification, and multitemporal CD problems are treated. The following sections are devoted to a detailed explanation of the fundamentals required for the aplication of the techniques developed in this thesis.

1.2 Similarity measures

Several remote sensing processes require the comparison between the spectrum represented in two different pixel-vectors, whether they came from the same image or from different images. There exist different approaches in order to compare the similarity between two pixel-vectors i and j [104, 154].

The straight forward approximation is to compare the spectrum of the two pixels through the classical Euclidean Distance (ED), computed as

$$Euclidean_{i,j} = \sqrt{\sum_{b=1}^{B} (j_b - i_b)^2}, \tag{1.1}$$

where B is the number of spectral bands.

Another possible measure is the Chi-squared distance, whose name is a derivation of the Pearson's chi squared test statistic. The Chi-squared distance can be calculated as

$$Chi-squared_{i,j} = \sum_{b=1}^{B} \frac{(j_b - i_b)^2}{(j_b + i_b)}, \tag{1.2}$$

Another possibility is to use the Mahalanobis distance, calculated as

$$Mahalanobis_{i,j} = \sqrt{((\vec{i} - \vec{j})' \Sigma^{-1} (\vec{i} - \vec{j}))}, \tag{1.3}$$

which represents the distance between two B-dimensional points ($\vec{i}$ and $\vec{j}$) scaled by the statistical variation in each component of the point (obtained through the covariance matrix Σ).

It is worth noting than the Mahalanobis distance is the same as the ED when the covariance matrix is the identity matrix.

The principal advantage of this distance is that it makes it possible to take into account the relevance that the same absolute difference can represent in different bands of an hyperspectral image.

The Spectral Information Divergence (SID) was proposed by [155] as a derivation from the divergence concept used in information theory. It is calculated as

$$SID_{i,j} = D(i||j) + D(j||i), \tag{1.4}$$

where

$$D(i||j) = \sum_{b=1}^{B} p_b \log\left(p_b/q_b\right) \tag{1.5}$$

and

$$D(j||i) = \sum_{b=1}^{B} q_b \log\left(q_b/p_b\right) \tag{1.6}$$

where p and q are the probability vectors associated to the pixel-vectors i and j, respectively, obtained as $p_b = i_b / \sum_{b=1}^{B} i_b$ and $q_b = j_b / \sum_{b=1}^{B} j_b$.

SID considers each pixel as a random variable whose spectral histogram is used to define a probability distribution. The spectral similarity between two pixels is then measured by the discrepancy of probabilistic behaviors between their spectra. This characteristic means that SID can be seen as a probabilistic approach, whereas other distances, such as Euclidean or Spectral Angle Mapper (SAM), are deterministic. One advantage of the probabilistic view of SID is that it can be used as a single-pixel measure as it measures spectral variability of a single mixed pixel from a probabilistic point of view.

Several papers indicate that SAM is a better metric for comparing spectra in hyperspectral datasets than ED, Chi-squared, Mahalanobis, or SID [104]. Its independence of the number of spectral components allows images of different spectral dimensionality to be compared. Besides, it is invariant to scale changes, making it insensitive to variations in illumination [105, 104]. The SAM distance is computed as shown in Equation 1.7

$$\alpha_{i,j} = \frac{2}{\pi} \cos^{-1}\left(\frac{\vec{j}\cdot\vec{i}}{\|\vec{j}\|\|\vec{i}\|}\right) \in [0,1], \tag{1.7}$$

where $\alpha_{i,j}$ is the spectral angle between the spectrum at pixel $\vec{i}$ and the one at $\vec{j}$).

1.3 Feature extraction and dimensionality reduction

The huge amount of information contained in the hyperspectral images must be correctly processed in order to be meaningful and to be efficiently processed. In order to do so, FE techniques are widely used to retain only the significant information contained in the raw datasets. The FE techniques are applied in hyperspectral data to avoid the Hughes phenomenon [140], which states that, in the context of supervised methods with limited training samples, the accuracy increases smoothly along with the spectral dimensionality of the data until some dimensionality is reached that causes a severe decrease in the accuracy achieved. The following paragraphs display a number of techniques for FE that have been applied in remote sensing hyperspectral images.

1.3.1 Principal Component Analysis

The most common FE technique is Principal Component Analysis (PCA) [83]. It is an unsupervised technique that performs a lineal transformation of the original data space in order to obtain a new representation where all the components are uncorrelated. Then, the number of Principal Components (PCs) that will be retained can be selected in relation to their significance. There exist different approaches in order to perform PCA. It can be done by Single Value Decomposition (SVD), or Eigenvalue Decomposition (EVD) or even by using an iterative approach, such as the Gram-Schmidt algorithm. In the EVD approximation, the eigenvectors are used to extract the PCs in decreasing order of significance by performing the following mathematical operations:

For a pixel-vector of n features $\mathbf{x} = (x_1, ..., x_n)^T$, where X is an n-dimensional image with K pixel-vectors whose mean across each dimension is zero, the mean position of the pixels in the space μ is defined as

$$\mu = \frac{1}{K} \sum_{k=1}^{K} \mathbf{x}_k. \tag{1.8}$$

The covariance matrix is obtained by

$$\sum_x = \frac{1}{K} \sum_{k=1}^{K} \left[(\mathbf{x}_k - \mu)(\mathbf{x}_k - \mu)^T \right]. \tag{1.9}$$

To obtain the PCA, the covariance matrix is transformed as

$$\sum_x = EDE^T, \tag{1.10}$$

where D is the diagonal matrix with eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$, and E the eigenvector matrix of $\sum_x$. Each pixel vector $\mathbf{x}_k$ is then transformed to $\mathbf{y}_k$ as

$$\mathbf{y}_k = E^T \mathbf{x}_k, \qquad k = 1, 2, \ldots, K. \tag{1.11}$$

PCA makes it possible to retain a high percentage of the variance of the original image keeping a reduced number of PCs.

Kernel Principal Component Analysis (K-PCA) [156] is an extension of PCA that uses kernel methods to perform the component analysis in nonlinear spaces, the only restriction being that the space must have the structure of a reproducing kernel Hilbert space [75].

1.3.2 Independent Component Analysis

Independent Component Analysis (ICA) is a method that pursues the disaggregation of a non-Gaussian multivariate signal into additive sub-components that are statistically as independent between them as possible. ICA is a particular case of the Blind Signal Separation [157] problem.

Let $\mathbf{x} = (x_1, ..., x_n)^T$ be the data represented by a random vector and $\mathbf{s} = (s_1, ..., s_n)^T$ the components of the random vector, ICA aims to transform the observed data $\mathbf{x}$ into maximally independent components $\mathbf{s}$ through the calculation of a linear static transformation $\mathbf{W}$ so that

$$\mathbf{s} = \mathbf{W}\mathbf{x}. \tag{1.12}$$

To do this, a function $f(s_1, ..., s_n)$ is required in order to measure the independence between the components of $\mathbf{s}$. The minimization of mutual information and the maximization of non-Gaussianity are the most common functions of independence for ICA

1.3.3 Non-parametric Weighted Feature Extraction

The Non-parametric Weighted Feature Extraction (NWFE) method was originally proposed in [79] to outperform the disadvantages of previously existent methods, such as Discriminant Analysis Feature Extraction (DAFE). The DAFE method presents three main disadvantages:

It only works for normal-like distributions. It only can extract $L-1$ features, where L is the number of classes. Finally, it achieves poor results when the within-class covariance matrix is singular, something which is common in high-dimensional problems.

Let $\mathbf{x}_k^{(i)}$ be the k^{th} sample of the class i, NWFE assigns different weights on every sample to compute the *weighted means* and it calculates the distance between samples and their weighted means as their *closeness* to boundary. The algorithm gives more weight to nearby samples, and defines new non-parametric between-class and within-class scatter matrices ($\mathbf{S}_b$ and $\mathbf{S}_w$), giving large weights to the samples close to the boundary and small weights to those samples far from the boundary. In short, the NWFE algorithm works as follows:

1. Compute the distances between each pair of sample points and form the distance matrix.
2. Use the distance matrix to compute $\mathbf{W}_l^{(i,j)}$, the weight for estimating the weighted means.
3. Compute the local weighted means $\mathbf{M}_j(\mathbf{x}_k^{(i)})$ in relation to the previously calculated $\mathbf{W}_l^{(i,j)}$.
4. Compute the scatter matrix weight $\lambda_k^{(i,j)}$ as a function of $\mathbf{x}_k^{(i)}$ and $\mathbf{M}_j(\mathbf{x}_k^{(i))}$ by using EDs.
5. Compute the between-class and within-class scatter matrices $\mathbf{S}_b$ and $\mathbf{S}_w$ as defined by the algorithm (See [79] for more details) and regularize $\mathbf{S}_w$ to prevent singularities.
6. Select the f eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S}_b$ corresponding to the f largest eigenvalues as the extracted features.

1.3.4 Stacked autoencoders

Autoencoders (AEs) are neural networks designed for the unsupervised learning of efficient coding. They work with unlabeled data, revealing hidden useful features. For this reason, one of their principal applications is the dimensionality reduction of multidimensional inputs [82, 85].

AEs work through an encoder (f) and a decoder (g) function. The encoder maps the original data ($\mathbf{x} \in \Re^d$) to an internal representation ($\mathbf{h}$), also known as *code*, while the decoder maps the created representation to an output ($\hat{\mathbf{x}}$).

Given an AE with one hidden layer, the encoder function is expressed as

$$\mathbf{h} = f(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{1.13}$$

where $\mathbf{W}$ is the learned weight matrix, $\mathbf{b}$ a bias vector, and g the sigmoid function defined as

$$g(t) = \frac{1}{1 + \exp(-t)}. \tag{1.14}$$

Thus, the decoder function maps $\mathbf{h}$ to a reconstruction of the input as

$$\hat{\mathbf{x}} = f'(\mathbf{x}) = g(\mathbf{W}'\mathbf{x} + \mathbf{b}'), \tag{1.15}$$

where $\mathbf{W}'$ and $\mathbf{b}'$ are the decoding weight matrix and a bias vector, respectively. It is common to tie the encoding and decoding weight matrices so that $\mathbf{W}' = \mathbf{W}^T$.

In order to evaluate the quality of the compressed representation learned, a loss function is used to measure the similarity between the input and the output representations. The most commonly used loss function is the mean squared error ($l = ||\mathbf{x} - \hat{\mathbf{x}}||^2$). Back-propagation is used to update the learned weights until the achieved reconstruction is good enough.

A Stacked Autoencoder (SAE), as shown in Figure 1.4, consists of multiple layers of AEs where the output of each layer is wired to the input of an adjacent layer. A reduction in the number of neurons between adjacent layers forces the AE to learn a more compact representation of the data. In the case of Figure 1.4, the SAE has two layers, the input of the SAE has n components, the first layer uses 4 neurons to reduce the original dimensionality, and the second layer provides a final compressed representation with three components.

1.3.5 Robust Color Morphological Gradient

The vectorial gradient operators are commonly used as a pre-processing step to image segmentation due to its capability to enhance the boundaries of the objects and between areas in an image. These transitions are characterized by gray-level intensity changes.

The morphological gradient operator for gray-scale images is defined as:

$$\nabla(f) = \delta_g(f) - \varepsilon_g(f), \tag{1.16}$$

where $\delta_g(f)$ and $\varepsilon_g(f)$ are the morphological operators for dilation and erosion, respectively, and g a Structuring Element (SE) that defines the neighborhood of a pixel in the image f.

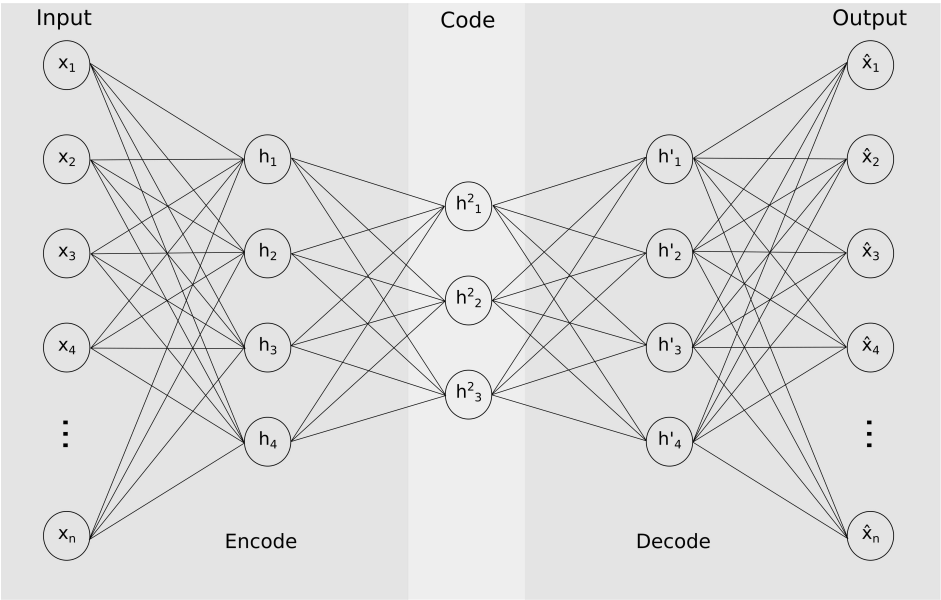


Figure 1.4: SAE scheme, including two hidden layers.

The gradient for n-dimensional images can be calculated by extending Equation 1.16 through multidimensional or vectorial gradients. The former approach consists on independently calculating the gradient for each band of the image and taking the final gradient value as the sum or maximal of these gradients. The later approach consists in calculating the gradient value between pixels as the distance between them. Euclidean, Mahalanobis, SAM, or chi-squared distances are some of the possible similarity measures that can be used to this end [155, 104, 154].

The Color Morphological Gradient (CMG) [158] is a vectorial gradient defined as

$$CMG = \max_{i,j \in \chi} \left\{ \left|\left| \mathbf{x}_k^i - \mathbf{x}_k^j \right|\right|^2 \right\}, \tag{1.17}$$

which returns the maximal ED between all pairs of pixel-vectors $\mathbf{x}_k$ in the set $\mathbf{X}$. Nevertheless, the CMG is highly sensitive to noise, which may produce edges that are not representative.

In order to improve the CMG, the Robust Color Morphological Gradient (RCMG) [158] was proposed as

$$RCMG = \max_{i,j \in \chi - R_S} \left\{ \left|\left| \mathbf{x}_k^i - \mathbf{x}_k^j \right|\right|^2 \right\}, \tag{1.18}$$

where R_S is the set of S pairs of pixel vectors furthest apart.

A 1-band gradient image is obtained after applying the RCMG, CMG or other vectorial gradient, regardless of the dimensionality of the input image.

1.4 Segmentation techniques

The segmentation techniques can be divided in two groups in relation to their application to multidimensional images. In the first group are those techniques that perform a dimensionality reduction of the image considered. In the other group are those that maintain all the spectral information contained in the image during the processing. In this thesis, the watershed segmentation [159], Really Quick Shift (RQS) [64] and Evolutionary Cellular Automata Segmentation (ECAS-II) [65] techniques are used. The watershed segmentation operates over an image previously reduced to one band. The RQS and the ECAS-II algorithm perform an implicit segmentation through the averaging of the spectra of the pixels belonging to the same region, maintaining all the spectral information.

1.4.1 Watershed transform

One of the most widely used methods for image segmentation is the watershed transform. This unsupervised method is particularly suitable for the case of low-contrast images.

The term watershed refers to a ridge dividing areas drained by different river systems. A catchment basin is the geographical area draining into a river and the lines dividing the different catchment basins are called watershed lines [159]. Similarly, a gray-scale image can be seen as a topographic relief where higher values or gray level represent higher areas and the watershed lines divide regions with different gray levels.

There exist two main group of watershed algorithms: those based on recursive methods [159] and those based on distance functions [160]. An intuitive vision of the workflow of the watershed segmentation is to imagine a terrain with holes pierced in local minima [161]. By flooding the terrain into water, catchment basins will fill up with water, as illustrated in Figure 1.5. A dam is constructed in the points where water coming from different basins meet,

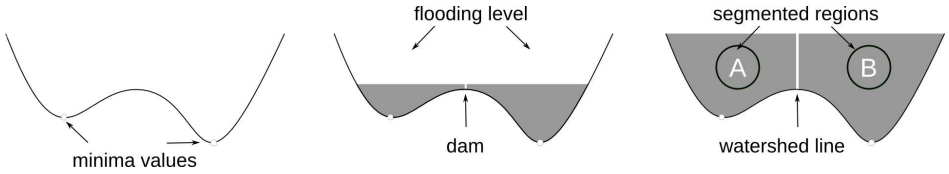


Figure 1.5: Flooding process in a image with two minima, generating a watershed line.

representing a watershed line. The process stops when the water level reaches the highest peak. The terrain is then partitioned into regions separated by the watershed lines.

The watershed transform always produces dense segmentation maps, even in low contrast scenarios. Nevertheless, the number of detected regions is usually too high, producing over-segmented maps. This drawback can be mitigated by applying some pre-processing to the image with the aim of reducing the number of regions.

When computing the watershed transform, each pixel must perform computations regarding those pixels in its closest neighborhood. The selection of the pixels to consider is called *connectivity*. There are two widely used connectivities. The Von Neumann neighborhood considers the orthogonal neighbors of the pixel (left, up, right, and down), establishing a connectivity of size four, whereas the Moore neighborhood takes into account the eight neighbors surrounding the pixel and, therefore, the size of the connectivity is eight.

A region of the image sharing the same gray value is called a *plateau*. If the lower border of a plateau is zero (it is impossible to reach a point of lower altitude without climbing), it is called a minimum plateau, otherwise, it is called a non-minimum plateau.

Among the different possible implementations for simulating the flooding process of the watershed transform [161], the Hill-Climbing algorithm based on the topographical distance by Meyer is adhered to in this thesis [160]. As it can be seen in Figure 1.6, this processing starts by labeling all the minimum plateaus in the image with unique labels (Figure 1.6 (a)) and then propagating these labels upwards, *climbing up the hill*, in the direction defined by the lower slope of each pixel (b and c in the figure). At the end, this processing generates a segmentation map in which every pixel belongs to a region and the limits between regions are identified by the watershed lines (Figure 1.6 (d)).

The watershed transform fits particularly well in a Cellular Automata (CA) computing model. CAs were originally proposed by Von Neumann [162]. They allow to complex problems to be modeled with the help of local information only. CAs are composed of a set of cells arranged in a grid. For the case of two dimensional grids, the cells are linked to their adjacent

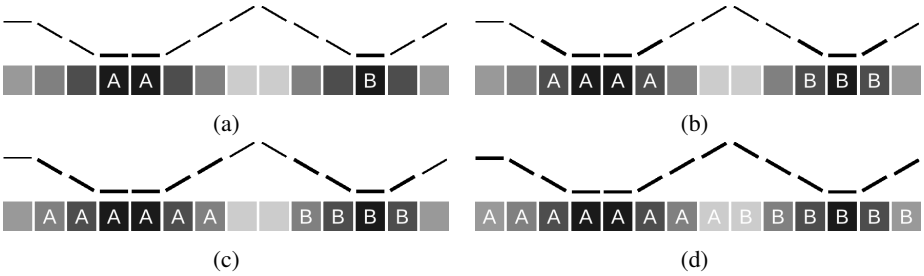


Figure 1.6: Watershed based on Hill-Climbing algorithm. Example of 1-Dimensional (1D) image represented as a terrain by dashed lines and as gray intensity values by squares: detecting and labelling all minima in the image with unique labels ("A" and "B") (a), propagating the labels upwards *climbing up the hill* (b,c), result of the segmentation in two regions (d).

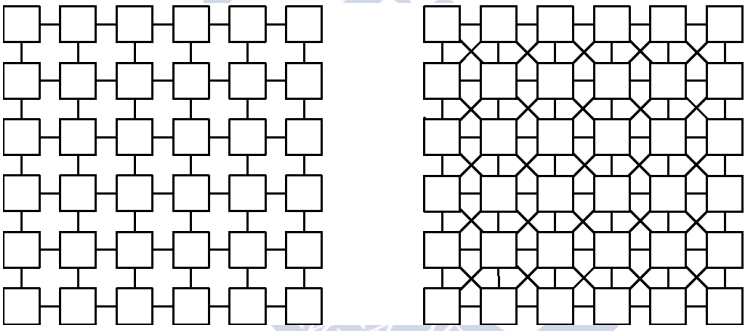


Figure 1.7: Cells of a CA arranged in a regular grid of two dimensions using the Von Neumann (left) and Moore (right) connectivities.

neighbors establishing a connectivity, as displayed on Figure 1.7. Therefore, two dimensional CAs can be mapped to two dimensional images, considering each pixel of the image as a cell of the automaton.

Each cell in a CA can be in one of a finite number of possible states. This state changes regarding its own current state and that of the cells in the considered connectivity in a process called updating of the CA. If the updates take place at the same time for all the cells, the CA is considered synchronous. Nevertheless, if the cells can be updated an unbounded number of times without synchronization, the automaton is asynchronous, which allows it to be partitioned into different regions that can be processed independently. An efficient computation of the watershed transform based on CA is presented in [163].

1.4.2 Really Quick Shift

RQS [64] is an algorithm to perform segmentation that can be applied to any feature space. The concept is similar to the mean shift algorithm [164]. It connects each one of the N points $(x_1 \cdots x_N)$ in the feature space with the nearest point in terms of a density estimate. This density $(P(x))$ is calculated around each point through the Parzen density, using the isotropic Gaussian window, as follows:

$$P(x) = \frac{1}{2\pi\sigma^2 N} \sum_{i=1}^{N} e^{\frac{-||x-x_i||^2}{2\sigma^2}}. \tag{1.19}$$

where σ is the standard deviation of the distribution.

Each connection established has an associated distance d_x, and the set of connections for all the pixels forms a tree where the root is the point with the highest density estimate.

RQS accounts for both the spectral and spatial information including the (x,y) position in the image along with the spectral feature space. The spectral space is re-scaled by a factor λ to adjust the trade-off between the spectral and spatial features. A threshold τ is also established to break all the links in the tree for which distance d_x is greater than τ.

1.4.3 Evolutionary Cellular Automata Segmentation

The ECAS-II segmentation algorithm [65] is an iterative method that segments hyperspectral images preserving the original dimensionality of the image data. By preserving the original dimensionality, the loss of potentially useful information is avoided, thus achieving a better segmentation of the image. To achieve this, for each iteration of the algorithm, the spectrum of each cell (pixel) is combined with the spectra of some of its neighbors, usually through weighted averaging. The selection of the neighboring pixels that will contribute to the final combined spectrum depends on a set of transition rules controlling the operation of a CA and which determine the complexity and behavior of the segmentation algorithm. Once the CA is iteratively applied to the datacube, the final output will be a transformed hyperspectral image in which pixels that belong to the same class share a similar spectrum. This means that the hyperspectral image is implicitly segmented but, unlike most segmentation methods, the final datacube preserves the complete wealth of spectral bands, avoiding the loss of spectral information. In other words, the segmentation is carried out in terms of spectra and not of any kind of projection or lower dimensional representation.

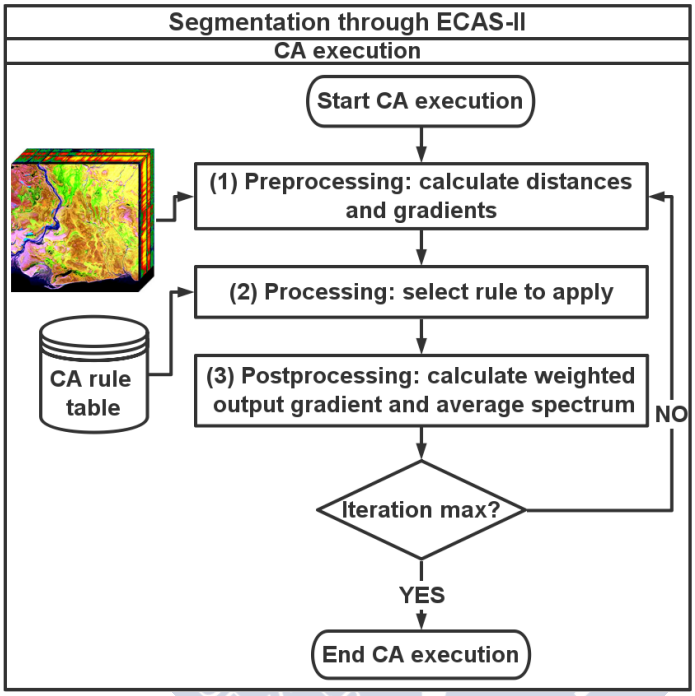


Figure 1.8: ECAS-II segmentation algorithm flowchart.

The segmentation algorithm is described in Figure 1.8. The algorithm comprises three main stages. As a first step, some neighborhood information is extracted for each cell (each cell maps to a pixel i whose state S_i is given by the B-band spectrum of the pixel). This information is represented in the form of spatio-spectral gradients taking into account the pixels contained in three different $N \times N$ 2-Dimensional (2D) windows, where $N \in \{3,5,7\}$. The computation of the spatio-spectral gradients requires the calculation of the spectral distance between each pixel and all of its neighbors in a given window. For this purpose, the normalized spectral angle distance was chosen,

$$\alpha_{i,j} = \frac{2}{\pi}\cos^{-1}\left(\frac{\sum S_j S_i}{\sqrt{\sum S_j^2}\sqrt{\sum S_i^2}}\right) \in [0,1], \tag{1.20}$$

where i is the reference pixel, j is a pixel in the neighborhood window, $\alpha_{i,j}$ is the spectral angle for cell i with respect to cell j, and the summation is performed over the components of the state of S_i, i.e., the spectral dimension of a pixel.

The spectral angle is widely used as a spectral similarity measure for material identification [105, 104]. The main advantage of using this spectral distance is that it is independent of dimensionality, which enables the method to be applied over RGB, multispectral or hyperspectral images.

Thus, for each pixel i, three spectral-spatial gradients (being $G_{X_N}(i)$ the horizontal component and $G_{Y_N}(i)$ the vertical component) are calculated using the spectral distances that were previously obtained and considering the three spatial window sizes $N \in \{3,5,7\}$:

$$G_{X_N}(i) = \sum_{j=1}^{N \cdot N} \alpha_{i,j} M_{X_{N_j}}, \quad G_{Y_N}(i) = \sum_{j=1}^{N \cdot N} \alpha_{i,j} M_{Y_{N_j}}, \tag{1.21}$$

where $M_{X_{N_j}}$ and $M_{Y_{N_j}}$ represent the j^{th} elements of the horizontal/vertical gradient masks M_{X_N} and M_{Y_N}.

As three different window sizes are considered in this version of the ECAS-II segmentation algorithm, the gradient calculation will lead to three moduli ($|G_N(i)|$) and angles ($\phi_N(i)$), expressed as:

$$|G_N(i)| = \sqrt{G_{X_N}^2(i) + G_{Y_N}^2(i)}, \quad \phi_N(i) = \tan^{-1}\left(\frac{G_{Y_N}(i)}{G_{X_N}(i)}\right). \tag{1.22}$$

The second step of the method consists in selecting one rule from the set of transition rules that make up the CA based on the information of gradient moduli and gradient angles.

The set of transition rules comprises a group of M rules, each one containing 6 parameters:

$$CA = \begin{pmatrix} |\mathfrak{G}_{3,1}| & |\mathfrak{G}_{5,1}| & |\mathfrak{G}_{7,1}| & \Phi_{5,1} & \Phi_{7,1} & \theta_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ |\mathfrak{G}_{3,k}| & |\mathfrak{G}_{5,k}| & |\mathfrak{G}_{7,k}| & \Phi_{5,k} & \Phi_{7,k} & \theta_k \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ |\mathfrak{G}_{3,M}| & |\mathfrak{G}_{5,M}| & |\mathfrak{G}_{7,M}| & \Phi_{5,M} & \Phi_{7,M} & \theta_M \end{pmatrix}, \tag{1.23}$$

where the first five parameters of each rule in (1.23) correspond to a direct representation of spatio-spectral gradients, three gradient moduli ($\mathfrak{G}_{N,k}$), and two gradient angles ($\Phi_{N,k}$), $k \in [1,M]$. The angle of the gradient for the window of size 3 ($\Phi_{3,k}$), was not included as

it is considered equal to 0 with the aim of making the three vector gradients, as a group, independent from angle rotations and reflections. Thus, the selection of one rule from the ruleset is performed by comparing the neighborhood information (moduli and angles of the spatio-spectral gradients) to the first five parameters of each one of the M rules and selecting the closest in terms of ED.

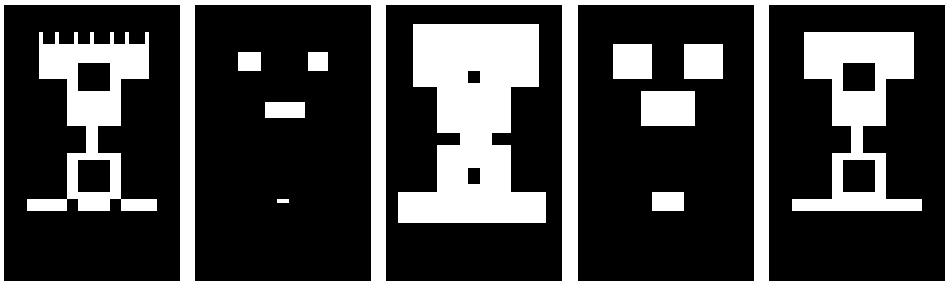
The final parameter of each rule (θ_k) contains the information needed to update the pixel spectrum. This parameter determines which of the neighboring pixels will contribute to the updated pixel's spectral average.

The transformed image is calculated by applying a filter to the spectra of the neighborhood pixels (stage 3 on Figure 1.8) so that the weight of a pixel decreases as the distance to the central one increases. This kind of filter is used because classification accuracies can be improved when a smoothing is applied to the hyperspectral image [165, 122]. Different filters assigning weights that are inversely proportional to the distance have been tested with different distance measures to achieve the best accuracy results. The transformed image is then taken as the input to the next iteration of the process or as the final output if the maximum number of iterations was reached.

The segmented hyperspectral datacube obtained after the application of the ECAS-II segmenter can be used as input to other processing tasks, such as the labeling of the datacube obtaining a 2D classification map.

1.5 Mathematical morphology

Mathematical Morphology (MM) is a theory for the analysis and processing of the spatial structures present in images [166]. It relies on two basic operators: erosion (ε) and dilation (δ), used to construct more advances tools. Both erosion and dilation operators are applied to an image through the use of an SE. An SE can have different sizes and shapes (circle, square, etc.) that determines the neighboring pixels involved in the calculus of the erosion and dilation of a given pixel. The erosion operator provides the *infimum* intensity value of the pixels covered for the SE whereas the dilation operator provides the *supremum* of the same pixels. Plainly speaking, the erosion operator shrinks objects brighter than their environment and the dilation operator enlarges them. Figure 1.9 shows the the effects of applying the erosion (b) and dilation (c) to an image (a).



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

Figure 1.9: Example of morphological operators application. From left to right: Original image of size 64 $\times$40 pixels (a), eroded image (b), dilated image (c), opened image (d), and closed image (e), considering an SE with disk shape of size 3.

The erosion and dilation operators can be used together, giving rise to the opening (γ) and closing (ϕ) operators. The opening operator corresponds to the application of the dilation to an eroded image and the closing operator corresponds to the opposite combination. Figure 1.9 (d) and (e) shows the the aplication of the opening and closing operators, respectively. The opening operator highlights those bright elements in the image where the SE can be fitted. The closing operator preserves dark regions that can completely contain the SE.

As stated before, these operators work on pixels instead of image structures. This may lead to the loss of edges between objects by the merging of adjacent regions. To mitigate this drawback, the opening and closing operators can be built based on the geodesic reconstruction so that the spatial structures of the image will be completely preserved or removed, depending on whether they fit in the SE or not [166]. The opening and closing by reconstruction are defined as the reconstruction by dilation/erosion from the erosion/dilation with a given SE. In the first place, a marker image is constructed by applying erosion or dilation. Subsequently, the correspondent reconstruction is applied as an iterative process on the marker image that lasts until stability is reached. This processing allows all structures that were not completely removed at the moment of creating the mask to be fully recovered.

By applying the opening and closing with a given SE to an image, the objects will be filtered or preserved depending on whether they match the SE size. Nevertheless, it is hardly possible to find a single SE that makes it possible to handle all objects in an image. For this reason, it is useful to perform a multilevel analysis with an increasing size of the SEs considered, creating a stack of images with different degrees of filtering [167] known as the

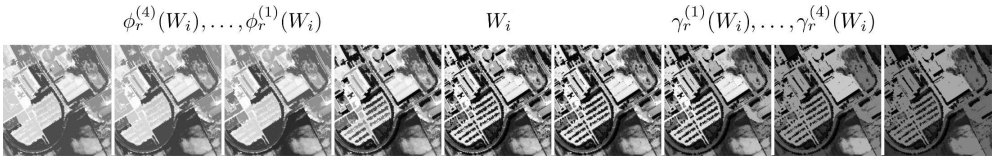


Figure 1.10: Morphological profile of a gray-level image W_i considering 4 different SE sizes. Closing images (ϕ) on the left side and opening images (γ) on the right side.

Morphological Profile (MP) of the image, as shown in Figure 1.10. If this technique is applied to each band of a multidimensional image, the stack of all the MPs is called an Extended Morphological Profile (EMP) and will allow us to extract the spatial information of the image at all the considered wavelengths. In the case of hyperspectral images, if all the components of theEMP are stacked together, the size of the stack can be excessively large. For this reason, it is common to perform a dimensionality reduction of the image (for instance, by means of PCA) before the creation of the EMP.

1.6 Supervised classification

A pixel-wise classification is the process whereby a unique label is assigned to each pixel in an image regarding the spectral signature of the pixel-vectors. This process can be tackled by supervised and unsupervised techniques.

Unsupervised approaches rely only on the information that can be extracted from the image itself to find a hidden structure from unlabeled data. Due to the fact that the inputs are unlabeled, it is not possible to perform an accuracy assessment of the result. Most of the unsupervised techniques consider the inputs as a set of random variables and attempt to find a density model allowing the separability of the samples in different classes. The number of classes is usually fixed as an input parameter. The most common unsupervised classifiers used in remote sensing are the K-means [168] clustering and the Iterative Self-Organizing Data Analysis Technique (ISODATA) [169].

In the case of supervised approaches, an initial stage where some a priori knowledge of the image is used to learn the main characteristics that define each one of the possible labels is needed. This knowledge is provided in the form of previously labeled pixels known as training samples. The use of correctly labeled pixels is critical in the learning process. For this reason, it is common to use a set of labeled pixels provided by an expert or achieved

through an in place visual exploration of the terrain to extract the training samples. This set of labeled samples is called *reference data*, and it is normal to have standard reference data for the most common datasets in in the literature available. The appropriate selection of the training samples is also a key factor for achieving a successful classification map of the image in the second clasification stage, where each pixel in the image is assigned to a specific class in accordance with the training previously performed.

Different approaches, such as linear discriminant analysis, maximum likelihood, random forests, nearest neighbors, artificial neural networks, Support Vector Machines (SVMs) or Extreme Learning Machines (ELMs), have been proposed in the past to be used as classifiers [43, 74, 51, 170, 171, 53, 172, 173, 174]. An evaluation of classifiers belonging to a wide collection of families over the entire University of California, Irvine machine learning classification database [175] , was presented in [172].

The SVM supervised classifier has been used as a gold standard in remote sensing applications as it achieves good accuracy results in environments with a reduced number of available training samples. This is usually the case in the field of remote sensing and particularly hyperspectral images. In recent years, the ELMs have been shown to be a good alternative to the SVM classifiers, achieving similar results in terms of accuracy in the same environments but with a lower execution time. The reason why the ELM is faster is that some of the parameters of the neural network are randomly fixed, avoiding the computation of back propagation processes.

In this thesis, the ELM classifier is selected and compared with the SVM in terms of accuracy and execution time. An efficient Graphics Processing Unit (GPU) implementation of the ELM algorithm for hyperspectral images was developed and it is detailed in Chapter 3.

1.6.1 Support Vector Machines

The traditional SVM classification method is a binary process aimed at finding the optimal hyperplane separating two training samples, members of different classes, maximizing the distance between the closest points of each class.

Given a set of N training points $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, assuming that they are separable into two classes, with training vectors $\mathbf{x}_i \in \Re^n$ and corresponding targets $y_i \in \{\pm 1\}$, the hyperplane can be estimated as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 1 \quad with \quad i = 1, 2, \ldots, N, \tag{1.24}$$

where $\mathbf{w} \in \Re^n$ is the hyperplane defined by the vector and $b \in \Re$ is the bias. Those training samples that maximize the distance are named support vectors. Maximizing the distance of samples to the optimal hyperplane is equivalent to minimizing the norm of $\mathbf{w}$.

The assumption of linear separability implies that there exist $\mathbf{w}$ and b satisfying Equation 1.24. In the case that data are linearly inseparable, non-negative slack variables are needed to handle misclassified samples [176]. Moreover, a constant parameter C is introduced as a penalty term to minimize the error in the minimization of the norm of $\mathbf{w}$.

The SVM classification is performed by computing the sign of the decision function

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b, \tag{1.25}$$

where α_i is the nonzero Lagrange multipliers, S the subset of training samples $\mathbf{x}_i$ and $\mathbf{x}$ the unknown sample. The training phase is responsible for finding the parameters α_i, S, and b.

In order to enhance the linear separability between classes, kernel tricks to map the data into higher dimensional spaces are usually introduced. The Radial Basis Function (RBF) kernel has been widely used for hyperspectral image classification, where the decision function is modified as

$$D(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}||^2) + b, \tag{1.26}$$

where γ is a positive parameter modifying the width of the kernel.

The adequate selection of the parameters C and γ is key in the quality of the classification obtained by SVM, A procedure known as k-fold cross validation can be use to optimize the selection of these parameters. During this procedure, several parameter values are tested, by means of a search in a range of values. The procedure divides the total number of samples into k partitions: one is used to validate the accuracy of the classification and the remaining k-1 are used for training the data. This process is repeated k times using each partition once for validation. The final classification accuracy is determined as the average of the k results.

As the standard SVM was designed to solve binary problems, a strategy for problems with a larger number or classes must be defined. There exist different approaches to deal with this task. Some methods perform a One-Against-All (OAA) classification, where a set of binary classifiers is used so as the ith classifier separates the ith class from the remaining ones. Others operate as One-Against-One (OAO) using $K(K-1)/2$ binary classifiers, where K is the number of classes, and performing a classification for each pair of classes. The final

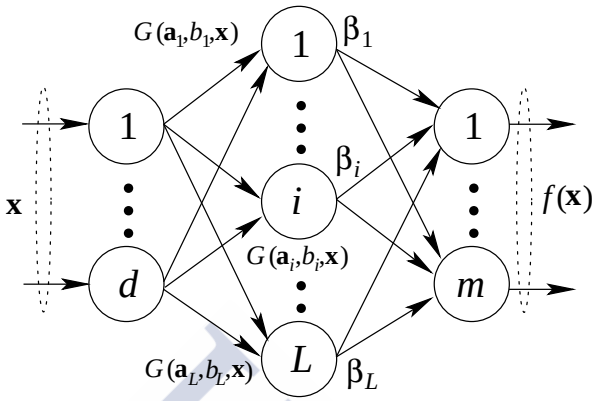


Figure 1.11: SLFN as used by ELM.

classification for each pixel is assigned to the class most repeated among all the classifiers. Finally, some methods work in an all-at-once strategy, combining a set of binary classifiers. The OAO approach is more suitable in practice as is the one with the shortest training time [177]. Therefore, it is the one used in the SVM classifications performed in this thesis.

1.6.2 Extreme Learning Machines

The raw pixel-wise ELM algorithm was proposed as an efficient learning algorithm for Single hidden Layer Feed-forward neural Networks (SLFNs) [127]. Figure 1.11 shows the structure of an SLFN. The output function of an SLFN with L hidden nodes, where $\mathbf{x}$ is the input vector, can be written as

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \ \beta_i \in \mathbb{R}^m, \tag{1.27}$$

where $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output function of the ith hidden node, being $\mathbf{a}_i$, b_i the hidden node parameters and β_i the weight vector connecting the ith hidden node to the output nodes. For the case of additive nodes with activation function g, it can be expressed as

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad \mathbf{a}_i \in \mathbb{R}^d, \ b_i \in \mathbb{R}. \tag{1.28}$$

An SLFN with L hidden nodes can approximate N arbitrary distinct samples and targets $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbb{R}^d \times \mathbb{R}^m$, if the following equation system can be solved:

$$\mathbf{H}\beta = \mathbf{T} \tag{1.29}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \dots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}, \tag{1.30}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times N}, \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1{}^T \\ \vdots \\ \mathbf{t}_N{}^T \end{bmatrix}_{N \times m}. \tag{1.31}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network. Huang et al. [178, 47] have proved that once they are randomly generated, the hidden node parameters $(\mathbf{a}_i, b_i)$ can remain fixed and training a SLFN is equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$, i.e.:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{1.32}$$

where $\mathbf{H}^{\dagger}$ is the *Moore-Penrose* generalized inverse of matrix $\mathbf{H}$ [179].

Thus, ELM can be summarized as shown in Fig. 1.12 [178, 47]. As it was stated in [48], ELM requires less human intervention than SVM as a single parameter, the number of neurons in the hidden layer, needs to be optimized, since all the other parameters are randomly initialized. The optimal selection of this single parameter is key to obtaining the best classification map. In addition, ELM has better scalability and it runs at much faster learning speed than SVM.

1.7 Registration

The registration process aims to obtain the geometric transformations needed to align at pixel level two or more images corresponding to the same area. This alignment will allow the later application of other multitemporal processes, such as the CD. This process can be tackled by

ELM training algorithm.
Input: training set $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$.
Output: SLFN parameters.
1: Randomly generate hidden node parameters $(\mathbf{a}_i, b_i),\ i = 1, \ldots, L$ where $\mathbf{a}_i$ and b_i are the input weight and bias values and L is the hidden node number.
2: Calculate the hidden layer output matrix $\mathbf{H}$.
3: Calculate the output weight vector, $\beta = \mathbf{H}^{\dagger}\mathbf{T}$.

Figure 1.12: ELM training algorithm.

area-based or feature-based methods. The former group includes methods that rely on correlation between images [180], mutual information [181], or the Fourier transform [182, 183] to directly work with the intensity of the images. The second group searches for distinctive features or points of interest at a higher level that share particular characteristics. They must be invariant to geometric transformations, have good localization accuracy, and be insensible to degradation. The scale-Invariant Feature Transform (SIFT) [184] is the most popular algorithm in this group, while the Speeded-Up Robust Features (SURF) method [185] was proposed as a faster alternative to SIFT. The datasets used in this thesis have been co-registered by using the Hyperspectral Fourier-Merlin (HYFM) method [182, 183] based on [137].

1.7.1 HYFM-based registration

Area-based registration methods are computationally more efficient. Among all the methods contained in this category, the ones based on the Fourier transform are particularly suitable to be applied efficiently to hyperspectral images.

The HYFM method was designed by our research group to exploit the information comprised in hyperspectral images by performing FE by means of PCA. After this, the corresponding pairs of PCs are used to apply the Multilayer Fractional Fourier Transform (MLFFT) technique [137] and obtain log-polar maps. Then, the phase correlation maps are obtained for each pair of log-polar maps and combined. Finally, the highest peaks of the maps are examined to determine the geometric transformations (scaling, rotation and translation) needed to align the considered hyperspectral images.

1.8 Change detection

The main focus of this thesis is CD in multitemporal hyperspectral datasets. The basic CD problem consists in discriminating pixels or areas of the considered dataset where changes have been produced in the time elapsed between the capture of the different images. In this case, the problem is called binary CD. Another approach aims to cluster or classify the different transitions between classes that have occurred over time. In this case the problem is called multiclass CD.

Another possible classification of CD techniques is based on the type of fusion of the multitemporal data, as shown in Table 1.1: at feature level or at decision level [12]. In the former, the information of both images is combined before performing the CD. The combination is usually based on algebraic operations over both images, such as image differencing, or image ratioing. The application of Normalized Difference Vegetation Index (NDVI) [32] or Change Vector Analysis (CVA) [33] are other alternatives. CVA is a technique that relies on the computation of distances and angles between pairs of pixels (each pixel is represented by a vector of values) corresponding to the same position in both images.

Other techniques, also based on the fusion of images at the feature level, perform FE over the image resulting from the fusion using, for example, PCA [153, 186]. Calculating morphological profiles, such as attribute profiles, is another well-known method in the remote sensing field for extracting spatial features from the images [152]. Context-based approaches, such as Markov Random Fields (MRFs) [35] or Expectation-Maximization-based Level Sets (EMLSs) [36], are applied in some papers.

Regarding those approaches where the fusion of the information provided by both images is carried out at the decision level, there are approaches based on post-classification that compute the changes as the differences in the classification of the two images [39]. Other techniques perform a direct multidate classification, i.e., the two images are combined and classified together [42].

A considerable number of the CD methods for multidimensional images available in the literature rely on CVA to discover the differences between the considered images. Hence, the result can be thresholded to obtain a binary CD map, or processed in different ways (for instance, with clustering techniques) to find different categories of change with common properties. The next section is devoted to explaining the fundamentals of these techniques. A scheme including a variant of the CVA technique is proposed in this thesis for the binary CD in hyperspectral images.

Fusion at the feature level	
Pixel-based operations	Image differencing, ratioing, NDVI [32], CVA [33]
FE	PCA [187, 188, 189, 153, 186]
Morphological profiles	Attribute profiles [152]
Context based approaches	MRF [35], EMLS [36]
Fusion at the decision level	
Post-classification	[39]
Direct multidate classification	[42]

Table 1.1: Classification of CD methods in terms of the fusion level of the multitemporal data.

1.8.1 Change Vector Analysis

The CVA technique is a well-known CD processing, originally proposed by [33]. The CVA algorithm comprises two measures of change: magnitude and direction. The magnitude of change is calculated as the ED between corresponding pairs of pixels in two co-registered images. This magnitude is enough to obtain a binary CD map. For this reason, the direction is disregarded in many works whose target is merely to detect the presence or absence of changes. The direction measure can be used to cluster different types of changes. This component can be calculated by sector-coding, as presented in [190], or by calculating vector direction cosines, as presented in [191].

A new version of CVA, called compressed change vector analysis (C^2VA) was presented in [192]. This version maintains all the spectral information of the multitemporal images while allowing a 2D representation of the data. It works over the multispectral difference image $\mathbf{X_D}$ obtained as $\mathbf{X_D} = \mathbf{X_2} - \mathbf{X_1}$, where $\mathbf{X_2}$ and $\mathbf{X_1}$ are the considered images of spectral dimensionality B. As stated in Equation 1.33, the magnitude ρ is calculated as the ED of the corresponding pixel-vectors of the two images. The direction α is computed as shown in Equation 1.34, as the angle between the vectors of $\mathbf{X_D}$ and a reference vector with all its components equal to $1/\sqrt{B}$.

$$\rho = \sqrt{\sum_{b=1}^{B} (\mathbf{X_D^b})^2} \tag{1.33}$$

$$\alpha = \arccos\left[1/\sqrt{B}\left(\sum_{b=1}^{B} \mathbf{X_D^b} / \sqrt{\sum_{b=1}^{B} (\mathbf{X_D^b})^2}\right)\right] \tag{1.34}$$

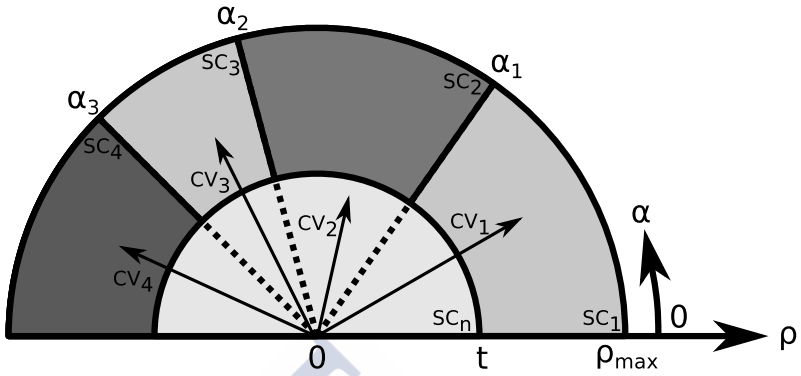


Figure 1.13: Example of polar representation of C^2VA including four different types of change.

By calculating the magnitude and direction of the multidimensional images as proposed in Equation 1.33 and 1.34, the obtained change information can be represented in a 2D polar representation, such as the one shown in Fig. 1.13. In the figure, the semicircle denoted as SC_n represents the magnitude values smaller than the threshold t, and therefore corresponding to pixels labeled as no change. The semi-annulus Sc_1, Sc_2, Sc_3, and Sc_4 correspond to the pixels whose magnitude is greater than the threshold, and which are considered as changes. Each semi-annulus represents a different kind of change regarding the direction angle α; thus, in this case, four different types of change are considered.

Recently, new versions of the algorithm have been presented. A new change vector representation for iteratively discovering different types of change, following a tree structure, is introduced in [37]. [193] takes into account the spatial information of neighboring pixels, together with the spectral one, through the inclusion of morphological processing.

1.8.2 Thresholding the changes

Regarding the thresholding process that is necessary to discriminate the change pixels, the most commonly used approaches are statistical methods, such as K-means clustering [194, 195], or Expectation-Maximization (EM) [196]; however, these techniques are computationally inefficient, as they are based on slow iterative methods. Instead, Otsu's method [103] allows suitable thresholds to be obtained with a much lower computational complexity, since it relies on histograms that can be efficiently computed on a GPU.

K-means

K-means is an iterative clustering algorithm that partitions n observations into k clusters, assigning each observation to the cluster with the nearest mean. The original algorithm was proposed by Lloyd in 1957 to perform pulse-code modulation [194]. K-means was first used by MacQueen in 1967, [197]. Hartigan and Wong presented a new, more efficient version in 1975 [195, 168].

The algorithm consists of an iterative process that refines the partitions in each iteration. It alternates between an assignment and an update step. In the assignment step, each observation is allocated to the cluster whose mean has the smallest ED with respect to the observation, obtaining a Voronoi diagram of the observations regarding the means of the clusters. In the update step, the means of the clusters are recalculated regarding the observations belonging to the cluster in the considered iteration. This process lasts until the assignments achieve stability. There is no guarantee that the optimum partition is achieved by the algorithm. Distance measures other then the Euclidean one can be applied, which can affect to the clusters obtained.

The algorithm needs to be initialized before the first assignment step. The most common options for performing this initialization are Forgy and Random Partitioning [198]. In the former approach, k observations are randomly chosen to be the initial means. In the latter approach, each observation is randomly assigned to a cluster and then the update step is performed. Both the initialization method and the particular initialization performed in each case also affect the achieved partitions.

Expectation-Maximization

The EM technique was presented in 1977 by Dempster et al. [196]. It consists of an iterative method to find the maximum likelihood parameters of a statistical model from known observations. The process alternates two different steps:

- The Estimation step is devoted to the calculation of the expected values for each observation of the likelihood function using the current estimate of the parameter values.

- The Maximization step re-estimates the parameters using the current expected values of each observation.

These steps are repeated until the parameter estimate converges; i.e., there is no longer change in the estimate.

The most common statistical model used in this case is the Gaussian Mixture Model (GMM), so that the number of Gaussians to estimate is equal to the number of clusters to be created. In this case, the parameters to be estimated for each Gaussian are the mean, the covariance and the mixing coefficients. A common approach is to use k-means to initialize the first estimation of the parameters.

At the end of the EM process, each observation will have a probability of membership to each cluster. The cluster with the largest probability is the one assigned to each observation.

Otsu's method

Otsu's method, proposed by Otsu in 1975 [103], performs clustering-based thresholding over a bi-modal histogram. This is done by calculating the optimum threshold that separates the two classes of observations by minimizing the intra-class variance and maximizing the inter-class variance.

The method works over a histogram through an exhaustive search in the possible threshold values. The threshold that minimizes the intra-class variance is that with the smallest weighted sum of variances, calculated as

$$\sigma_w^2(t) = w_0(t) * \sigma_0^2(t) + w_1(t) * \sigma_1^2(t) \tag{1.35}$$

where w_0, w_1 and σ_0^2, σ_1^2 are the probabilities and variances of the two classes, respectively, for a given threshold t.

The value of t that provides the minimum σ_w^2 is the one selected as the threshold for dividing the observations into two mutually exclusive groups.

1.9 HypeRvieW

HypeRvieW is a desktop application for the analysis and processing of hyperspectral images [132]. It is a license-free application that was developed by members of the investigation group using C language and the GTK library. It is based on plug-ins, so that the new techniques developed by the group, or any other user, can be easily added to improve the functionality of the application. The application is capable of dealing with data in MATLAB and raw

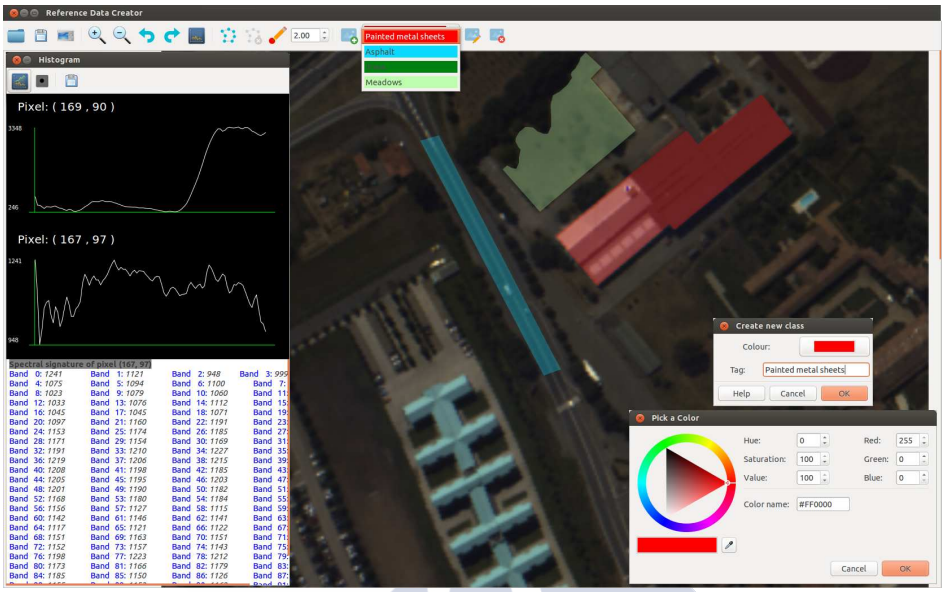


Figure 1.14: HypeRvieW reference data editor GUI.

formats and also propose a new format called Hyperspectral Raw (HRW) designed to interact easily with the application.

Concerning the analysis capabilities of the application, it includes visualization, zoom, crop, shift, rotation and scaling, band navigation, color composition or spectral signature comparison, among others. However, the most advanced capability of the application in this group is the reference data generation. As can be seen in Figure 1.14, HypeRvieW provides a Graphical User Interface (GUI) that allows the creation of reference maps with different labels and colors based on polygon structures. The information needed to obtain an adequate reference data can be acquired through the other analysis tools included in the application. Nevertheless, field knowledge or expert participation is always needed to obtain the best reference data possible.

Regarding the processing of hyperspectral images, HypeRvieW, in its current version, includes plug-ins for the registration and supervised spectral-spatial classification of hyperspectral images. The registration plug-in allows this task to be performed using the MLFFT and the HYFM methods. A pipeline classification chain makes it possible to combine watershed and RQS segmenters with SVM or ELM classifiers in order to obtain a spectral-spatial

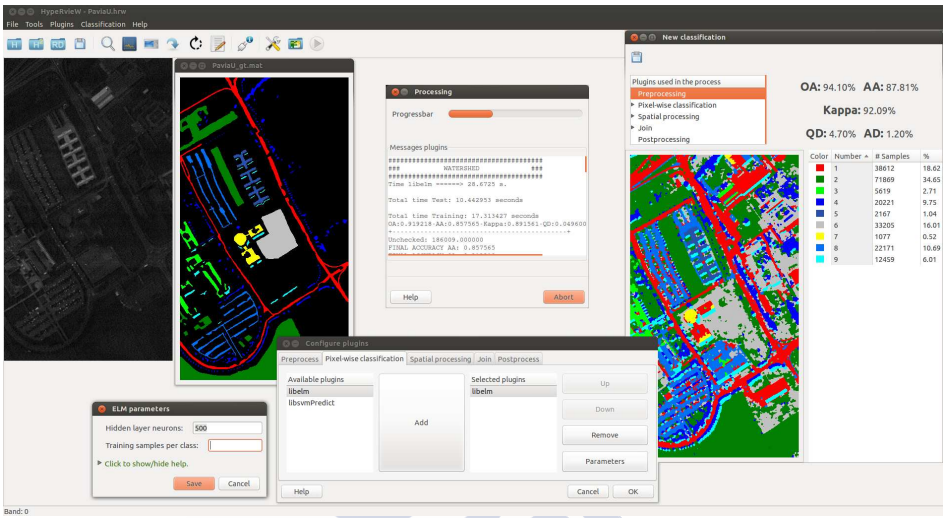


Figure 1.15: HypeRvieW classification GUI.

classification map. Figure 1.15 shows the configuration of a classification chain and, on the right side of the figure, the results achieved, in terms of both numeric accuracies and visual classification map.

HypeRvieW was intensively used during the realization of this thesis both for analysis of hyperspectral images and to carry out some of the experiments that will be introduced in the following chapters. Moreover, the registration and reference data creation capabilities of the application were used to prepare some of the datasets used in the experiments of this thesis.

1.10 Discussion

The main characteristics of the hyperspectral images have been introduced in this chapter. These images contain a large amount of information that makes them suitable for performing several remote sensing tasks. The different processing stages for extracting useful information from these images have also been analyzed showing that hyperspectral images are very valuable when they are correctly processed.

The *HypeRvieW* desktop tool was also presented. An application for the analysis and processing of hyperspectral images developed by the research group, providing an easy access to some of the hyperspectral processing schemes proposed in this thesis.

The central sections of this chapter introduce the fundamentals of different processing techniques to exploit the information stored in hyperspectral images:

- There are several similarity measures available for comparing the spectral signatures of different pixels. The use of the adequate measure is an important stage in several both in single image processing and in multitemporal image processing, such as CD.

- Different FE techniques can be used to project the hyperspectral data to more effective representations,facilitating their processing while maintaining the relevant information contained in the original data. This includes classical techniques, such as the PCA, but also currently trending techniques, such as SAE.

- Segmentation and mathematical morphology methods, such as watershed, ECAS-II, or EMP, can be used in the hyperspectral image domain to include the spatial information contained in datasets of this kind, along with the spectral information, in different processing schemes, giving rise to the so-called spectral-spatial schemes that provide better results than those schemes that consider the spectral information alone.

- The use of supervised classification techniques, such as SVM or ELM, allows the accurate categorization of the information contained in hyperspectral datasets. This makes it possible to obtain detailed classification maps providing useful information on the materials and objects present in a hyperspectral scene.

- The registration process aligns at pixel level hyperspectral images corresponding to different time frames, which makes it possible to perform subsequent processing tasks, such as the CD in multitemporal datasets.

- The CD in multitemporal hyperspectral images is an important task that can be tackled with different techniques. Some of the existing techniques focus on the discovery of changes and others also aim to group the detected changes into different categories. Nevertheless, there is a lack of efficient CD processing techniques that produce a detailed pixel level change classification map providing useful *from-to* change information.

The following chapters of this thesis will focus on the introduction of efficient techniques aimed at the achievement of a CD scheme for multitemporal hyperspectral datasets that provides a change classification map with detailed information on the different transitions occurred across time.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

EFFICIENT COMPUTATION OF HYPERSPECTRAL IMAGES ON GPU

Efficient computation, i.e. the execution in the lowest possible time while exploiting the available computational resources, is fundamental in any current application. In particular, this thesis develops techniques whose execution time, regarding the context of application, may be critical. For instance, in the field of catastrophe monitoring, such as oil spreading in the sea or forest fire control.

It is also desirable to avoid the use of large computational infrastructures and, instead, take advantage of the architectures that are usually available in remote sensing environments. Specifically, this thesis focuses in the achievement of efficient implementations for multicore Central Processing Units (CPUs) and NVIDIA GPUs suitable for commodity hardware, such as desktop computers. The main characteristics of this type of hardware are that it is inexpensive, widely available, and easily interchangeable with other hardware of the same type.

Two different parallel programming paradigms were used to achieve efficient implementations of the schemes and techniques developed in this thesis: the OpenMP and Compute Unified Device Architecture (CUDA) parallel programming models. The first section of this chapter is devoted to the revision of the main characteristics of these two paradigms. The different techniques used to achieve efficient GPU implementations are then described.

The next section of this chapter details the experimental setup used during the experiments carried out to validate the schemes proposed in this thesis. First, the hardware used is introduced. Then, the performance metrics applied to evaluate the quality of the proposed

techniques are detailed, including accuracy, execution time, and hardware occupancy metrics. Finally, the different datasets used for the validation of the schemes are introduced and their main characteristics are analyzed in detail.

2.1 Parallel programming models

OpenMP and CUDA were the programming paradigms used during the development of this thesis. OpenMP is an Application Program Interface (API) devoted to the writing of multi-threaded applications on shared memory architectures in CPU. The CUDA platform, developed by NVIDIA, is a hardware/software combination for the development of parallel applications to be executed in NVIDIA GPUs.

2.1.1 OpenMP

The OpenMP (abbreviation for *Open Multi-Processing*) API was defined by a group of major computer hardware and software vendors and its first version was published in 1997. It provides a portable and scalable model for developing shared memory parallel applications in C/C++ and Fortran.

OpenMP is intended for use in multi-core systems with shared memory following a fork-join model, as shown in Figure 2.1. The smallest unit of processing that can be scheduled is called a thread. In a typical use of OpenMP, the number of threads matches the number of cores available. The programming model always starts with a single thread, called the master thread, which executes the code sequentially until a parallel construct is found. At this point, the master thread takes care of the launch of all the parallel threads declared. These threads work in parallel until the statements belonging to the parallel region are executed. At this point an implicit synchronization is performed and all the threads but the master one are terminated. The execution continues sequentially until a new parallel construct is reached.

Different parallel constructs can be used, including implicit barriers of synchronization at the end of each one. For instance, the *single* construct, which ensures that a section of code is executed by only one of the running threads, the *sections* construct, which makes different portions of code be executed by different threads in parallel, or the *for loop* construct, which divides the iterations of a loop among the available threads.The OpenMP API makes it possible to set up different work-sharing scheduling strategies (static, dynamic, guided, etc.), with different chunk sizes, resulting in different data access patterns. An optimal choice of

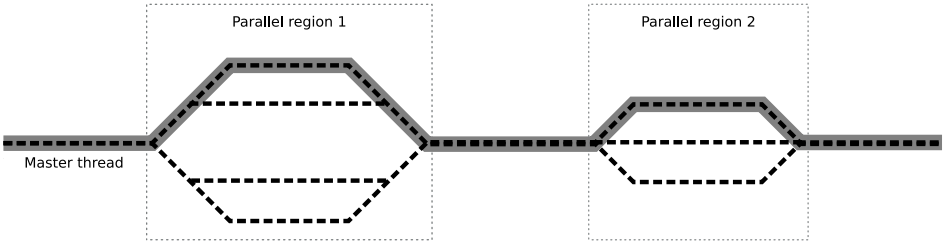


Figure 2.1: Fork-join parallel model as used by OpenMP.

the scheduling strategy will maximize the efficiency of the execution. To this end, two main factors must be considered: data locality and workload balancing among iterations. There also exist directives for critical or atomic sections, lock routines or explicit synchronizations. The parallel constructs can use different types of variable scopes (shared, private, firstprivate, etc.), affecting the synchronizations needed among threads.

The memory hierarchy management is transparent from an OpenMP point of view. However, some parameters that are up to the programer can severely affect the efficiency of the code. For instance, a chunk size multiple of the cache line size will avoid cache lines being shared among different threads, preventing the occurrence of *false sharing*; i.e., the invalidation of a cache line as a consequence of different threads writing in different positions of the same line.

2.1.2 Compute Unified Device Architecture

The CUDA platform is a hardware/software combination for the development of parallel applications to be executed in NVIDIA GPUs. Each GPU includes several Single Instruction Multiple Datas (SIMDs), which constitute the basic hardware unit of a GPU. Each Streaming Multiprocessor (SM) has fixed, limited resources, some of them exclusive and others shared among all the SMs in the GPU. The CUDA platform enables these GPUs to execute programs invoking parallel functions called kernels that execute across many parallel threads. Threads run in groups of 32, called warps, which are executed simultaneously. These threads are organized into blocks so that each thread executes an instance of the kernel following a SIMD programming model. The blocks are arranged in a grid that is mapped to a hierarchy of CUDA cores in the GPU so that each block of the grid is computed by an available SM. This grid can be one-dimensional, two-dimensional or even three-dimensional. It is up to the programer to

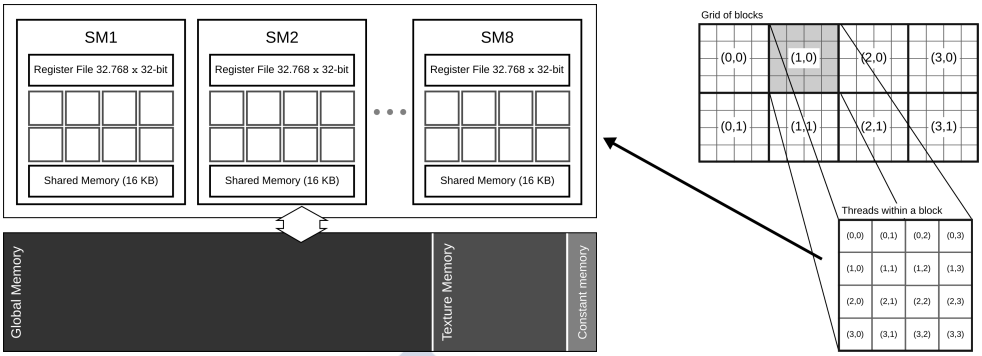


Figure 2.2: CUDA architecture overview (a), grid of blocks and block of threads scheduled to any of the available SMs (b).

select the best configuration for each kernel regarding the data to be processed. For instance, Figure 2.2(b) shows a 2D grid of size 4×2 including blocks of size 4×4.

Threads can access data from multiple memory spaces. First, each thread has a private local memory and registers. Each block of threads has a shared memory, visible exclusively to the threads within this block, and whose lifetime equals that of the block. As it is on-chip, shared memory is much faster than off-chip memory, but its lifetime prevents data sharing among thread blocks. Thus, the shared memory is especially useful when there is reuse of data among threads in the same block, avoiding reading the same data from the off-chip memory more than once.

Finally, all threads access the same global memory space, which is persistent across kernel launches by the same application. This is the main memory of the GPU, but it has the slowest access time. The number of accesses to global memory can be reduced by using coalesced access patterns. If threads in the same warp request consecutive and aligned values from the global memory, the device coalesces the load/store transactions minimizing them. In the best case, only one transaction will suffice).

The off-chip memory also includes two special memory spaces: the texture and constant memories. Data allocated in texture memory is automatically cached. Furthermore, this memory is optimized for 2D spatial locality, which makes it particularly useful when threads access values in regular spatial neighborhoods. Nevertheless, the available texture memory is highly reduced. The constant memory allows broadcasting to a full warp and its accesses are also

cached. So, the first time data in constant memory are accessed, only one transaction from global memory is needed, and the following times, only one read from the constant caches is required.

Data movement between on-chip and off-chip memory spaces is not automatic. It is up to the programer to select the best memory space for each task and explicitly perform the data movements needed.

There exist different generations of CUDA capable devices, each one with a particular codename (Tesla [available from 2008], Fermi [2010], Kepler [2012], Maxwell [2014], Pascal [2016], and, more recently, Volta [2018]). Each CUDA device has an associated compute capability, expressed as a version number (1.x, to 7.x, currently), which determines the resources available in its hardware and also the compatibility with different software operations. In this thesis, both the Kepler and Pascal architectures have been used. Besides the advance in compute capabilities, the major change between both architectures is related with the distribution of the cache and shared memories. The NVIDIA Kepler GPU architecture provides a two-level cache hierarchy including a configurable shared memory and L1 cache. There are 64 KB of on-chip memory for each SM, which can be configured in three different ways: as half each for the shared memory and the L1 cache, 48 KB of shared memory, and 16 KB of L1 cache or vice-versa. There is also a unified L2 cache of 1536 KB that is shared among all the SM units. Beginning with the Maxwell architecture, the cache hierarchy was changed: the SMs have their own dedicated pool of shared memory and a L1 cache that can also serve as a texture cache depending on the workload. A unified L1/texture cache acts as a coalescing buffer for memory accesses, gathering up the data requested by the threads of a warp prior to delivery of that data to the warp.

2.2 Techniques for efficiency in GPU

To achieve the best performance in terms of execution time, several GPU programming techniques have been applied in the implementations developed in this thesis. The objective of these techniques is to achieve the maximum exploitation of the available architecture both in terms of memory management and processing capabilities. Therefore, the techniques detailed below always aim to hide the memory latency in the access to data and to provide enough computational load to the maximum number of cores of the GPU at every moment of the execution time.

1. *Minimize the use of global memory and the number of data transfers between memory spaces.* The use of the global memory, with higher access time than the other memories, is minimized using in-place computations whenever possible. The device memory carries all the computations, so the costly data copies between the CPU and the GPU are reduced to the movement of inputs and outputs. Furthermore, data are stored in the memory so that the accesses will be coalesced whenever possible. It is also essential to minimize the data transfers between global and shared memory spaces, organizing the algorithm into computational blocks that can be executed independently in shared memory, minimizing communications among them.

2. *Profitable exploitation of the memory hierarchy.* Once data are stored in a block's shared memory, all the computations possible are performed, minimizing data movements to global memory. The configurable L1 cache/shared memory size of the Kepler architecture is set to maximize the occupancy and therefore, to reduce the execution times [199].

3. *Reduce the number of global synchronizations by computing asynchronous blocks.* This technique performs intra-block asynchronous updates reusing the shared memory and inter-block updates requiring global synchronizations. Several block-level asynchronous updates are performed before a global synchronization takes place to reduce synchronization overloads. The technique called block-asynchronous updating is explained in detail in [200] and an example is shown in Figure 2.3, showing the intra-block (left) and inter-block (right) updates for a grid of threads of size 6×6.

4. *Optimized data load pattern in shared memory.* The objective of this technique is to reduce the number of conditionals required to load data in shared memory, reducing divergence and improving the occupancy. Due to the data dependency required by some algorithms, each block needs to store in shared memory a 2D data block bigger than the thread number. For instance, when a block of threads stores the pixels corresponding to their indices in the image and also an apron. Figure 2.4 (a) shows an example considering a 9×10 thread-block, where 9 sections with different load patterns are needed to load all data (the central data, 4 apron borders, and 4 apron corners). With the optimized load pattern introduced in Figure 2.4 (b), only 4 different sections with different load patterns (the main data, 2 larger borders, and 1 corner) are needed to load all data.

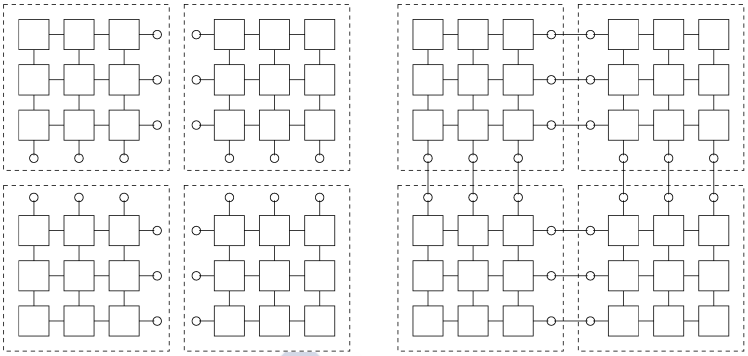


Figure 2.3: Example of block–asynchronous updating. Intra-block updating (left) and inter-block updating (right).

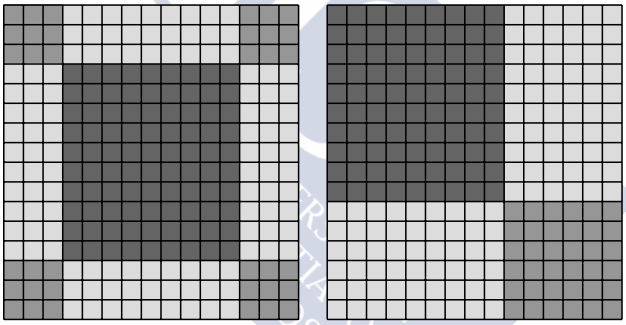


Figure 2.4: Shared Memory load pattern. Classical pattern (a), optimized pattern (b).

5. *Efficient computation of mathematical operations exploiting efficient libraries.* The cuBLAS [201] , MAGMA [202] and CULA [203] libraries are used to achieve the efficient computation of algebraic operations. All of them are optimized linear algebra libraries including efficient implementations of commonly used functions. For instance, the matrix by matrix or matrix by vector multiplications, the calculus of the pseudo-inverse of a matrix, or the calculus of the eigenvalue decomposition of a matrix. The NVIDIA Performance Primitives (NPP) library is used to efficiently perform the MM operations of erosion and dilation. The Caffe framework [204] is used for the SAE computations. It performs calls to CuDNN [205], CUBLAS and MAGMA libraries.

CuDNN is a GPU-accelerated library for deep neural networks providing implementations for standard routines.

6. *Fine-tuning of the kernel parameters.* A profile-driven optimization technique (using the NVIDIA Visual Profiler [206]) was used to determine the best kernel configuration for each kernel. Performance can be improved by reducing the block dimensionality whenever possible (mapping 2D data in 1D blocks, for instance). The block size is tuned for each kernel to achieve better efficiency and minimize execution time. The factors involved in the block size selection are the number of registers and the shared memory used by the program. In our applications, the selection of the block size depends on the GPU model, whereas the dimensions of the image are not relevant for the decision. Given that each SM can have a maximum number of active blocks, larger blocks are preferable. The selected block size is a multiple of 32 (the warp size) to avoid divergences among threads and thus maximize parallelism. A size of 512 or 1024 threads per block was selected for most of the kernels. This size is large enough to exploit the memory bandwidth of the device. Additionally, if a computation involves steps requiring different numbers of threads, it is split into consecutive kernels allowing the optimization of the resources and, therefore, maximizing efficiency.

7. *Reduction in the number of registers used.* The use or registers is forced to be the smallest possible with the *-maxrregcount* option when it affects the execution time.

8. *Avoid writing collisions.* When several threads in a kernel need to atomically write in the same structure, it is more efficient to perform independent partial results per block and combine these results in a second kernel. In this way, the writing collisions are minimized.

   This can be applied, for instance, in the calculation of a histogram, as shown in Figure 2.5. In the first kernel, each CUDA block calculates the histogram of a spatial portion of the input image. Then, the second kernel merges all the partial results into the global histogram.

9. *Appropriate selection of the data regions size.* Those operations where the computation requires data from neighborhood pixels use an extended data region including a border. The objective is to reduce the number of dependencies among blocks that could produce data transfers among blocks in shared memory.

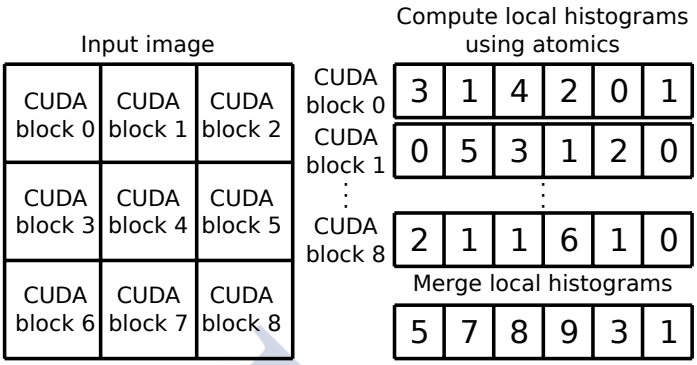


Figure 2.5: Example of histogram calculation using 9 blocks and 6 histogram levels.

2.3 Experimental setup

This section is devoted to the description of the environment available for the experiments of this thesis. The proposed schemes were evaluated both in CPU and GPU hardware under Linux using the gcc compiler for the OpenMP implementations and the nvcc compiler for the CUDA implementations. The optimization flag -O3 was used in all the cases. The following sections will describe the hardware used, the performance measures adopted to evaluate the schemes both in accuracy and execution time terms and, finally, the hyperspectral datasets where the different schemes were applied.

2.3.1 Hardware

Tables 2.1 and 2.2 summarize the main characteristics of the hardware used to carry out the experiments of this thesis. Two different commodity GPU architectures have been used during this thesis for validation of the proposed schemes. A NVIDIA GTX TITAN (Kepler) and a NVIDIA GTX TITAN X (Pascal) whose main characteristics were introduced in Section 2.1. A quad-core Intel Core i5-3470 at 3.20 GHz with 28 GB of Random Access Memory (RAM) was also used, for the validation of the CPU-based versions of the schemes.

Additionally, the FT2 cluster at CESGA [207] has been used to carry out experiments in a multi-GPU system. In this cluster, each computing node includes an Intel Xeon E5-2680 v3 processor at 2.50 GHz with 128 GB of RAM. A three-level cache hierarchy is available, with 30 MB of shared L3. Four of the nodes also include a NVIDIA Tesla K80 device. The K80 has a dual-GPU design with Kepler architecture including 2496 cores at 0.82 GHz and

CPU hardware	# of cores	Core clock (MHz)	RAM (GB)	L1 (KB)	L2 (KB)	L3 (MB)
Intel Core i5-3470	4	3200	24	64	256	6
Intel Xeon E5-2680 v3	12	2500	128	32	256	30

Table 2.1: CPU hardware specifications

GPU hardware	GTX TITAN	GTX TITAN X	Tesla K80
Compute capability	3.5	6.1	3.7
Number of GPUs	1	1	2
Streaming Multiprocessors	14	28	26
CUDA cores per SM	192	128	192
Threads per block	1024	1024	1024
Threads per SM	2048	2048	2048
Blocks per SM	16	32	16
Active warps per SM	64	64	64
Registers per SM	65536	65536	131072
Registers per thread	255	255	255
Device memory	6 GB	12 GB	2×12 GB
Shared memory	16/32/48 KB	96 KB	80/96/112 KB
L1 cache	48 KB	48 KB	48 KB
L2 cache	1536 KB	3072 KB	1536 KB
CUDA version	5.5	8.0	5.5

Table 2.2: GPU hardware specifications for a GTX TITAN (Kepler architecture), a GTX TITAN X (Pascal architecture), and a Tesla K80 (Kepler architecture).

12 GB of memory per GPU. Each GPU also incorporates an L1 / L2 cache hierarchy of 1.5 MB. The L1 cache is available only for the threads running in the same SM, the L2 cache is shared among all the threads.

2.3.2 Performance measures

Different performance measures have been used to evaluate the quality of the schemes proposed in this thesis. The following paragraphs detail these measures grouped in three different categories. The first group of metrics is devoted to assessing the quality of the proposed methods in terms of (classification or CD) accuracy. The second group is devoted to assessing the performance of the execution of those methods in terms of execution time, speedup, and hardware exploitation. Finally, a metric to evaluate the noise level of the images used in the

experiments is introduced, along with different types of noise used to evaluate the robustness of the proposed schemes in the presence of noise.

Accuracy assessment

The classification techniques produce classification maps where a unique label is assigned to each pixel of the input hyperspectral image. A quantitative evaluation of the achieved accuracies was performed comparing the achieved classification maps with the reference data available for each image. Different metrics [208, 8] have been used to perform this evaluation during this thesis. Let C_i be the class i, C_{ij} the number of pixels classified to class j referenced as the class i, and K the total number of classes.

- *The Class-Specific Accuracy (CS)*, or producer's accuracy, is the percentage of correctly classified pixels for a given class i.

$$CS_i = \frac{C_{ii}}{\sum_j^K C_{ij}} \times 100\% \tag{2.1}$$

- *The Overall Accuracy (OA)*, which represents the total percentage of correctly classified pixels.

$$OA = \frac{\sum_i^K C_{ii}}{\sum_{ij}^K C_{ij}} \times 100\% \tag{2.2}$$

- *The Average Accuracy (AA)*, representing the average percentage of correctly classified pixels for each class. It is the mean of all the class-specific accuracies.

$$AA = \frac{\sum_i^K CS_i}{K} \times 100\% \tag{2.3}$$

- *The Kappa coefficient (k)*, defined as the percentage of agreement corrected by the amount of agreement that could by expected purely by chance.

$$k = 1 - \frac{1 - p_o}{1 - p_e} \times 100\% \tag{2.4}$$

  where p_o is the observed agreement and p_e represents the hypothetical probability of chance agreement.

Regarding the change detection techniques, there are some techniques known as binary change detection techniques, where the result obtained is a binary change detection map where

each pixel can be assigned to only two different labels: *change* or *no change* [102]. The quality of these maps is evaluated through the use of four different metrics.

- *Correctly classified pixels*, the total number of pixels that were correctly assigned to the correspondent label. It can be displayed both in absolute terms and as a percentage of the total amount of pixels in the image.
- *Missed Alarms (MAs)*, i.e., the number of pixels that should be labeled as change but were labeled as no change. This is expressed in absolute terms.
- *False Alarms (FAs)*, the number of pixels that were incorrectly labeled as change. It is delivered as an absolute value.
- *Total error*, defined as the sum of missed alarms and false alarms. It can be expressed as an absolute value or as a percentage of the total amount of pixels in the image.

Execution time and speedup

The implementations of the schemes proposed in this thesis are also evaluated in terms of time efficiency. This is done through the measurement of the execution time (in seconds) required for the scheme to be computed. The execution times achieved by the GPU implementations are also compared against optimized multi-threaded OpenMP implementations obtaining a speedup measure.

The speedup is a metric for relative performance showing the number of times than the optimized version is faster than the baseline one; i.e., a different version used as a reference for comparison purposes. This baseline version can be a non-optimized version of the implementation (for example, a single thread version of the implementation), a version implemented in a less efficient programming language (such as MATLAB) or in a different hardware platform (for instance, a multicore processor implementation as a baseline to a GPU implementation). The speedup is calculated as the ratio between the baseline and optimized versions execution times.

$$Speedup = \frac{T_{baseline}}{T_{optimized}}. \tag{2.5}$$

The execution times detailed in this thesis represent the elapsed time between the start and the end of the computation, without taking into account the time required for I/O operations. This time has been excluded because each implementation is part of a general scheme

where the different stages are concatenated in a pipeline processing, so, during the pipeline execution, the output of one stage is kept in RAM to be used as the input of the following stage.

Occupancy

The achieved occupancy is a measure of the hardware exploitation of the GPUs [209]. Regarding the different configurations made when launching a kernel, the occupancy may change, with the consequent change in the performance of the kernel.

The occupancy is defined as the ratio between the number of active warps per SM and the maximum number of possible active warps. Therefore, it can be expressed as a percentage. There are three possible limiting factors in the occupancy achieved by a kernel:

- *The number of registers used.* If a kernel requires too many registers per thread, the number of concurrent warps per SM will be smaller than the theoretical one as there are not enough available registers for all the threads. For instance, with 63 registers per thread in the Kepler architecture (the maximum allowed by the architecture), the maximum number of concurrent threads will be 1024 instead of the theoretical 2048.

- *The amount of shared memory required.* Each SM has a finite amount of shared memory among all its threads. If the amount of shared memory required by each thread exceeds a certain value, the number of concurrent warps will be reduced.

- *The block size selected.* Each architecture has a maximum number of threads per SM, for instance this number is 2048 in the Kepler architecture. If an excessively large block size is selected, this number will be reached before all the theoretically possible concurrent warps are launched.

When launching a kernel, in general, the highest number of concurrent warps is preferable, as this maximizes the occupancy and allows us to hide memory access latencies.

Noise level assessment

The processing of hyperspectral image capture may result in the appearance of noise in the original spectral signatures. For this reason, two types of noise are used in this thesis to evaluate the robustness of the proposed schemes:

- **Additive White Gaussian Noise (AWGN).** A noise model used to resemble the effect of random processes that occur in nature. Its name summarizes its main characteristics. It is additive because the noise signal is added to the original signal. It has uniform power across the different frequencies, just as the color white is composed of all frequencies in the visible spectrum. Finally, it is Gaussian as the probability distribution of the noise samples is Gaussian.

- **Speckle noise.** A granular multiplicative noise, such as those that appear in the capturing process of Satellite Aperture Radar (SAR) images. This kind of noise comes from random fluctuations in the return signal from an object that is no bigger than a single image-processing element. It increases the mean gray level of a local area [210].

Different levels of noise are applied to the original spectral signatures by selecting the variance of the noisy signal to be introduced (σ). The Peak Signal-to-Noise Ratio (PSNR) obtained after the application of the noise is used as a measure of the level of degradation of the input images. The PSNR represents the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. It is expressed in terms of decibels (db), on a logarithmic scale, to deal with the wide dynamic range present in many signals.

2.3.3 Datasets

Several hyperspectral datasets have been used during the realization of this thesis. The next sections are devoted to describe them in detail.

The datasets considered are provided by four different sensors:

- *The ROSIS-03 sensor* [211]. The Reflective Optics System Imaging Spectrometer provides 115 spectral bands covering a spectral wavelength from 0.43 µm to 0.86 µm and a spatial resolution of 1.3 meters per pixel.

- *The AVIRIS sensor* [212]. The Airborne Visible-Infrared Imaging Spectrometer grants a spectral coverage between the 0.4 µm and 2.4 µm with a total amount of 224 spectral bands. The sensor provides different spatial resolutions between 1 and 20 meters per pixel, depending on the flight parameters.

- *The Hyperion sensor* [213]. This sensor captures 220 spectral bands covering the range from 0.4 µm to 2.5 µm. All the images from this sensor share a spatial resolution of 30 meters per pixel.

- *The Thematic Mapper sensor* from the Landsat-5 [214]. This sensor captures 6 spectral bands (plus a thermal band) covering the range from 0.45 µm to 2.35 µm. The first 3 bands correspond to the RGB spectra and the other 3 correspond to near and short-wave infrared. The images from this sensor have a spatial resolution of 30 meters per pixel.

One image datasets

This section includes the datasets containing a single image. The datasets of this kind used in this thesis are the Pavia University, Indian Pines, and Salinas datasets obtained from the Computational Intelligence Group from the Basque University website [215].

Pavia University

The Pavia University dataset consist of an image acquired by the ROSIS-03 sensor over the urban area near the Engineering School of the University of Pavia, Italy (N 45°12'13.176", E 9°8'11.108"). Its spatial dimensions are 610 $\times$ 340 pixels and it contains 103 spectral bands. The spatial resolution of this image is 1.3 meters per pixel. The reduced number of bands as compared to the total number acquired by the sensor is explained by the fact that the 12 noisiest bands were removed from the original data. A color representation of the dataset is shown in Figure 2.6, which also displays the available reference data for this image, including 9 different classes. Table 2.3 details the distribution of the pixels among the existing classes.

Indian Pines

The Indian Pines dataset was acquired by the AVIRIS sensor covering an area in the north-western of Indiana. The image includes agricultural and forested areas with a spatial resolution of 20 meters per pixel. It is a perfect square of 145 pixels per side and includes 220 spectral bands, as four bands covering the water absorption channels were removed. Figure 2.7 shows the color representation and the available reference data for this image, including 16 different classes. Table 2.4 details the distribution of the pixels among the existing classes. This dataset is a challenge for classification tasks as it involves a considerable number of

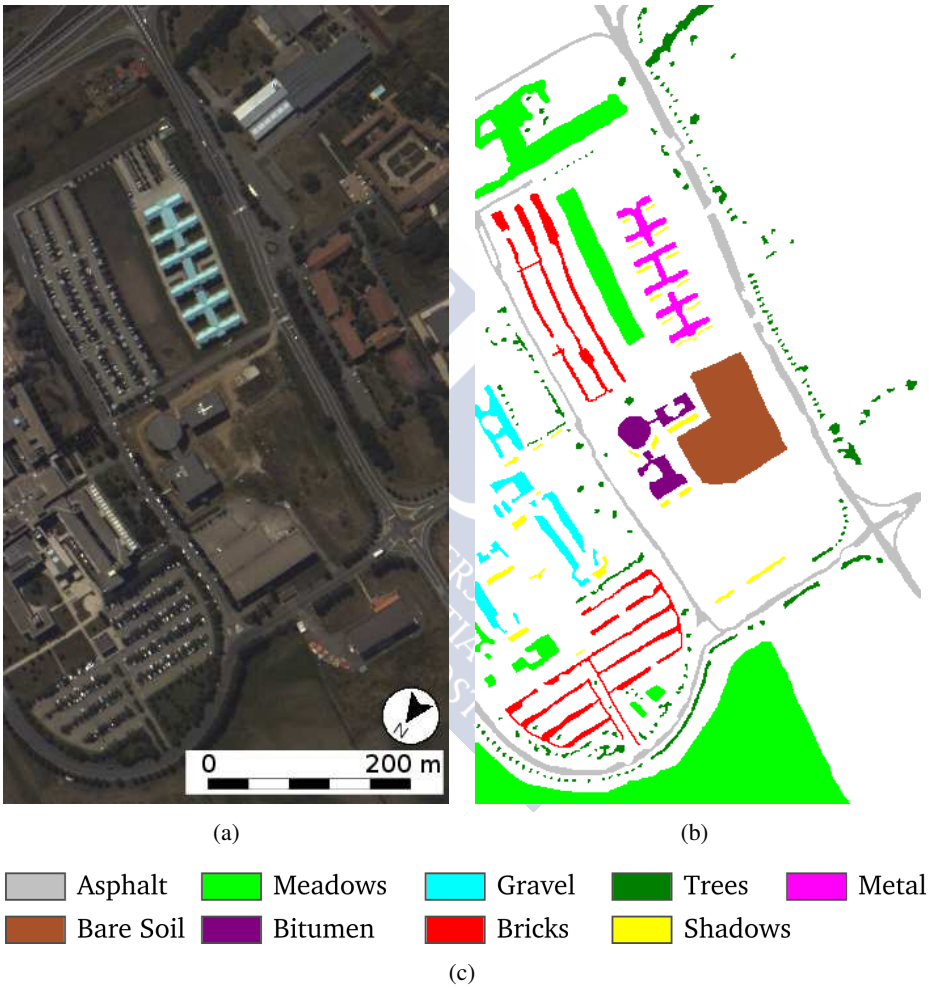
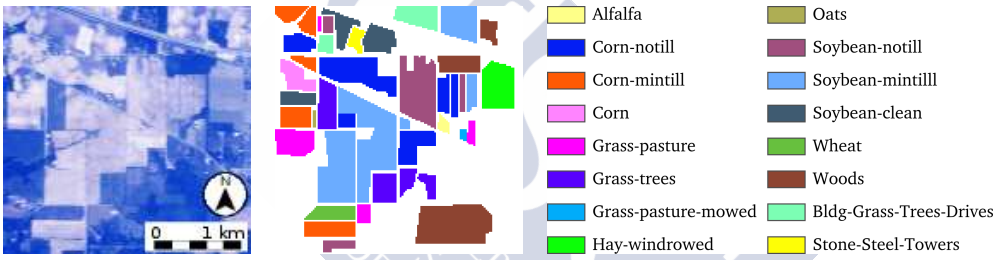


Figure 2.6: Pavia University dataset. Color representation (a), reference data map (b), and classes identified (c).

Class	Name	Number of samples
1	Asphalt	6631
2	Meadows	18649
3	Gravel	2099
4	Trees	3064
5	Metal	1345
6	Bare soil	5029
7	Bitumen	1330
8	Bricks	3682
9	Shadows	947
Total		42776

Table 2.3: Classes and distribution of samples for the Pavia University dataset.



(a) (b) (c)

Figure 2.7: Indian Pines dataset. Color representation (a), reference data map (b), and classes identified (c).

similar classes (sometimes the same crop in different growth states) and it includes a reduced amount of samples for some of them.

Salinas

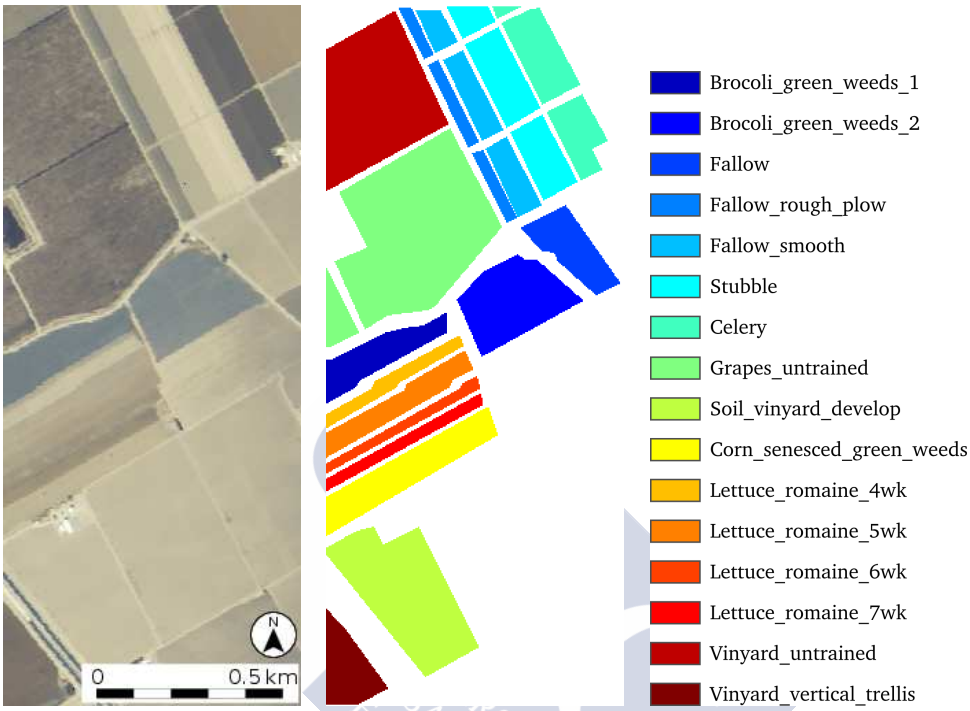
The Indian Pines dataset was also acquired by the AVIRIS sensor. It shows an area over the Salinas Valley in California. The spatial resolution of this image is 3.7 meters per pixel. Its spatial dimensions are 512 $\times$ 217 pixels and all the 224 spectral bands from the acquired data are retained. The color representation and reference data for this image can be seen in Figure 2.8. 16 different classes, whose distribution is shown in Table 2.5, are considered.

Class	Name	Number of samples
1	Alfalfa	54
2	Corn-notill	1434
3	Corn-mintill	834
4	Corn	234
5	Grass/pasture	497
6	Grass-trees	747
7	Grass/mowed	26
8	Hay-windrowed	489
9	Oats	20
10	Soybean-notill	968
11	Soybean-mintill	2468
12	Soybean-clean	614
13	Wheat	212
14	Woods	1294
15	Bld-Grs-Trs-Drs	380
16	Stone-Steel	95
Total		10366

Table 2.4: Classes and distribution of samples for the Indian Pines dataset.

Class	Name	Number of samples
1	Broccoli green weeds 1	2009
2	Broccoli green weeds 2	3726
3	Fallow	1976
4	Fallow rough plow	1394
5	Fallow smooth	2678
6	Stubble	3959
7	Celery	3579
8	Grapes untrained	11271
9	Soil vinyard develop	6203
10	Corn senesced green weeds	3278
11	Lettuce romaine 4 weeks	1068
12	Lettuce romaine 5 weeks	1927
13	Lettuce romaine 6 weeks	916
14	Lettuce romaine 7 weeks	1070
15	Vinyard untrained	7268
16	Vinyard vertical trellis	1807
Total		54129

Table 2.5: Classes and distribution of samples for the Salinas dataset.



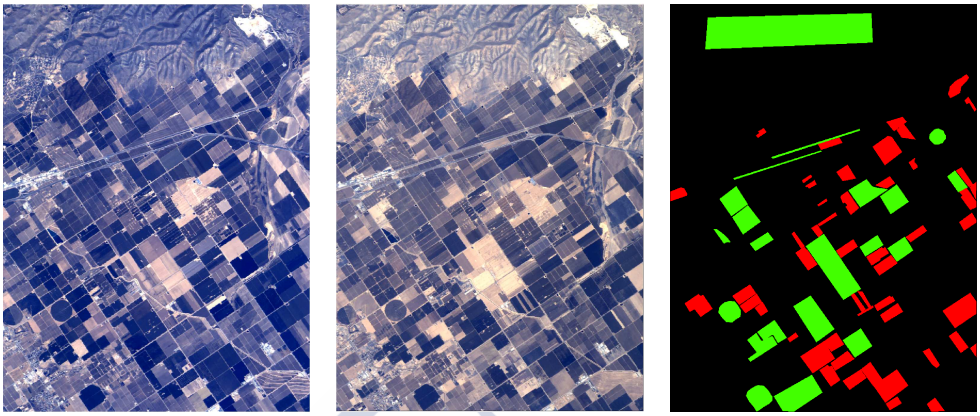
(a) (b) (c)

Figure 2.8: Salinas dataset. Color representation (a), reference data map (b), and classes identified (c).

Multitemporal datasets

This section is devoted to the introduction of those datasets containing two multitemporal images that have been used for the evaluation of the change detection techniques developed in this thesis. Some of these datasets include a binary reference data, that only allows to perform a binary change detection discriminating between changed and unchanged areas. Others contain reference data with several classes that allows a classification of the changes detected to be performed.

For the multitemporal datasets, a pre-processing where the images of the dataset are co-registered is required, so that each pixel in both images corresponds to the same spatial location. This process was made by using HypeRvieW [132], a desktop tool that performs a



(a) (b) (c)

Figure 2.9: Santa Barbara dataset. Color representation in 2013 (a) and 2014 (b), and reference change map (green represents unchanged areas and red represents changed areas) (c).

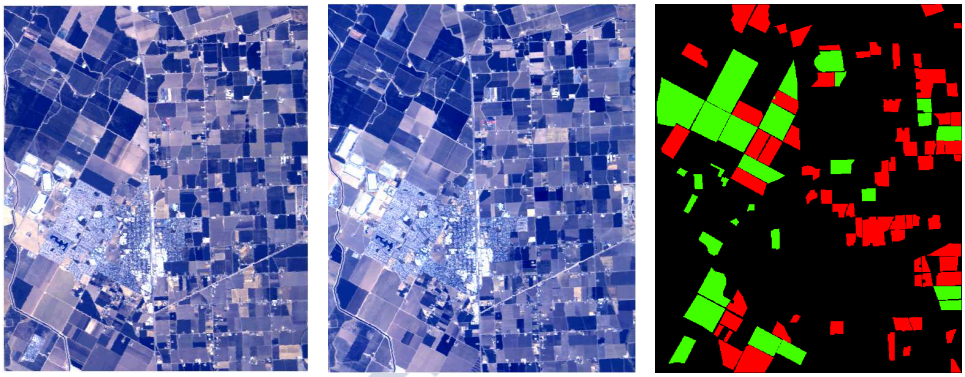
Class	Number of samples	Percentage
Changed pixels	52134	7.16%
Unchanged pixels	80418	11.94%
Unlabeled pixels	595608	81.80%
Total	728160	100.00%

Table 2.6: Classes and distribution of samples for the Santa Barbara dataset.

registration based on the computation of the multilayer fractional Fourier transform developed by our group[137].

Santa Barbara

The Santa Barbara dataset was captured by the AVIRIS sensor in the years 2013 and 2014. It covers a semi-urban area in the Santa Barbara region, California (N 35°18'14.3", W 118°50'22.6"). The spatial resolution of both images in this dataset is 15.2 meters per pixel. The dimensions of the images are 984 $\times$ 740 pixels $\times$ 224 spectral bands. The color representation and binary reference data available for this dataset can be seen in Figure 2.9. Table 2.6 summarizes the distribution of changed and unchanged pixels between the multitemporal images.



(a) (b) (c)

Figure 2.10: Bay Area dataset. Color representation in 2013 (a) and 2014 (b), and reference change map (green represents unchanged areas and red represents changed areas) (c).

Class	Number of samples	Percentage
Changed pixels	38425	12.81%
Unchanged pixels	34211	11.40%
Unlabeled pixels	227364	75.79%
Total	300000	100.00%

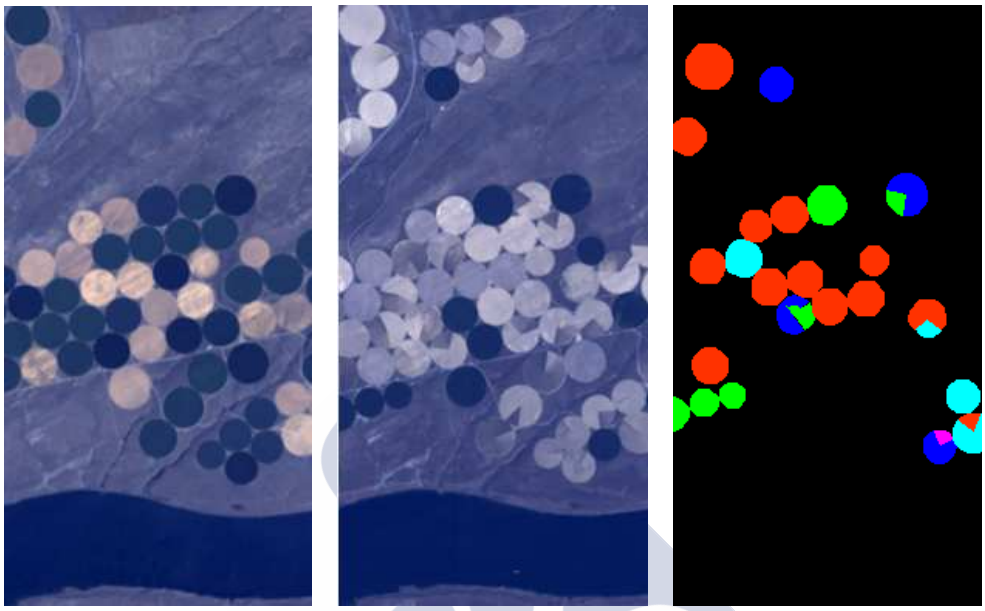
Table 2.7: Classes and distribution of samples for the Bay Area dataset.

Bay Area

The Bay Area dataset, acquired by the AVIRIS sensor in the years 2013 and 2015, shows a semi-urban area in Patterson, California (N 37°28'05.8", W 121°08'45.1"). The dataset has a spatial resolution of 16.9 meters per pixel. The size of the images is 600 $\times$ 500 pixels $\times$ 224 spectral bands. The color representation and binary reference data for this dataset can be seen in Figure 2.10. Table 2.7 summarizes the distribution of changed and unchanged pixels between the multitemporal images.

Hermiston

The Hermiston dataset was acquired by the HYPERION sensor over the Hermiston City area, Oregon (N 45°58'49.4", W 119°13'09.6"). It includes two images corresponding to the years 2004 and 2007. The images were downloaded from the United States Geological Survey



(a) (b) (c)

Figure 2.11: Hermiston dataset. Color representation in 2004 (a) and 2007 (b), and reference change map including 5 types of changes (c).

(USGS) website. The dimensions of the images are 390 $\times$ 200 pixels $\times$ 242 spectral bands, with a spatial resolution of 30 meters per pixel. The dataset displays a rural area were several circular crop areas are placed. Figure 2.11 shows the color representation and reference data available for this dataset. Five types of changes can be found in the crops between the years 2004 and 2007, whose densities are detailed in Table 2.8.

Different reference maps with different number of types of change can be found in the literature for this dataset regarding the focus of CD [14, 216]. Some of the types of change presented in these reference maps are related with disconnected pixels or noise. As the objective in this thesis is to focus in object-based change detection, only those types of change related with consistent areas or objects have been considered in the reference data used.

Class	Name	Number of samples
1	Type 1	5558
2	Type 2	1331
3	Type 3	79
4	Type 4	1557
5	Type 5	1461
Total		9986

Table 2.8: Classes and distribution of samples for the Hermiston dataset.

(a) (b) (c)

Figure 2.12: Sardinia dataset. Color representation in 1995 (a) and 1996 (b), and reference change map including 3 types of changes (c).

Sardinia

The Sardinia dataset was acquired by the Thematic Mapper sensor mounted on the Landsat-5 satellite [214]. This dataset was kindly provided by Sicong Liu from the University of Trento. This is not a hyperspectral dataset but was included in the experiments in order to compare the methods proposed in this thesis with other state of the art CD methods from the literature [217, 193]. It covers a region on the island of Sardinia in Italy, including the Lake Mulargia (N 39°37'39.6", E 9°14'13.4"). It includes two images corresponding to the years 1995 and 1996. The dimensions of the images are 300 $\times$ 412 pixels $\times$ 6 bands. The spatial resolution is 30 meters per pixel. Figure 2.12 shows the color representation and reference data available for this dataset. Three types of changes are considered in this dataset. An enlargement of an open quarry between the two branches of the lake, a simulated burned area and the enlargement of the lake surface due to the increase of the water volume of the Lake Mulargia. The magnitude of these changes in number of pixels is detailed in Table 2.9.

Class	Name	Number of samples
1	Enlargement of the quarry	214
2	Simulated burned area	2414
3	Increase of the lake surface	7480
Total		10180

Table 2.9: Classes and distribution of samples for the Sardinia dataset as described in [193].

Synthetic datasets

Currently, there is a lack of multitemporal hyperspectral datasets with an appropriate and robust reference data available. Reference data of this kind are mandatory for performing an accurate assessment of the proposed schemes for CD. For this reason, a tool for automatically generating synthetic multitemporal datasets with complete and reliable reference maps has been developed.

This tool, called *HyperGenerator*, is developed in MATLAB and includes a GUI in order to simplify its use, as can be seen in Figure 2.13. It includes a database with the spectral signatures of different materials extracted from the USGS repository [133]. The user can choose which of these materials is to be added to the synthetic image. A graph is displayed as support to compare the spectral signatures of the different materials.

Once the materials to be included in the dataset are determined, there are two possible ways to generate the dataset: On the one hand, the user can use the parameters on the right side of the GUI to generate a random pattern based on squared and circled objects with different sizes and positions on an image of the desired size. This may be used as a simulation of a crop field pattern. on the other hand, a segmentation map can be loaded and used as the pattern to be filled with the different previously selected materials.

In order to achieve a more challenging and realistic scenario, both approaches allow the inclusion of noise in the original spectral signatures of the materials in three different ways:

- Firstly, the spectral signature used for each region is created as a mixture of two materials chosen randomly from the selected ones. In order to guarantee the correctness of the reference data, half the weight in the mixture will always be assigned to the principal material, the other half will be randomly distributed.

(a)

Figure 2.13: HyperGenerator GUI.

- Then, for each region, a deviation from the created spectral signature is randomly added, taking the noise parameter selected by the user as a base. This may be used to simulate different lightning conditions in different regions of the image with the same materials.
- Finally, in order to avoid perfectly delimited regions with uniform spectral signatures, speckle noise is added to each pixel of the generated image using the noise parameter selected by the user.

The tool provides the option of simulating a multitemporal dataset by creating a second image called *change image*. This image will share all the configuration with the first one, but, for each region, there will be a probability of selecting a different material from the one used in the first image. This probability is also a parameter that can be modified at the bottom of the GUI.

After execution, the tool provides the following files (the ones related with the change image are only created when applicable):

1. The generated image.

2. The generated image free of noise.
3. The reference data for the image.
4. The change image.
5. The change image free of noise.
6. The reference data for the change image.
7. The reference data of changes between the two images.
8. The fusion of the two images as a stacked image.
9. The fusion of the two images as a difference image.
10. The parameters used for the generation of the image.

Synthetic 1 dataset

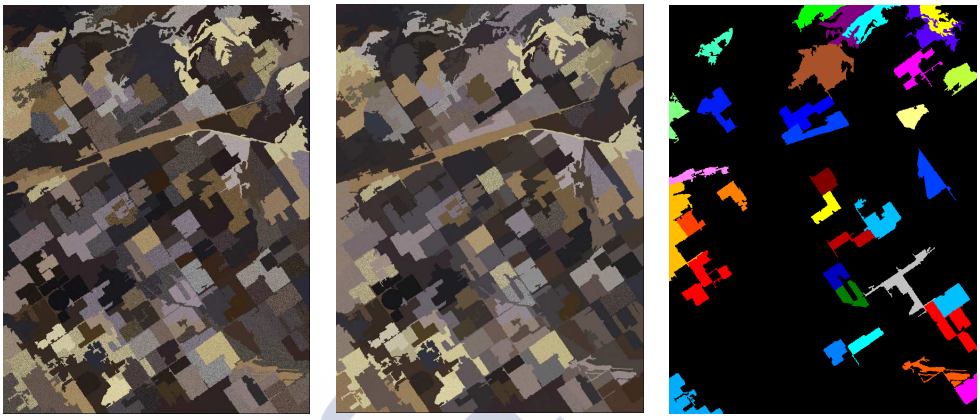
The Synthetic 1 dataset was generated with HyperGenerator using a segmentation map with 191 different regions corresponding to a real capture of the Santa Barbara region, California and 16 different materials. Table 2.10 shows the materials used in the creation of the dataset. It includes two images whose dimensions are 1024×769 pixels $\times$ 239 spectral bands. Figure 2.14 shows the color representation an reference data available for this dataset. The two images present 32 different types of changes between them, whose densities are detailed in Table 2.11.

2.4 Discussion

The need for computationally efficient CD methods has been discussed in this chapter. For some applications, it is critical to obtain time efficient schemes that can be projected into commodity hardware architectures.

In order to achieve this goal, efficient multicore and GPU techniques are proposed in this thesis and the OpenMP and CUDA programming models are selected for the implementation of the different schemes that will be introduced in the remaining chapters.

The main features of the OpenMP and CUDA parallel programming models were revised in the first section of this chapter. Both of them are adequate paradigms for obtaining parallel



(a) (b) (c)

Figure 2.14: Synthetic 1 dataset. False color representation of the first image (a) and the second image (b), and reference change map including 32 different types of changes (c).

Class	Name	Number of samples image 1	Number of samples image 2
1	Asphalt Light Gray	38610	32759
2	Red Brick	61506	45677
3	Painted Aluminum	66223	72568
4	Burn Area	49123	38752
5	Concrete	42838	73336
6	Rangeland	54626	69002
7	Asphalt Dark Gray	58698	51243
8	Cedar Shake	39517	32717
9	Cinder Block	54568	67830
10	Coated Steel Girder	26282	26145
11	Galvanized Sheet Metal	39214	38499
12	Plastic Gray Bag	44758	44750
13	Roofing Felt	52619	51558
14	Wood Beam	67120	63043
15	Cheat Grass	74267	62090
16	Grass Golden Dry	17487	17487
Total		787456	787456

Table 2.10: Materials used in the Synthetic 1 dataset.

Class	Original class	New class	Number of samples
1	Asphalt Light Gray	Painted Aluminum	2441
2	Asphalt Light Gray	Concrete	4761
3	Asphalt Light Gray	Cinder Block	9072
4	Red Brick	Concrete	2250
5	Red Brick	Rangeland	9086
6	Red Brick	Galvanized Sheet Metal	4493
7	Painted Aluminum	Asphalt Light Gray	2758
8	Painted Aluminum	Cedar Shake	2288
9	Painted Aluminum	Wood Beam	3587
10	Burn Area	Asphalt Light Gray	2436
11	Burn Area	Concrete	7935
12	Concrete	Asphalt Light Gray	2479
13	Asphalt Dark Gray	Coated Steel Girder	2408
14	Asphalt Dark Gray	Galvanized Sheet Metal	5917
15	Asphalt Dark Gray	Roofing Felt	2759
16	Cedar Shake	Painted Aluminum	2714
17	Cedar Shake	Rangeland	5290
18	Cedar Shake	Asphalt Dark Gray	3629
19	Cinder Block	Concrete	2363
20	Cinder Block	Plastic Gray Bag	2571
21	Coated Steel Girder	Cedar Shake	2545
22	Galvanized Sheet Metal	Concrete	11125
23	Plastic Gray Bag	Cinder Block	5915
24	Roofing Felt	Painted Aluminum	5780
25	Roofing Felt	Plastic Gray Bag	3336
26	Wood Beam	Asphalt Light Gray	2750
27	Wood Beam	Concrete	4543
28	Wood Beam	Cinder Block	3209
29	Wood Beam	Cheat Grass	2415
30	Cheat Grass	Painted Aluminum	4043
31	Cheat Grass	Roofing Felt	5296
32	Cheat Grass	Wood Beam	5253
Total			137447

Table 2.11: Classes and distribution of samples for the Synthetic 1 dataset.

implementations of the hyperspectral processing schemes developed in this thesis. OpenMP focuses on multicore CPU architectures, whereas CUDA is devoted to the programming of NVIDIA GPUs.

The different programming techniques that must be taken into account to achieve the optimal implementation of a scheme when using CUDA have also been introduced in this chapter.

Then, the experimental setup used to validate the proposed schemes in both OpenMP and CUDA platforms has been detailed. Several well-known performance metrics have been introduced and they will be used in the remaining chapters of this thesis to assess the quality of the proposed schemes both in accuracy, execution time and hardware exploitation terms.

The last sections of this chapter are devoted to the introduction of the main characteristics of the hyperspectral datasets that will be used in the next chapters to validate the proposed schemes for the efficient processing of hyperspectral images.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 3

EFFICIENT ELM-BASED CLASSIFICATION SCHEMES ON GPU

The inclusion of a final classification stage is common in CD processing tasks. In the CD schemes proposed in this thesis, this last stage is performed by methods based on supervised classification. ELM is one of the classifiers selected owing to its low computational cost, which makes it appealing for real-time applications. The first GPU implementation of the ELM classifier is introduced in this thesis. Different classification methods based on ELM are also proposed and analyzed with the aim of increasing the accuracy of the classification maps obtained.

In particular, the first section of this chapter describes in detail the CUDA implementation of the ELM classifier for hyperspectral images developed in this thesis. The ELM algorithm fundamentals were introduced in Section 1.6.2. The efficient computation of the *Moore-Penrose* inverse of a matrix in GPU is highlighted, as it is one of the most relevant steps in the processing. Finally, an approach based on the use of ensembles to achieve more robust classification results is also introduced.

The next section is devoted to the introduction of spectral-spatial classification schemes based on the ELM classifier. The schemes consist of a spatial processing branch and a spectral one. Two different schemes relying on segmentation in the spatial branch of the processing are introduced. The first one is a scheme based on watershed segmentation. A RCMG is also included to obtain an optimal single-band input for the watershed segmentation. A Majority Vote (MV) process is included to join the spectral and the spatial branches of the scheme.

The second scheme uses RQS segmentation instead of watershed. These segmentation techniques allows us to work with the entire dimensionality of the hyperspectral images. Different similarity measures introduced in Section 1.2 are implemented in the RQS algorithm in order to explore their suitability for the comparison of spectral signatures. A spatial regularization process based on the closest neighborhood pixels is also introduced in both schemes.

The third section of this chapter shows accuracy and execution time results for all the schemes proposed in this chapter. The experiments have been conducted over some of the datasets presented in Chapter 2. The results are also compared with similar state of the art approaches for each scheme.

The last section is reserved for a discussion of the results and the exhibition of the main conclusions.

3.1 Efficient ELM in GPU

This section describes the GPU implementation of the ELM algorithm introduced in 1.6.2. The algorithm has three main phases: pre-processing, training and test. The pseudocode in Figure 3.1 shows the algorithm that has been implemented, including host and device codes. The kernels executed in GPU are placed between <> symbols. The pseudocode also includes the GM and SM acronyms to indicate kernels executed in global memory and shared memory, respectively.

First, all the data are scaled in the range $[0:1]$ (line 1 in the pseudocode). The pixel-vectors of the considered dataset are randomly distributed between two non-overlapping sets: training and test. These two sets are stored in matrices $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$, respectively, where each row represents a sample and each column a spectral band (lines 2 and 3 in the pseudocode). Data matrices are converted to column major format in order to be used by the MAGMA library (line 4 in the pseudocode). The ground truth labels are also split into two target matrices: $\mathbf{T}_{train}$, which is used during the learning phase, and $\mathbf{T}_{test}$, which will be used to check the accuracy results. Finally, the training and test target matrices are processed so that each row represents a sample and each column a class, where a value of 1 indicates membership to a class, otherwise a value of -1 is assigned (line 5 in the pseudocode). The pre-processing phase is computed in CPU and the results are stored in the global memory of the GPU. All the remaining steps will be computed in GPU.

GPU ELM.
Input: hyperspectral dataset $\mathbf{X}$*, label set* $\mathbf{T}$*, L: number of neurons in the hidden layer, C: number of classes, N: number of samples.*
Output: classification map.

1: Scale hyperspectral dataset in [0:1] ▷ Pre-processing phase (1)
2: Randomly choose the training points
3: Take the remaining points for test
4: Store data in column major order matrices $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$
5: Process target matrices $\mathbf{T}_{train}$ and $\mathbf{T}_{test}$
▷ Training phase (2)
6: <Generate random weights ($\mathbf{a}_i$) and biases (b_i), i=1,...,L> ▷ GM
7: <Transpose the weight matrix and multiply by $\mathbf{X}_{train}$ > ▷ SM
8: <Add the biases> ▷ SM
9: <Apply activation (sigmoid) function to obtain $\mathbf{H}$ > ▷ GM
10: <Calculate $\mathbf{H}^{\dagger}$ as the Moore-Penrose pseudoinverse of $\mathbf{H}$ > ▷ SM
11: <Calculate output as $\beta = \mathbf{H}^{\dagger} \times \mathbf{T}_{train}$ > ▷ SM
▷ Test phase (3)
12: <Transpose weight matrix and multiply by $\mathbf{X}_{test}$ > ▷ SM
13: <Add the biases> ▷ SM
14: <Apply activation (sigmoid) function to obtain $\mathbf{H}$ > ▷ GM
15: <Calculate output as $\mathbf{Y} = (\mathbf{H})^T \times \beta$ > ▷ SM
16: <Calculate the estimated output label> ▷ GM
▷ GM: Global Memory, SM: Shared Memory

Figure 3.1: Pseudocode for the ELM algorithm (stage (I) in Figure 3.4).

The training phase starts by generating random weights and biases (line 6 in the pseudocode). The weights matrix must have values within the range $[-1:1]$, where the dimensions thereof are the number of neurons in the hidden layer and the number of input neurons (equal to the spectral band number). The bias vector will have values in the range $[0:1]$ and a size equal to the number of neurons in the hidden layer. These two matrices are then stored in the global memory before calculating the hidden layer output matrix $\mathbf{H}$. This matrix is calculated in three steps: first, the transpose of the weights matrix is multiplied by the training matrix, then the bias vector is added and, finally, an activation function is applied to each element of the matrix (lines 7- 9 in the pseudocode corresponding to (1.28)). The sigmoid function ($g(x) = 1/(1+e^{-x})$) is applied as activation function through a CUDA kernel, with each thread operating over a single element of the matrix. The final training step consists in calculating the output weights multiplying the transpose of the pseudoinverse of matrix $\mathbf{H}$ by

the training targets matrix (line 11 in the pseudocode corresponding to (1.32)). The steps for efficiently calculating the pseudoinverse of a matrix are detailed in 3.1.1.

The test phase (lines 12 to 16 in the pseudocode) starts by multiplying the transpose of the previously generated weights matrix by the data matrix $\mathbf{X}_{test}$. Then, the bias vector is added and the activation function is applied, as in the training phase, to obtain the test hidden layer matrix $\mathbf{H}$. These operations are analogous to those of lines 7 to 9 in the training phase. Afterwards, the output matrix $\mathbf{Y}$ is calculated by multiplying $\mathbf{H}^T$ by the output weights matrix β obtained in the training phase. Finally, the estimated output label $\mathbf{T}_i$ is calculated as the class c that maximizes $\mathbf{Y}_i$ for each sample,

$$\hat{\mathbf{T}}_i = \underset{c=1,\ldots,C}{\arg\max}\, \mathbf{Y}_{i,c}. \tag{3.1}$$

3.1.1 Efficient GPU computation of the Moore-Penrose inverse of a matrix

The calculation of the pseudoinverse of a matrix is the most computationally costly operation in the training phase of the ELM algorithm (line 11 in the pseudocode). For this thesis, it is implemented as described in [179] to run it efficiently in CUDA using the MAGMA library.

First, the dimensions of the input $\mathbf{H}$ matrix are checked in order to ascertain whether the number of rows is smaller than the number of columns. If this is the case, the MAGMA library is used to compute a matrix $\mathbf{A}$ as the multiplication of the original matrix by its transpose, otherwise $\mathbf{A}$ is computed as the multiplication of the transpose matrix by the original matrix. This operation ensures that $\mathbf{A}$ is a symmetric positive definite matrix.

The next step consists in calculating the Cholesky factorization of $\mathbf{A}$ with the MAGMA *dpotrf* function and then applying a kernel to nullify the upper triangle of the factorized matrix obtaining matrix $\mathbf{L}$. Unlike the other steps, this last kernel is launched in global memory.

Afterwards, an $\mathbf{M}$ matrix is calculated by multiplying $\mathbf{L}^T$ by $\mathbf{L}$ and then computing the inverse of this matrix using the MAGMA *dgetrf* and *dgetri* functions.

Finally, once all of these matrices have been calculated, the inverse of the original $\mathbf{H}$ matrix is obtained through a set of consecutive multiplications computed using the MAGMA *dgemm* function. If the dimension check of the $\mathbf{H}$ matrix at start resulted in that the row number is lower than the column number, $\mathbf{H}^\dagger$ is computed as

$$\mathbf{H}^\dagger = \mathbf{H}^T \times \mathbf{L} \times \mathbf{M} \times \mathbf{M} \times \mathbf{L}^T,$$

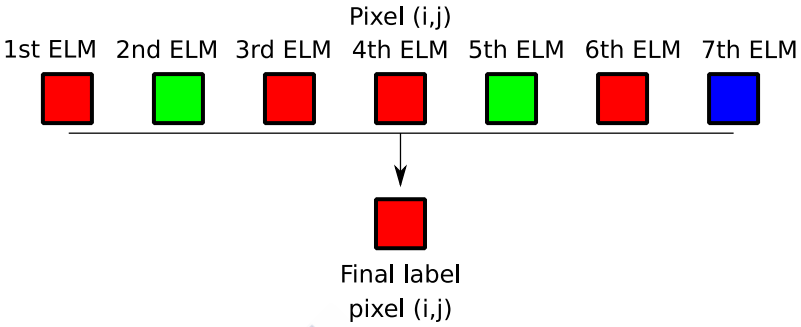


Figure 3.2: Example of democratic MV algorithm applied to n classification maps.

otherwise it is computed as

$$\mathbf{H}^{\dagger} = \mathbf{L} \times \mathbf{M} \times \mathbf{M} \times \mathbf{L}^{T} \times \mathbf{H}^{T}.$$

3.1.2 Voting-based ELM scheme

An approach based on the use of ensembles that is called Voting-based Extreme Learning Machine (V-ELM) is used in this thesis in order to improve the accuracy of a single ELM [50, 56]. An ensemble comprises a number of independent ELMs with the same number of hidden nodes and the same activation function in each hidden node, whose results are later combined in a final classification. The individual ELMs are trained with the same dataset and the learning parameters of each ELM are randomly initialized independently.

MV is the simplest method to implement among all the combination methods as it assumes no prior knowledge of the behavior of the individual classifiers and it requires no training [218]. A democratic MV method where each classifier vote counts equal to the others and the final decision for each sample is the most repeated vote for the sample is used in this thesis. An example of the application of MV to 7 independent ELMs is displayed in Figure 3.2.

The pseudocode for the MV phase is shown in Figure 3.3. For each sample, a vector $\mathbf{S}_i$ is used to store the vote of the k ELM and then, the final label ($\mathbf{m}vT_i$) is calculated as the most repeated output value produced by the different ELMs for the sample.

Therefore, after all the ELM computations are carried out, an N by C matrix (where N is the number of pixels and C the number of classes in the dataset) with the vote of each ELM for every pixel of the image is obtained. In the MV on GPU, a CUDA kernel is launched and

1: **for each** sample i ($i = 1,\ldots,N$) **do**
2: **for each** ELM k ($k = 1,\ldots,K$) **do**
3: $\mathbf{S}_i(\hat{\mathbf{T}}_i^k) = \mathbf{S}_i(\hat{\mathbf{T}}_i^k) + 1$
4: **end for**
5: $\mathbf{m}vT_i = \arg\max_{c=1,\ldots,C} \mathbf{S}_{i,c}$
6: **end for**

Figure 3.3: Voting phase of the V-ELM algorithm.

computed in global memory where each thread computes the MV for one pixel of the image; i.e., each thread operates over a row of the matrix.

3.2 Spectral-spatial classification schemes based on ELM on GPU

In this section the spectral-spatial classification schemes developed to improve the classification accuracies of the ELM classifier are introduced. The first approach is based on watershed segmentation for the spatial branch of the scheme, while the second one relies on the use of the RQS segmentation.

3.2.1 Watershed-based spectral-spatial classification scheme

This section introduces a spectral-spatial classification scheme following the structure displayed in Figure 3.4. On the one hand, a classification map is produced through a pixel-wise classifier. On the other hand, a segmentation map is created from a one-band image generated through a gradient calculation. Finally, spectral and spatial results are combined through a democratic MV performed among the pixels belonging to the same segmentation region. A post-processing technique based on the use of information of the closest neighborhood of each pixel is also considered to improve the final classification map.

In the spectral-spatial classification a pipeline processing is applied in GPU to combine the different stages of the algorithm reducing data movement through global memory.

RCMG GPU implementation

The gradient calculation is divided into two steps whose pseudocode is introduced in Figure 3.5 [219]. First, for all the pixel vectors, the threads within the same block cooperate to calculate the distances of the set χ. For each region, data are stored in row-major order for

(I) **Raw ELM pixel-wise classifier.**

Input: hyperspectral dataset.

Output: classification map.

(1) Pre-processing stage.

(2) Training stage.

(3) Test stage.

(II) **Gradient calculation.**

Input: hyperspectral dataset.

Output: single band image.

(1) Calculate the distances among pixels.

(2) Compute the gradient.

(III) **Segmentation.**

Input: single band image from gradient calculation.

Output: segmentation map.

(1) Initialization stage.

(2) Updating stage.

(IV) **Combination of maps.**

Input: ELM classification map, segmentation map.

Output: spectral-spatial classification map.

(1) Voting stage.

(2) Winner stage.

(3) Updating stage.

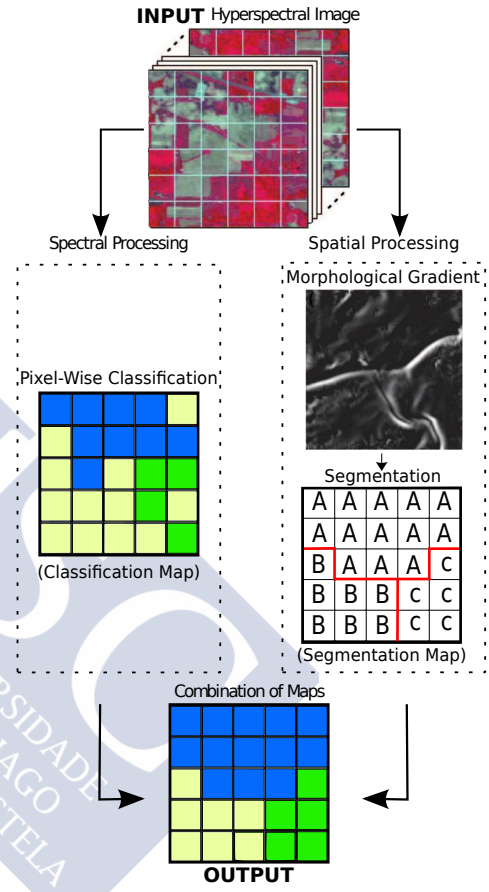


Figure 3.4: Spectral-spatial classification scheme. (I) spectral stage, (II-III) spatial stage, (IV) combination stage.

each band. The kernel is configured to work in two-dimensional thread blocks. Threads within a block process a region of each spectral band in a sequentially processed loop through all the bands. In each iteration, data corresponding to a new band are loaded in shared memory (line 2 in the pseudocode of Figure 3.5) and the partial results are computed and stored (loop in lines 3-5). When the outer loop is finished (lines 1-7), all the distances for each pixel are available in shared memory.

GPU RCMG kernel.
Input: hyperspectral dataset $\mathbf{X}$.
Output: single band image. ▷ Distances step (1)

1: **for each** band i of $\mathbf{X}$ **do**
2: Load band i in shared memory
3: **for each** pixel x in band i **do**
4: Compute and accumulate the corresponding term in the EDs $\mathrm{D}_{y,z} \,|\, y,z \in \mathscr{X}$, where $\mathscr{X}$ is the set of neighbors of pixel x
5: **end for**
6: Synchronize threads within the block
7: **end for**

▷ Gradient step (2)

8: Compute $\mathrm{CMG}(\mathbf{X}) = \max_{y,z \in \mathscr{X}} \mathrm{D}_{y,z}$
9: Find the pair of pixels $r = (y,z) \,|\, \mathrm{D}_{y,z} = \mathrm{CMG}(\mathbf{X})$
10: Compute $\mathrm{RCMG}(\mathbf{X}) = \max_{y,z \in \mathscr{X}-r} \mathrm{D}_{y,z}$
11: Write RCMG($\mathbf{X}$) to global memory

Figure 3.5: Pseudocode for the RCMG kernel (shared memory) corresponding to stage (II) in Figure3.4.

In the second step, each thread finds the maximum of the distances of its set χ (line 8) and the corresponding pair of pixels which generated the maximum (line 9). Once the two farthest pixel vectors are identified and removed, each thread computes the RCMG with the remaining distances (line 10) and writes the result in the global memory of the device (line 11).

GPU implementation of the CA-Watershed

The segmentation map is calculated through the CA-Watershed algorithm [163]. The input data to this algorithm are directly the output obtained by the RCMG. The CA-Watershed can be asynchronously implemented, which is up to five times faster than the CUDA synchronous implementation [220]. The implementation includes intra-block asynchronous updates computed in shared memory and inter-block synchronous updates computed in global memory to efficiently deal with the fact that the CA algorithm needs data from the neighbors of each pixel. This implementation has the advantage of reusing information within a block to efficiently exploit the shared and cache memories of the device.

```
   GPU Asynchronous CA-Watershed.
   Input: single band image from RCMG.
   Output: segmentation map.                                  ▷ Initialization stage (1)
1: <Initialize CA data (labels, distances and states)>        ▷ GM
                                                              ▷ Updating stage (2)
2: while CA is not stable do                                  ▷ inter–block updating
3:     <Asynchronous updating of the CA>                      ▷ intra–block updating (SM)
4:     Global synchronization among blocks
5: end while
                                          ▷ GM: Global Memory, SM: Shared Memory
```

Figure 3.6: Pseudocode for the host and device asynchronous CA-Watershed scheme corresponding to the stage (III) in Figure 3.4.

```
   GPU Asynchronous CA-Watershed updating kernel.
   Input: image CA data (labels, distances and states).
   Output: updated image CA data.                             ▷ intra–block updating
1: Load CA data in shared memory
2: while CA is not stable at block level do
3:     Update labels, distances and states within the block
4:     Local synchronization among threads
5: end while
6: Write CA data to global memory
```

Figure 3.7: Pseudocode for the asynchronous updating kernel corresponding to the intra-block updating stage (line 3 in Figure 3.6).

The algorithm used (a pseudocode of the algorithm is shown in Figs. 3.6 and 3.7), as further described in [163], comprises two kernels implementing the initialization and updating stages of the CA-Watershed, which are configured to work in two-dimensional thread blocks operating in same-size pixel regions of the image. The updating stage is an iterative process that lasts until no modifications are made to the available data inside the region (lines 2-5 in Figure 3.6). This implementation generates a segmentation map where the pixels are connected so that every pixel in the same region has the same label.

MV GPU implementation

In the spectral-spatial scheme, the MV processes the pixels within each segmented region. In this implementation, a region can be assigned to different thread blocks as long as all the pixels belonging to the same region are connected.

GPU MV kernels.
Input: ELM classification map, watershed segmentation map.
Output: spectral-spatial classification map.

1: <Count number of watershed regions> ▷ GM
▷ Voting stage (1)
2: <Count number of pixels of each class in each region> ▷ GM
▷ Winner stage (2)
3: <Obtain the winner class for each region> ▷ GM
▷ Updating stage (3)
4: <Update the pixels inside each region to the winner class> ▷ GM
▷ GM: Global Memory

Figure 3.8: Pseudocode for the MV corresponding to stage (IV) in Figure 3.4.

One MV process per watershed region is needed. As the number of regions is unknown beforehand, they are counted before the voting step (line 1 in Figure 3.8). Afterwards, a two dimensional data structure is allocated in global memory, whose dimensions are the number of watershed regions and the number of spectral classes.

The MV algorithm consists of three steps: *voting*, *winner* and *updating* that are listed in Figure 3.8. In the first step (line 2), for each watershed region the number of pixels for each class is counted. In the voting kernel, the voting is done by atomic operations where each thread adds one vote to the appropriate class in the region as more than one thread can vote in the same region to the same class with no predictable order. Then, the winner step finds the most repeated class inside each region (line 3). Finally, the updating kernel assigns all the pixels inside a region to the winning classes producing a new spectral-spatial classification map (line 4). Each step is performed by a separate kernel configured to work in one dimensional blocks of threads. In the first-step and third-step kernels (lines 2 and 4), one thread operates on one pixel, while in the second one, each thread operates on the information collected for one region of the segmentation map.

Spatial regularization on GPU

This post-processing technique introduces spatial information form the closest neighborhood of a pixel to the classification map. It is particularly helpful, as shown in Figure 3.9, to remove disconnected points, which are commonly misclassified, and to smooth the edges among classes, solving the classification accuracy problem frequently present in these areas. Once the classification map is generated, an iterative process starts in which each pixel checks

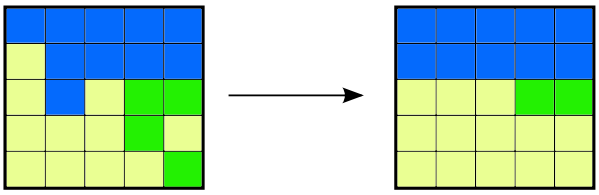


Figure 3.9: Example of spatial regularization applied to a classification map.

the class label of its neighbors and all the pixels do so simultaneously. If more than half of the neighbors share the same label, and this label is different from that of the pixel, the pixel updates its own class. This process lasts until stability is reached; i.e., until there are no changes between two consecutive iterations in any of the pixels [53]. This is computed by a single kernel that is executed as many times as it takes to reach stability. The kernel is launched in 2D blocks of threads covering the entire classification map in each call. Each pixel of the image is processed by a thread.

3.2.2 RQS-based spectral-spatial classification scheme

A new scheme for spectral-spatial classification, replacing the RCMG and watershed processing of Figure 3.4 with an RQS-based segmentation is introduced in this section. Unlike the watershed segmentation, this segmentation allows the full dimensionality of the image to be maintained throughout the entire processing. The RQS implementation in GPU is based on the one presented in [64] for RGB images.

Figure 3.10 shows the pseudocode of the RQS segmentation on GPU. Different multidimensional distances can be used to calculate the distance in both the density computation and the neighbor-connectivity computation kernels. In this work the euclidean, SID and SID-SAM distances are compared. The segmentation starts with the density computation kernel, where one thread is used to calculate the density of each pixel as the accumulation of the distances between the considered pixel and all the neighbors in a window whose size is determined by the parameter σ. After this, in the neighbor-connectivity computation kernel, that once again works with as many threads as pixels in the image, all the pixels of the image are linked with those inside a neighborhood window whose size is determined by the parameter τ. If a pixel inside the window has a greater density than the one considered and the distance between this pair of pixels is the smallest inside the window, this pixel becomes the parent of

GPU RQS segmentation.
Input:Hyperspectral dataset $\boldsymbol{X}$. *Parameters* σ *and* τ.
Output: segmentation map.

```
1:  <Density computation>                                                    ▷ GM
2:  for each pixel i in X do
3:      for each pixel j in neighborhood window σ of i do
4:          Density[i] += distance(i, j)
5:      end for
6:  end for
7:  <Neighbors connectivity computation>                                     ▷ GM
8:  for each pixel i in X do
9:      for each pixel j in less than τ away do
10:         if Density[j] > Density[i] and distance(i, j) is smallest among all j then
11:             Density[i] = distance(i, j)
12:             Parent[i] = j
13:         end if
14:     end for
15:     Label i regarding connectivity
16: end for                                                  ▷ GM: Global Memory
```

Figure 3.10: Pseudocode for the RQS segmentation.

the considered pixel, forming a tree of connected pixels. It there is no neighbor that satisfies these conditions, the pixel becomes a root node in the tree.

3.3 Results

This section is devoted to the analysis of the results of the different ELM schemes proposed in this chapter. All the GPU results presented in this section are for the NVIDIA GTX TITAN architecture presented in Section 2.3.1. The schemes are evaluated in the Pavia University, Indian Pines and Salinas datasets introduced in Section 2.3.3. The evaluation is performed in terms of the accuracy of the classifiers as well as in terms of execution times and speedups of the different implementations.

The number of training samples for the ELM is 200 per class, or half the number of samples in the class if there are not enough samples. These samples are randomly chosen and all the remaining samples are used for testing. The number of hidden layer neurons employed is 500 for Pavia University, 950 for Indian Pines, and 350 for Salinas in all the cases [53].

	Pavia University			Indian Pines			Salinas		
	OA	AA	k	OA	AA	k	OA	AA	k
SVM [63]	81.01	88.25	75.86	78.76	69.66	75.75	81.25	–	–
ELM	86.75	89.55	82.61	**80.72**	**85.48**	**77.70**	**91.55**	95.97	**90.55**
	(0.71)	(0.26)	(0.87)	**(0.58)**	**(1.31)**	**(0.64)**	**(0.27)**	(0.13)	**(0.29)**
V-ELM-1	**90.32**	**91.81**	**85.77**	79.84	90.62	72.40	90.74	**96.22**	89.18
	(0.31)	**(0.20)**	**(0.43)**	(0.83)	(2.82)	(1.08)	(0.11)	**(0.15)**	(0.13)

Table 3.1: Classification accuracy as percentages (and standard deviations between brackets). The ELMs contained 500, 950, and 350 nodes in the hidden layer for the Pavia University, Indian Pines, and Sailnas datasets, respectively.

3.3.1 ELM-based classification results

In this section the results of applying only ELM for classification are presented. Two different GPU optimized configurations using ELM are compared:

1. A single ELM trained with 200 samples for each class (ELM).
2. A V-ELM comprising 8 ELMs trained with 200 samples for each class for each one of the ELMs, so that each ELM is the same as in the first configuration (V-ELM-1).

Table 3.1 shows accuracy results for the images in terms of OA, AA, and k. The best results are highlighted in bold in the table. The first thing to highlight is that both configurations obtain acceptable accuracy results, being slightly better than the SVM for all three datasets.

For the Pavia University image, the V-ELM-1 configuration clearly outperforms the ELM configuration in terms of accuracy results, while for the Indian Pines and Salinas images, both configurations obtain similar results, with the ELM configuration being only slightly better. Finally, it is worth noting than the standard deviation values remain low in all the cases. Figure 3.11, shows the ground truths and false color classification maps obtained by the ELM algorithm.

The performance results in terms of execution times and speedups calculated over the OpenMP multicore implementations are detailed in Table 3.2. It has been observed in the experiments that the V-ELM-1 configuration provides more stable accuracy results than a single ELM at the cost of slightly higher execution times.

The speedups of the ELM as compared to the SVM in both, the CPU and the GPU architectures, are displayed in Table 3.3. For the three images, the single ELM configuration

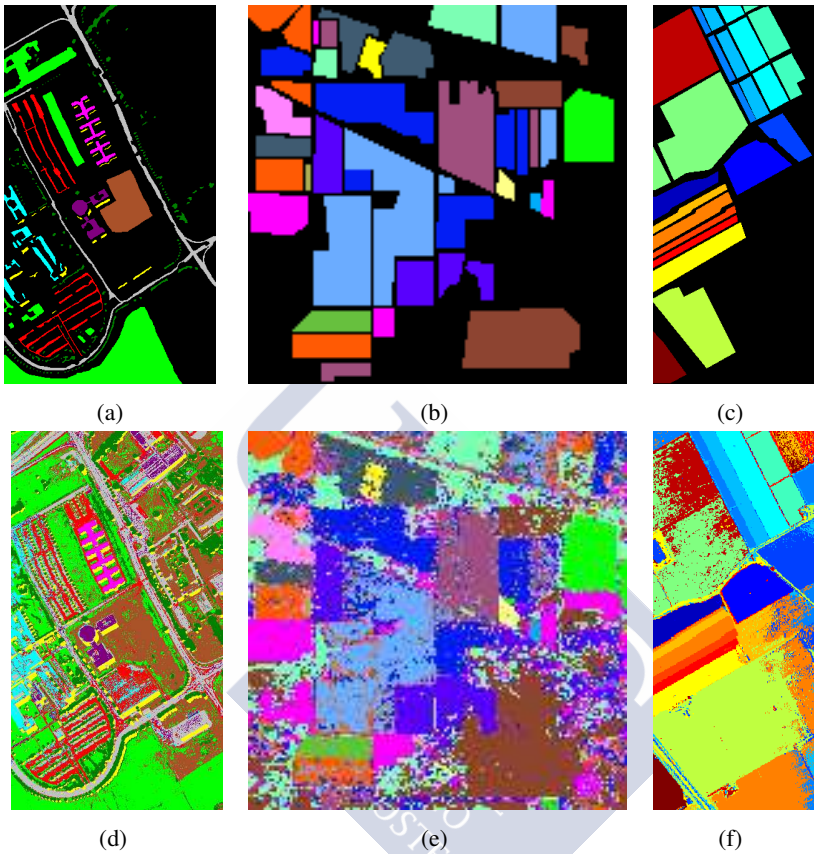


Figure 3.11: Ground truths (a,b,c) and classification maps (d,e,f) for the Pavia University (a,d), Indian Pines (b,d), and Salinas (c,f) images, respectively.

is faster than SVM, achieving, for the Pavia University image, a speedup of 8.8× in CPU and 8.4× in GPU. The V-ELM-1 configuration is more adequate when the dataset size is large because, otherwise (as in the case of Indian Pines), there are not enough samples to take advantage of the voting to improve accuracy results.

Summarizing, on the one hand, for the remote sensing datasets considered the raw ELM algorithm is significantly faster than SVM and, on the other hand, the V-ELM-1 algorithm always approaches or outperforms the raw ELM accuracy although it requires a higher number of training samples. This final example is a good configuration when the execution time is the priority.

	SVM	ELM	V-ELM-1
Pavia University			
OpenMP CPU	20.5876s	2.3304s	18.9022s
CUDA GPU	2.5834s	**0.3063s**	2.4501s
Speedup	8.0×	7.6×	7.7×
Indian Pines			
OpenMP CPU	3.0084s	1.1653s	9.6903s
CUDA GPU	0.7652s	**0.3096s**	2.6058s
Speedup	3.9×	3.8×	3.7×
Salinas			
OpenMP CPU	5.6018s	1.1023s	8.7055s
CUDA GPU	0.8708s	**0.3439s**	3.0114s
Speedup	6.4×	3.2×	2.9×

Table 3.2: Execution time and speedups for different supervised classifiers.

	Pavia University		Indian Pines		Salinas	
	CPU	GPU	CPU	GPU	CPU	GPU
ELM	**8.8**×	**8.4**×	**2.6**×	**2.5**×	**5.1**×	**2.5**×
V-ELM-1	1.1×	1.1×	0.3×	0.3×	0.6×	0.3×

Table 3.3: Speedups of the ELM implementations against the SVM ones.

3.3.2 Classification results for the spectral-spatial schemes

In this section, the experimental results obtained by the application of the spectral-spatial classification schemes based on segmentation with watershed and RQS introduced in Figure 3.4 are shown. The impact of spatial regularization over an ELM classification map is also studied.

The classification accuracy of the proposed method is compared to results published in the literature, as the pixel-wise spectral classification by a SVM, spatial regularization (SVM+reg) [221], and the similar spectral-spatial schemes based on segmentation (SVM+wat) [63] and (SVM+EM) [221]. In addition, the combination of segmentation and spatial regularization (SVM+EM+reg) [221] is also included in the results. SVM+wat denotes that the segmentation map of the spectral-spatial scheme is created by watershed, and SVM+EM the same but using EM [196] for segmentation by partitional clustering. In all the schemes, the spectral-spatial information is combined by the MV algorithm within each segmented region. The spatial

	Pavia University			Indian Pines			Salinas		
	OA	AA	k	OA	AA	k	OA	AA	k
SVM [63]	81.01	88.25	75.86	78.76	69.66	75.75	81.25	–	–
SVM+reg [221]	84.27	90.89	79.90	88.58	77.27	86.93	–	–	–
SVM+wat con(8) [63]	85.42	91.31	81.30	92.48	77.26	91.39	–	–	–
SVM+EM [221]	93.59	94.39	91.48	87.25	70.34	85.43	–	–	–
SVM+EM+reg [221]	94.68	95.21	92.02	88.83	71.90	87.24	–	–	–
V-ELM [55]	89.18	–	–	70.08	–	–	**93.88**	–	–
ELM	86.75	89.55	82.61	80.72	85.48	77.70	91.55	95.97	90.55
	(0.71)	(0.26)	(0.87)	(0.58)	(1.31)	(0.64)	(0.27)	(0.13)	(0.29)
ELM+reg	95.13	95.51	93.50	91.04	92.32	89.54	93.56	**97.02**	92.78
	(0.65)	(0.40)	(0.86)	(0.82)	(1.25)	(0.94)	(0.28)	**(0.15)**	(0.32)
ELM+wat con(4)	93.84	94.05	91.79	88.73	90.76	86.90	92.91	96.15	92.06
	(0.83)	(0.47)	(1.08)	(0.67)	(1.55)	(0.76)	(0.25)	(0.18)	(0.27)
ELM+wat con(8)	95.09	95.14	93.44	91.41	93.91	89.98	93.31	96.52	92.51
	(0.71)	(0.47)	(0.93)	(0.97)	(1.32)	(1.12)	(0.33)	(0.17)	(0.37)
ELM+reg+wat con(4)	95.37	95.00	93.81	90.90	91.47	89.38	93.46	96.48	92.67)
	(0.67)	(0.47)	(0.88)	(0.96)	(1.63)	(1.10)	(0.31)	(0.18)	(0.35)
ELM+reg+wat con(8)	95.65	95.52	94.18	**92.67**	94.29	**91.43**	93.70	96.78	**92.95**
	(0.77)	(0.52)	(1.02)	**(1.08)**	(1.22)	**(1.24)**	(0.35)	(0.16)	**(0.39)**
V-ELM-1+reg+wat con(8)	**96.66**	**95.92**	**95.00**	90.41	**95.35**	86.21	92.43	96.75	91.15
	(0.28)	**(0.29)**	**(0.42)**	(1.06)	**(1.83)**	(1.43)	(0.31)	(0.16)	(0.36)
ELM+reg+RQS (Euclidean)	95.05	94.61	93.37	91.99	91.17	90.64	93.61	96.38	92.84
	(0.90)	(0.51)	(1.18)	(0.89)	(1.45)	(1.02)	(0.48)	(0.21)	(0.54)
ELM+reg+RQS (SID)	95.39	95.23	93.84	92.01	80.32	90.63	93.63	96.75	92.86
	(0.85)	(0.60)	(1.12)	(1.07)	(0.87)	(1.24)	(0.39)	(0.18)	(0.44)
ELM+reg+RQS (SID-SAM)	95.04	94.81	93.37	92.38	92.53	91.09	93.45	96.50	92.67
	(0.77)	(0.46)	(1.01)	(0.97)	(1.98)	(1.11)	(0.39)	(0.22)	(0.44)
V-ELM-1+reg+RQS (Euclidean)	96.28	95.10	94.42	90.02	93.95	85.64	92.48	96.70	91.21
	(0.26)	(0.31)	(0.38)	(1.24)	(2.88)	(1.68)	(0.24)	(0.17)	(0.29)

Table 3.4: Classification accuracy as percentages (and standard deviations between brackets). 'reg' indicates that the pixel-wise classifier was spatially regularized. 'wat' indicates spatial processing by watershed. 'con(x)' indicates connectivity of 'x' neighbors.

regularization of SVM+reg and SVM+EM+reg is performed using Chamfer connectivities of eight and sixteen neighbors [221]. Results for another work based on ensembles of ELM and a similar spectral-spatial scheme (V-ELM) are also included [55].

Table 3.4 shows the accuracy obtained using the developed classification schemes (best results for each dataset in bold). Results from the literature obtained using a SVM pixel-wise classifier are also included for comparison purposes.

As it can be observed in Table 3.4, the ELM-based strategy obtains better results for the three datasets. Therefore, it can be stated that, in terms of accuracy, ELM is suitable for replacing SVM in this spectral-spatial scheme. The connectivity of 8 neighbors, as expected, improves on the results of the 4 neighbors one. Table 3.4 also shows that the spatially regularized configurations always give better results. It is worth noting that the spectral-spatial scheme using a spatially regularized ELM (ELM+reg+wat con(8)) requires less computation time that the one based on ensembles of ELM (V-ELM-1+reg+wat con(8))) but achieves better results.

From Table 3.4 it can also be observed that the best scheme based on watershed segmentation achieves slightly better results than the ones based on RQS. For the case of the schemes based on RQS (ELM+reg+RQS(Euclidean), ELM+reg+RQS(SID), and ELM+reg+RQS(SID-SAM)), the three considered distances achieve very similar classification results, with the one based on SID distances being slightly better in two of the three datasets. The application of ensembles to this scheme achieves results similar to those of the watershed-based scheme.

Table 3.5 details the class-specific accuracies of the ELM scheme (ELM+reg+wat con(8)) for the Pavia University and Indian Pines datasets. Results for the same scheme replacing ELM with SVM from the literature are included for comparison purposes. The same comparison is made for the ELM and SVM without spatial information. It can be seen that the schemes including spatial information outperform the accuracy of the pixel-wise schemes for all classes but one. It can also be seen that the proposed scheme with ELM outperforms the SVM for most of the classes in both datasets.

Figures 3.12, 3.13, and 3.14 show the results of the spectral-spatial scheme using a spatially regularized ELM (ELM+reg+wat con(8)) for the Pavia University, Indian Pines and Salinas datasets, respectively. For these three figures, it can be seen that the application of a raw ELM classifier (a) results in noisy areas with misclassified pixels. A good deal of the misclassified pixels are removed when the spatial regularization based on pixels on the closest neighborhood is applied (b) to the original classification map. The quality of the classification map is further improved when the spectral classification is combined with the spatial segmentation (c and d) achieved after the computation of the RCMG of the input image (c) and the watershed segmentation of this RCMG (d), obtaining a final spectral-spatial classification map through the use of MV inside each watershed region (e).

The performance results of the ELM-based spectral-spatial classification scheme, in terms of execution time and speedup, are detailed in Table 3.6. For all the datasets, the GPU imple-

Class	Pavia University				Indian Pines			
			Watershed				Watershed	
	SVM [63]	ELM	SVM [63]	ELM+reg	SVM [63]	ELM	SVM [63]	ELM+reg
1	84.93	80.86	**93.64**	88.48	32.65	78.89	44.90	**96.00**
2	70.79	92.78	75.09	**96.99**	74.59	81.48	87.45	**96.14**
3	67.16	84.57	66.12	**88.13**	64.58	76.72	88.42	**90.17**
4	97.77	96.07	**98.56**	94.95	58.57	91.47	81.04	**95.68**
5	99.46	99.60	**99.91**	99.70	87.07	95.99	94.42	**98.52**
6	92.83	92.98	97.35	**99.81**	92.72	96.93	**99.11**	98.39
7	90.42	94.32	96.23	**99.66**	29.17	77.69	00.00	**100.00**
8	92.78	84.93	**97.92**	95.53	96.37	99.34	98.41	**100.00**
9	98.11	99.59	96.98	**99.68**	22.22	**78.00**	00.00	72.80
10	–	–	–	–	69.76	83.15	83.26	**95.23**
11	–	–	–	–	79.21	64.47	**98.47**	87.14
12	–	–	–	–	75.41	90.99	94.39	**96.64**
13	–	–	–	–	90.58	99.17	99.48	**99.33**
14	–	–	–	–	91.07	90.56	**97.68**	91.05
15	–	–	–	–	65.50	89.00	84.21	**98.96**
16	–	–	–	–	84.88	69.38	84.88	**93.48**
OA	81.01	86.75	85.42	**96.65**	78.76	80.72	92.48	**92.67**
AA	88.25	89.55	91.31	**95.52**	69.66	85.48	77.26	**94.29**
k	75.86	82.61	81.30	**94.18**	75.73	77.70	91.39	**91.43**

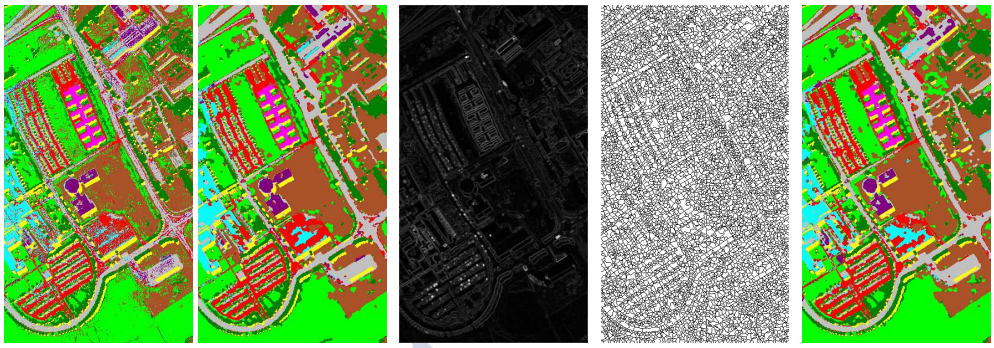
Table 3.5: Per-class classification accuracy as percentages (. 'reg' indicates that the pixel-wise classifier was spatially regularized. 'Watershed' indicates spatial processing by watershed with a connectivity of 8 neighbors.

mentation improves the CPU implementation, achieving speedups up to 5.9×. It can also be noticed that the larger the number of pixel vectors in the dataset, the better the speedup. This is due, particularly, to the ELM and watershed phases, which achieve higher speedups when the dataset is larger in the spatial domain, as can be observed in Table 3.6.

3.4 Discussion

The classification stage of the proposed CD techniques has been studied in this chapter. Most of the proposals are based on supervised classification by ELM. In particular, the following items have been introduced:

An ELM-based GPU implementation to efficiently classify hyperspectral datasets have been presented in this chapter. The GPU implementation takes advantage of the thousands



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

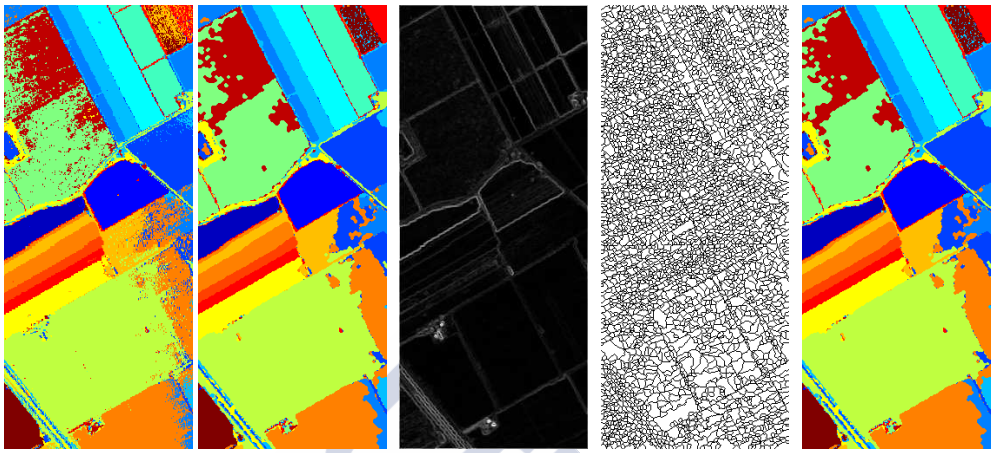
Figure 3.12: Spectral-spatial phases for the Pavia University image. ELM (a), spatial regularization (b), RCMG (c), watershed (d), and MV (e).

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

Figure 3.13: Spectral-spatial phases for the Indian Pines image. ELM (a), spatial regularization (b), RCMG (c), watershed (d), and MV (e).

of threads available, using shared memory to make an effective use of the memory hierarchy, and using a linear algebra library in order to fully exploit the GPU architecture.

Results have shown that commodity GPUs, such as the GTX TITAN used in the experiments, are good candidates for reducing computation times in order to achieve real-time hyperspectral processing. For the raw ELM, speedups of 7.6×, 3.6×, and 5.2 × are achieved, respectively, for the Pavia University, Indian Pines and Salinas datasets, as compared to the multicore CPU version of the same classifier. The results also show that the hyperspectral dataset classification using ELM is faster than the correspondent classification using the traditional SVM. In the experiments, the ELM is up to 8.8× faster in CPU and 8.4× in GPU than the SVM.



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

Figure 3.14: Spectral-spatial phases for the Salinas image. ELM (a), spatial regularization (b), RCMG (c), watershed (d), and MV (e).

	ELM	Reg	RCMG	Watershed	MV	Total
Pavia University						
OpenMP CPU	2.3164s	0.1674s	0.3740s	0.0232s	0.0010s	2.8820s
CUDA GPU	0.3066s	0.0070s	0.1710s	0.0020s	0.0004s	0.4870s
Speedup	7.6×	23.9×	2.2×	11.6×	2.5×	5.9×
Indian Pines						
OpenMP CPU	1.1701s	0.0194s	0.0871s	0.0016s	0.0004s	1.2786s
CUDA GPU	0.3293s	0.0011s	0.0402s	0.0007s	0.0001s	0.3714s
Speedup	3.6×	17.6×	2.2×	2.3×	4.0×	3.4×
Salinas						
OpenMP CPU	1.1017s	0.0984s	0.3957s	0.0159s	0.0006s	1.6123s
CUDA GPU	0.2110s	0.0039s	0.1829s	0.0017s	0.0001s	0.3996s
Speedup	5.2×	25.2×	2.2×	9.4×	6.0×	4.0×

Table 3.6: Performance results of the spectral-spatial scheme based on ELM (ELM+reg+wat con(8)).

Different ELM-based schemes designed to improve the accuracy results have also been explored. An ensemble configuration is considered to achieve better and more robust classification accuracies as well as a spatially regularized version of the algorithm. Finally, the results of incorporating the ELM to spectral-spatial classification schemes based on segmentation

with watershed and RQS have also been analyzed. A comparison to similar spectral-spatial schemes based on SVM was also performed.

In accuracy terms, the spectral-spatial scheme using a spatially regularized ELM and segmentation (ELM+reg+wat con(8)) has been confirmed as a good candidate for the classification of hyperspectral datasets applied to remote sensing. In our experiments, accuracy results up to 10 percentage points better than those obtained by a raw ELM have been achieved. The best accuracy result was 96.66% for the Pavia University image.

Results have also shown that commodity GPUs are good candidates for reducing the computation times of these spectral-spatial schemes aiming to achieve real-time hyperspectral processing. For the ELM+reg+wat con(8) configuration, an execution time of 0.487 seconds in GPU and a speedup of 5.9$\times$ as compared to an OpenMP multicore CPU spectral-spatial scheme were achieved for the Pavia University dataset.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 4

ECAS-II SEGMENTATION ON GPU

Segmentation is a key task associated to the hyperspectral image processing in the field of remote sensing. Most of the classification schemes applied to hyperspectral images can be improved by incorporating spatial information by means of, for instance, a segmentation algorithm. Some of the possible segmentation algorithms to be applied, as the case of the watershed transform, require an initial dimensionality reduction, thus loosing information that could be relevant from the original hyperspectral images. Nevertheless, the algorithm based on a CA called ECAS-II provides segmenters considering all the spectral information contained in a hyperspectral image, without applying any technique for dimensionality reduction.

This chapter presents an efficient GPU implementation of the type of segmenters produced by ECAS-II for land cover hyperspectral images. Results based on applying ECAS-II as a spatial pre-processing for a stage spectral-spatial classification scheme of remote sensing hyperspectral images are also presented. Finally, the main topics of this chapter are discussed in the last section.

4.1 ECAS-II segmenter GPU projection

This section is devoted to explaining the details of the GPU implementation of the general ECAS-II segmentation algorithm. The algorithm fundamentals were introduced in detail in Section 1.4.3. Figure 4.1 shows the detailed pseudocode that has been implemented. Each process executed in GPU is placed between <> symbols and may involve one or more kernels. The pseudocode also includes the GM and SM acronyms to indicate kernels executed in global memory and shared memory, respectively. The process is computed entirely on GPU and a

Require: Hyperspectral Image, Ruleset.
1: ▷ Pre-processing
2: <Create window filters> ▷ GM
3: **while** iteration < maximum iterations **do**
4: <Add apron to hyperspectral image> ▷ GM
5: <Calculate distance to neighborhood pixels> ▷ SM
6: <Calculate and normalize gradients for all windows> ▷ SM
7: ▷ Processing
8: <Calculate gradient distances for all windows> ▷ GM
9: <Calculate gradient angles> ▷ GM
10: <Select rule to apply> ▷ GM
11: ▷ Post-processing
12: <Calculate and weight output gradient> ▷ GM
13: <Average spectrum (filter and update hyperspectral image)> ▷ SM
14: **end while**
▷ GM: Global Memory, SM: Shared Memory

Figure 4.1: Pseudocode for the ECAS-II segmentation algorithm.

graphic representation of its computation scheme for a block of data at a given instant of time can be seen in Figure 4.2.

First, a pre-processing step is computed once the inputs are stored in the GPU global memory. The first thing to do in order to execute the algorithm is to create the different window filters considered (line 2 on the pseudocode of Figure 4.1). In this case, the window sizes are 3×3, 5×5, and 7×7, respectively. This means that each pixel will need access to the spectra of 49 neighboring pixels. It is worth noting than each filter contains vertical and horizontal separable components.

After this, an iterative process starts (lines 3 to 14) adding an apron to the hyperspectral image (line 4) in order to avoid memory access errors when reading the neighbors of the border pixels. For the window sizes considered, the apron must be a border of 3 pixels as displayed in Figure 4.2.

Then, the processing stage begins. A kernel is launched in shared memory that calculates the spectral angle distances between each pixel and all of its neighbors, as indicated in (1.20), and stores them in a matrix in global memory (line 5 in the pseudocode). In this kernel, as represented in Figure 4.2, each thread processes the values for all the neighbors of a particular pixel.

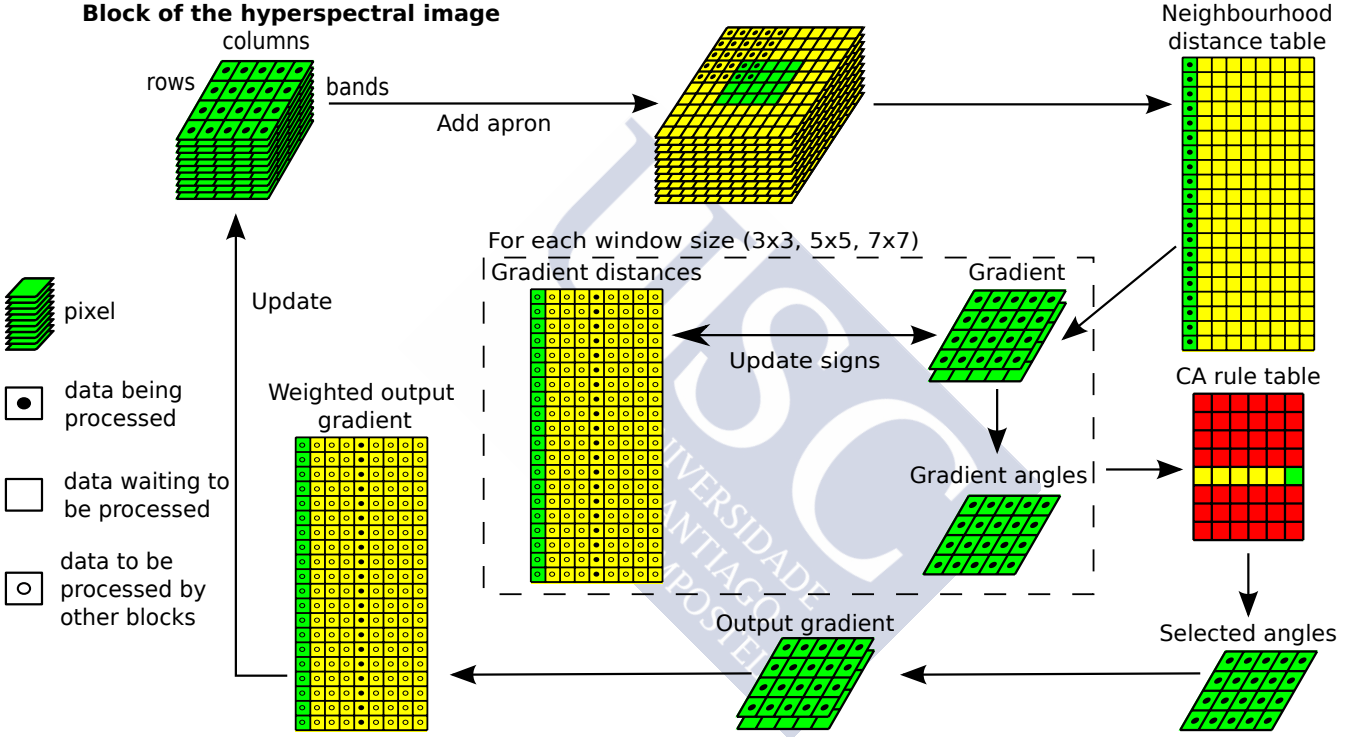


Figure 4.2: GPU computation scheme of a block of threads of size 5×4, showing the data structures and the activity of each thread at a given instant of time.

The next steps (line 6) calculate the gradients for the three windows multiplying the matrix of the previously calculated distances by the corresponding filter in shared memory using the Magma library as indicated in (1.21) , calculate the norms of these gradients, and normalize them (dividing by their norms).

Then, a process starts for each window size, as indicated in Figure 4.2, where three new distance matrices are computed (one for each window size) to indicate the distance of the neighborhood pixels to a line perpendicular to the gradient vectors that divides the neighborhood into two areas (line 8 in the pseudocode). After this, the angles of the gradients for each window are directly updated with respect to the angle where the average distance of the pixels within the neighborhood is smallest. The angles of the gradients are calculated (line 9) as the *arctangent* of the vertical and horizontal components of the gradients (1.22) and stored in a global memory matrix.

Once the gradient modules and angles have been calculated, the process for selecting the rule to apply for each pixel can begin (line 10). In this process the values of the pixel gradients (3 modules and 3 angles, one for each window size) are compared to the parameters of each rule in the ruleset (1.23). For better performance, translations and rotations are applied in the comparison in order to make the rules invariant to these effects. Once the rule is selected, a post-processing stage takes place in which the pixel will be averaged with those within a radius of size 1 moving in the direction indicated by the angle θ of the selected rule.

The output gradient to be applied is calculated and stored in global memory (line 12 on pseudocode) as the cosine and sine of the angle indicated by the selected rule (for the vertical and horizontal components), thereby producing a data structure with two bands, as seen in Figure 4.2. The contribution of each neighboring pixel is determined through a filter that assigns a weight inversely proportional to the distance to the central pixel. Then, the hyperspectral image is updated (line 13) as the weighted average of the spectrum of the selected neighbors for each band and for each pixel, so that no hyperspectral information is lost.

Finally, this updated image is the input to the next iteration of the algorithm. The process lasts until the selected number of iterations is reached. At the end, the last updated image is returned as the output of the algorithm.

4.2 Results

The accuracy results achieved for the proposed classification scheme are addressed in this section. The scheme is evaluated over the Pavia University, Indian Pines, and Salinas dataset introduced in Section 2.3.1. Some performance results are also provided. The GPU performance of this algorithm is evaluated on the NVIDIA GTX TITAN architecture introduced in Chapter 2. Results are shown for a spectral-spatial classification scheme such as the one described in Chapter 3 but replacing the watershed based segmentation with the ECAS-II algorithm. This scheme is called ECAS-II+ELM+reg.

Table 4.1 shows accuracy results for the three test images in terms of OA, AA, and k. The best results are highlighted in bold in the table. When a result is not displayed it is because it is not available in the literature. The results are compared to a similar approach that considers SVM instead of ELM (ECAS-II+SVM). The SVM needs to optimize two parameters (C,γ) corresponding to a penalty term and the radius of the Gaussian function, respectively. In this case, the SVMs have been trained using C=256 and γ=16 for Pavia University, C=512 and γ=0.5 for Indian pines, C=512 and γ=8 for Salinas and the same sampling configurations used for the ELM-based one. These values have been obtained by five-fold cross validation.

The results are also compared to the pixel-wise classifier based on ELM of Chapter 3 trained under the same configurations as the ECAS-II-based schemes, to the ELM+reg+wat con(8) introducing an approach based on watershed segmentation combined with a spatially regularized ELM through a majority vote and the V-ELM-1+reg+wat con(8) configuration adding the use of ensembles to the previous approach. Other spectral-spatial techniques available in the literature are included for comparison purposes: SVM+reg [221] and SVM+wat con(8) [63] introduce similar schemes relying on SVM instead of ELM. SVM+EM [221] and SVM+EM+reg [221] use a segmentation based on partitional clustering instead of the watershed based one. Some approaches using the SAM distance to preserve all the spectral information in the segmentation process are also included: a configuration that combines Hierarchical Segmentation (HSeg) and SVM through a Plurality Vote (PV) (HSeg+PV) is introduced in [146]. Finally, SAM+MV [170] and Stochastic Minimum Spanning Forest (RD-MSF) [147] use Minimum Spanning Forest (MSF) to add spatial information to an SVM through an MV process.

It can be seen that the implementations based on ECAS-II outperform all the different alternatives included from the literature. The configuration based on a spatially regularized ELM presented in this thesis achieves the better results for all the datasets. It achieves 98%

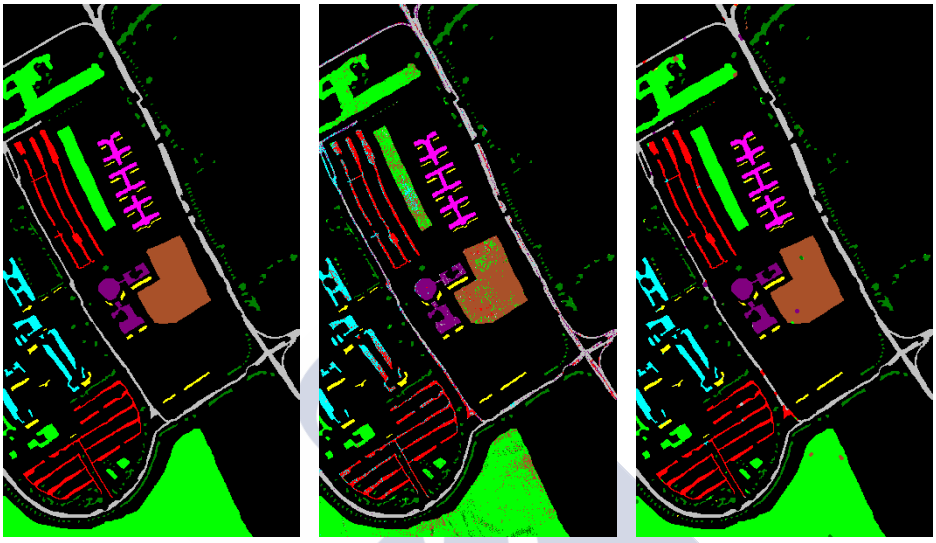
	Pavia University			Indian Pines			Salinas		
	OA	AA	k	OA	AA	k	OA	AA	k
ECAS-II+ELM+reg	**98.43**	98.05	**97.90**	**98.67**	**98.53**	**98.42**	**97.22**	96.92	**96.28**
ECAS-II+SVM	97.51	96.12	96.71	95.77	95.47	95.18	95.85	**98.06**	95.39
ELM [122]	86.75	89.55	82.61	80.72	85.48	77.70	91.55	95.97	90.55
ELM+reg+wat con(8) [122]	95.65	95.52	94.18	92.67	94.29	91.43	93.70	96.78	92.95
V-ELM-1+reg+wat con(8) [122]	96.66	95.92	95.00	90.41	95.35	86.21	92.43	96.75	91.15
SVM+reg [221]	84.27	90.89	79.90	88.58	77.27	86.93	–	–	–
SVM+wat con(8) [63]	85.42	91.31	81.30	92.48	77.26	91.39	–	–	–
SVM+EM [221]	93.59	94.39	91.48	87.25	70.34	85.43	–	–	–
SVM+EM+reg [221]	94.68	95.21	92.02	88.83	71.90	87.24	–	–	–
HSeg+PV [146]	98.35	**98.15**	97.79	86.89	89.83	84.84	–	–	–
SAM+MV [170]	–	–	–	91.80	94.28	90.64	–	–	–
RD-MSF [147]	–	–	–	91.33	93.73	–	–	–	–

Table 4.1: Classification accuracy as percentages. 'reg' indicates that the pixel-wise classifier was spatially regularized. 'wat' indicates spatial processing by watershed. 'con(x)' indicates connectivity of 'x' neighbors.

of overall accuracy for the Pavia University and Indian Pines datasets and 97% for the Salinas dataset and an average accuracy also up to 98%. It can also be noticed that the approaches that preserve all the spectral information thanks to the use of the SAM distance achieve better accuracy results in general, especially in the case of the Pavia University dataset for the HSeg+PV scheme. Figures 4.3, 4.4, and 4.5 show the reference data for classifications and false color classification maps obtained by the ECAS-II+ELM+reg algorithm and also for the raw ELM for comparison purposes. It can be noted that the spatial segmentation through the ECAS-II algorithm removes most of the isolated misclassified pixels.

Table 4.2 shows the sizes of the data structures in Figure 4.2 for the case of the Pavia University dataset. The maximum memory required in a given instant of time is 1.5 times the size of the dataset, corresponding to the step for calculating the gradient distances.

Regarding the performance in terms of execution times, Table 4.3 compares the execution times (in seconds) of the ECAS-II segmentation algorithm for the different images depending on the configuration chosen for the configurable shared/cache memory on the GPU as explained in Section 2.2, optimization technique 2. "Prefer SM" indicates the case when 48 KB are dedicated to shared memory and 16 KB for L1 cache and "Prefer L1" indicates the opposite. "Prefer none" is for the case when 32 KB are devoted to each one. As stated above, the application needs a large amount of shared memory, which explains why the execution times

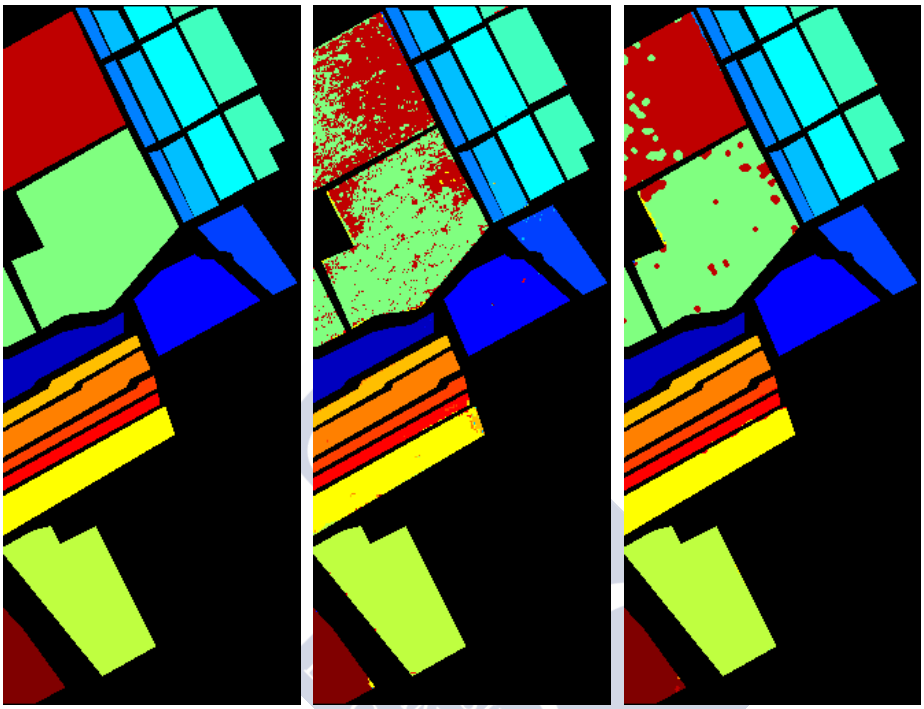


(a) (b) (c)

Figure 4.3: Reference data for classification (a), ELM classification map (b), and ECAS-II+ELM+reg classification map (c) for the Pavia University dataset.

(a) (b) (c)

Figure 4.4: Reference data for classification (a), ELM classification map (b), and ECAS-II+ELM+reg classification map (c) for the Indian Pines dataset.



(a) (b) (c)

Figure 4.5: Reference data for classification (a), ELM classification map (b), and ECAS-II+ELM+reg classification map (c) for the Salinas dataset.

Data structure	**Dimensions**	**Size**
Image + apron (6 pixels)	616×346×103	175.624
Neighborhood distances	(610×340)×49	81.301
Gradients	3×(610×340×2)	9.955
Gradient distances	3×((610×340)×49)	243.902
Gradient angles	3×(610×340)	4.978
CA rule table	20×6	0.001
Selected angles	610×340	1.659
Output gradient	610×340×2	3.318
Weighted output gradient	(610×340)×49	81.301

Table 4.2: Size (in megabytes) of the data structures in Figure 4.2 for the Pavia University dataset (Double data type).

	Pavia University	**Indian Pines**	**Salinas**
Prefer SM	**13.31**	**3.21**	**12.89**
Prefer L1	30.05	7.27	30.62
Prefer none	13.32	3.21	12.89

Table 4.3: ECAS-II segmenter execution times (in seconds) for Pavia University, Indian Pines, and Salinas images for the different L1-SM configurations.

Line	% of CPU time	**Pavia University**			**Indian Pines**			**Salinas**		
		CPU	GPU	Speedup	CPU	GPU	Speedup	CPU	GPU	Speedup
4	0.67%	0.0965	0.0026	37.8×	0.0158	0.0009	17.2×	0.0931	0.0029	31.8×
5	28.48%	3.5691	0.4978	7.2×	0.7724	0.1488	5.2×	3.7583	0.5490	6.8×
6	2.55%	0.5053	0.0270	18.7×	0.0502	0.0030	16.5×	0.2643	0.0150	17.6×
8	2.34%	0.4288	0.0786	5.5×	0.0513	0.0122	4.2×	0.2403	0.0440	5.5×
9	0.03%	0.0068	0.0008	8.5×	0.0007	0.0001	6.3×	0.0035	0.0007	5.3×
10	12.09%	2.3965	0.1196	20.0×	0.2380	0.0124	19.2×	1.2578	0.0658	19.1×
12	0.35%	0.0720	0.0112	6.4×	0.0065	0.0019	3.5×	0.0376	0.0063	5.9×
13	52.70%	6.5618	0.1273	51.5×	0.8227	0.0303	27.1×	12.6365	0.1395	90.6×
TI		13.6368	0.8650	15.8×	1.9576	0.2096	9.3×	18.2913	0.8233	22.2×

Table 4.4: ECAS-II segmenter detailed execution times (in seconds) and speedups OpenMP CPU – CUDA GPU by step (for one iteration of the algorithm) for Pavia University, Indian Pines, and Salinas images. 'line' refers to the corresponding line in Figure 4.1. 'TI' indicates the total time for one iteration. '% of CPU time' represents the percentage of 'TI' in CPU for each step of the algorithm.

are notably lower for every image when more shared memory than L1 cache is assigned. Regarding this, the remaining results will follow the "Prefer SM" configuration.

Table 4.4 details the execution times and speedups of each step of the algorithm for the three datasets both in CPU (OpenMP) and GPU (CUDA) where shared memory is used. It also shows the percentage of the total CPU time (on average for the three datasets) for each step. 'TI' indicates total time per iteration; i.e., the sum of the time for the different steps included in Table 4.4. The steps for calculating the neighborhood distances, select rule, and average spectrum (lines 5, 10, and 13 in Table 4.4 and in the pseudocode of Figure 4.1, respectively) cover 93% of the execution time of each iteration. These three steps are therefore the most relevant ones to be optimized.

In particular, the kernel that calculates the distance to the neighborhood pixels (line 5) obtains a high speedup because of the data load pattern, the kernel configuration and the efficient use of shared memory (Section 2.2, optimization techniques 2, 4, and 6). As its

operation involves data reuse when accessing the neighborhood of each pixel, the shared memory of the GPU is fully exploited to achieve better speedups. The kernel works with 2D blocks of data in the spatial dimension. For this reason, the speedup is better the bigger the spatial dimension of the image, achieving the best result of 7.2× for the Pavia University dataset. The theoretical occupancy is 43%, limited by the amount of shared memory needed, and the achieved occupancy is between 41% and 42% for the three datasets.

For the average spectrum kernel (line 13 in Table 4.4 and in pseudocode of Figure 4.1) GPU optimization techniques 2, 4, and 6 (Section 2.2) are applied. The kernel involves the reuse of neighborhood pixels and, therefore, the use of shared memory allows kernel efficiency to be improved. It works with 3D blocks of data, which allows a better speedup the bigger the dimensions of the image, both spectral and spatial. As shown in Table 4.4, the Salinas dataset achieves the best speedup for this kernel (90.6×). The occupancy achieved in this kernel is between 72% and 73%, with the maximum theoretical occupancy being 75%, limited by the amount of shared memory available.

Finally, the kernel for selecting the optimum rule for each pixel (line 10) exploits the faster accesses to the registers and its speedup is based on the optimization techniques 1 and 6 described in Section 2.2. This kernel computes a fixed-size set of rules. It works in 1D blocks of threads simultaneously comparing each pixel gradient against the ruleset, which is stored in local memory and registers to minimize access times. It achieves a speedup between 19× and 20× and the execution time is higher the bigger the spatial dimensions of the image. In this case, the theoretical occupancy is limited by the number of registers needed to 25%, achieving in practice a occupancy of 24.9%.

The global performance results in terms of execution times (in seconds) and speedups calculated over the OpenMP multicore implementations for 4 threads are shown in Table 4.5. Two GPU implementations are presented in the table, including the time needed to transfer data to the global memory of the GPU. The first one in the table, CUDA GPU GM stands for an implementation fully executed in global memory. On the other hand, CUDA GPU SM stands for the implementations indicated in Figure 4.2, where the faster shared memory is exploited as far as possible. The GPU implementation achieves large speedups for the three datasets. Execution times show that the GPU versions achieve better speedup the larger the dimensions of the image. They also show that the shared memory version (CUDA GPU SM in Table 4.5), as expected, achieves better speedup than the global memory version (CUDA GPU GM in the table) the larger the spatial dimensions of the image, achieving a maximum

	Pavia University	Indian Pines	Salinas
OpenMP CPU	206.57	29.69	275.16
CUDA GPU GM	16.45	3.34	15.72
CUDA GPU SM	13.31	3.21	12.89
Speedup CPU-GPU GM	12.6×	8.9×	17.5×
Speedup CPU-GPU SM	**15.5**×	**9.2**×	**21.4**×

Table 4.5: ECAS-II segmenter execution times (in seconds) for Pavia University, Indian Pines, and Salinas images.

speedup of 21.4× for the Salinas image. The reason is a better exploitation of the high number of computing threads available that simultaneously process each spatial pixel of the image. If the entire spectral-spatial classification scheme is considered, the ELM algorithm is much faster than the SVM, with speedups of up to 8.8× with respect to the SVM as presented in [122].

4.3 Discussion

The high relevance of the spatial processing in order to increase the accuracy of the classification methods when they are applied to hyperspectal images gives rise to the need for the proposal of efficient spatial processing techniques, as is the case for the segmentation algorithms. In particular, a GPU segmentation algorithm that exploits all the information available in a hyperspectral images is proposed in this chapter.

ECAS-II is based on CA that consider the full spectra of the hyperspectral image at all time in order to compute gradients related with different neighborhood window sizes for the updating the spectrum of each pixel. After the application of the algorithm, the modified hyperspectral dataset is implicitly segmented through regions with equal or similar spectral signatures.

Furthermore, in this chapter the ECAS-II segmenter is also combined with the ELM classifier introduced in Chapter 3 in order to obtain a final spectral-spatial classification map. The ECAS-II based spectral-spatial classification scheme was evaluated in three hyperspectral datasets. The accuracies achieved in the evaluation of this scheme show better results than other state-of-the-art techniques, reaching 98% of overall accuracy for the Pavia University and Indian Pines datasets and 97% for the Salinas dataset. Regarding the efficiency of the

algorithm, the GPU projection developed achieves speedups of up to 21.4× compared to an efficient OpenMP CPU version of the same algorithm.

A detailed analysis of the speedups obtained in each step of the algorithm and the exploitation of the GPU in terms of occupancy has been carried out. An evaluation of the benefits of taking advantage of the available shared memory and the different cache configurations has also been performed. An appropriate use of the shared memory returns better speedups for the three considered datasets with respect to a global memory implementation of the ECAS-II. For the case of the segmentation algorithm proposed in this chapter, where the neighborhood information in the spatial domain is widely used, the configuration that maximizes the shared memory available provides the best execution times of 13.32, 3.21, and 12.89 seconds for the Pavia University, Indian Pines, and salinas datasets, respectively.

CHAPTER 5

BINARY CHANGE DETECTION FOR MULTITEMPORAL DATASETS ON GPU

The CD process aims to detect and/or classify the significant changes existent between a set of multitemporal images covering the same space at different time-frames. In this context, CD techniques are aimed at the binary or multiclass detection of changes. The former approach is devoted to label the pixels in the image as change or no change. The latter one deals with the classification of the change pixels regarding the different types of transitions detected.

The CD scheme for multitemporal hyperspectral datasets proposed in this thesis includes a processing at feature level in order to discriminate between change and no change pixels (binary branch). Then, a multiclass processing allows us to classify the types of change. Regarding the performance in terms of execution time, a time-efficient binary CD branch will allow a faster computation time, as the multiclass branch of the CD is computationally more costly and its cost can be reduced if it is only applied to the pixels previously selected as change. This chapter is devoted to the introduction of the binary branch of the CD scheme. The proposed method considers both the spectral and the spatial information from the dataset to efficiently achieve an accurate object level binary CD map. The multiclass CD scheme will be studied in Chapter 6.

The first section of the chapter is devoted to the introduction of the proposed scheme for binary CD. The following section details the efficient GPU implementation of the scheme, along with the main differences with sequential and OpenMP based implementations of the same scheme. After this, the results achieved for two multitemporal hyperspectral datasets

(I) **Segmentation.**

Input: two co-registered hyperspectral images.

Output: two segmented hyperspectral images.

(1) Generate segmented images.

(II) **Fusion.**

Input: two segmented hyperspectral images.

Output: difference image.

(1) Apply CVA with SAM distance.

(III) **Thresholding.**

Input: difference image.

Output: raw change map.

(1) Obtain the range of the data in the image.

(2) Calculate the histogram of the data.

(3) Calculate threshold through Otsu's method.

(4) Generate binary image regarding threshold.

(IV) **Spatial regularization.**

Input: raw change map.

Output: spatially regularized change map.

(1) Apply neighbor-based spatial regularization to the raw change map.

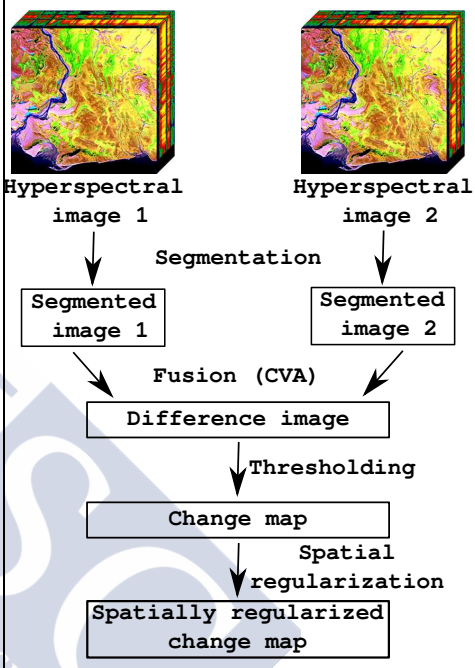


Figure 5.1: Binary CD flowchart in the spectral-spatial scheme.

through the application of the proposed scheme are provided and compared with alternative techniques for each stage. Finally, the last section studies the projection of the proposed scheme to multi-GPU systems.

5.1 Spectral-spatial binary CD scheme

The proposed spectral-spatial scheme for binary CD is introduced in this section. The flowchart of the scheme and its stages are detailed in Figure 5.1. As can be seen in the right half of the figure, the inputs of the scheme are two co-registered hyperspectral images acquired in two different time-frames over the same area.

The scheme focuses on the detection of changes at the object level. Therefore, it includes a first stage aimed at finding objects in the input images. This stage is tackled through a segmentation technique (Stage I in Figure 5.1). In this stage, both input images are independently segmented through a watershed transform. The watershed segmentation fits this approach particularly well as it produces over-segmented region maps, minimizing the probability of including more than one semantic object in the same region. The resulting segmentation map is then used to perform a region averaging of the spectra of all the pixel-vectors in the same watershed region, thus obtaining the segmented images.

The second stage consists of the fusion of the multitemporal information (Stage II in Figure 5.1) to obtain a difference image. In this thesis, the SAM distance is used to perform this fusion as it provides better results when working with hyperspectal images than the ED, as explained in Section 1.2.

The next stage (Stage III in Figure 5.1) performs a thresholding of the difference image in order to obtain a binary CD map. Otsu's method is used to this end in this thesis. The application of Otsu's method requires the prior computation of the histogram of the difference image, which makes it necessary to know the range of the data in the image. Finally, the obtained threshold value is used to generate the binary map. In this map, only the labels 1 and 0 are present, indicating change and no change pixels, respectively.

The last stage (Stage IV in Figure 5.1) applies an iterative neighborhood-based spatial regularization so that a pixel labeled as change is updated to the no change label if more than half of its neighbors share that label. The same is applied under the opposite conditions. This technique allows us to remove misclassified disconnected pixels from the final CD map.

5.2 GPU projection of the binary CD scheme

This section explains in detail the GPU implementation of the binary CD scheme for hyperspectral datasets proposed in the previous section. Figure 5.2 shows the data structures employed along with the thread activity on GPU. Some pseudocodes are introduced, where each process executed in GPU is placed between <> symbols and may involve one or more kernels. The pseudocodes also include the GM and SM acronyms to indicate kernels executed in global memory or shared memory, respectively. Different optimization techniques have been applied in order to improve the GPU efficiency of the codes. References to the optimization techniques explained in Section 2.2 are mentioned throughout this section.

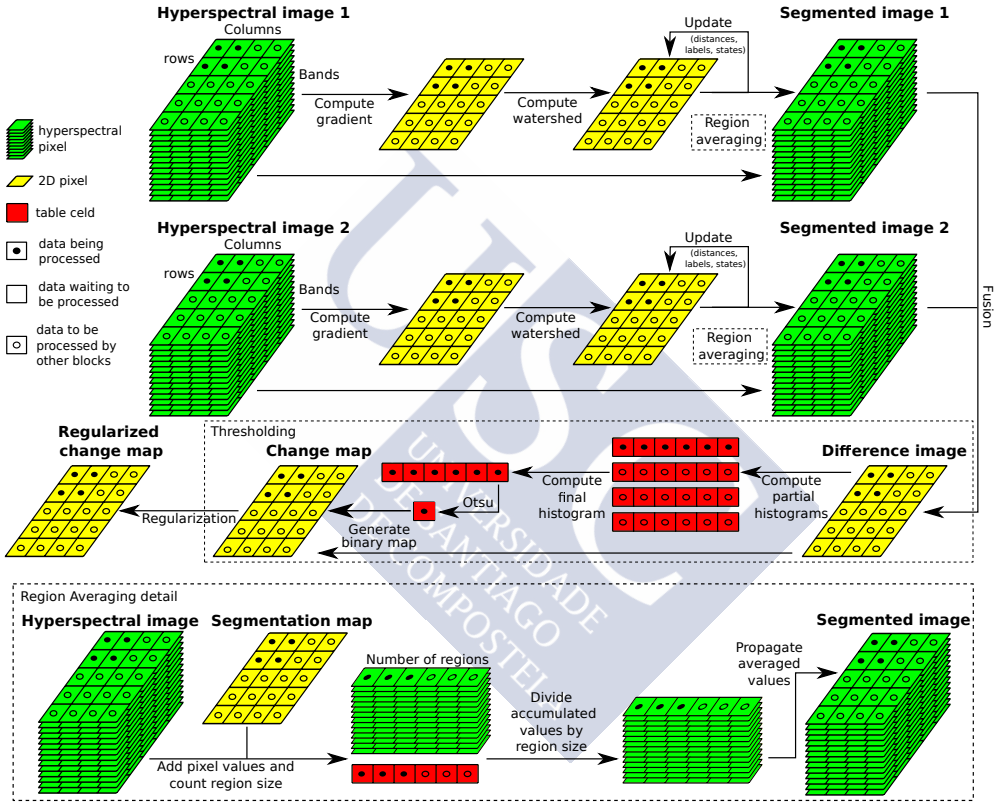


Figure 5.2: GPU computation diagram for the CD scheme, showing the data structures and the activity of each thread at a given instant. The data structures stored in global or shared memory are classified as hyperspectral data, 2D data and table (all the structures that do not correspond to the first two categories). In a given instant, data may be computed by a thread of the current block, by a thread of another block or it may be waiting to be processed.

Gradient on GPU.
Input: hyperspectral image $\boldsymbol{X}$.
Output: single band image. ▷ Distances step (1)

1: **for each** band k of $\mathbf{X}$ **do**
2: <Load band k in shared memory> ▷ SM-GM
3: **for each** pixel i, j in band k **do**
4: <Compute the x term of the gradient as $gradx(i,j) = I(i+1,j) - I(i-1,j)$ > ▷ SM
5: <Compute the y term of the gradient as $grady(i,j) = I(i,j+1) - I(i,j-1)$ > ▷ SM
6: <Accumulate the gradient module as $grad(i,j) += sqrt(gradx(i,j)^2 + grady(i,j)^2)$ > ▷ SM
7: **end for**
8: < Synchronize threads within the block> ▷ SM
9: **end for**
10: <Write $grad(i,j)$ to global memory> ▷ SM-GM

▷ GM: Global Memory, SM: Shared Memory

Figure 5.3: Pseudocode for the gradient kernel required by the watershed-based segmentation corresponding to step (I) in Figure 5.1.

5.2.1 Segmentation stage on GPU

As shown in Figure 5.1, and 5.2, each image is independently segmented using the watershed transform [163]. The segmentation stage involves a three-step processing: calculation of a gradient, creation of a watershed-based segmentation map, and averaging of the pixels belonging to the same regions, as explained in Section 5.1.

The first step consists in calculating the magnitude of a gradient. This gradient reduces the hyperspectral image to one band as shown in Figure 5.3. For each pixel, and for each band of the pixel, the horizontal and vertical terms of the gradient are calculated separately. Then, the norm of these terms is computed and the value of the gradient for each band is accumulated to produce the final gradient value for each pixel. This process is computed in shared memory (optimization technique 1 in Section 2.2) and requires a load operation in shared memory for each band of the image.

A segmentation map is then calculated through the watershed technique based on a cellular automaton [163] introduced in Section 3.2.1, using the one-band image obtained by the previous gradient as input.

Region averaging on GPU.
Input: hyperspectral image, watershed segmentation map.
Output: region-averaged hyperspectral image.
1: <Add pixel values in each watershed region> ▷ GM
−Check the region of the pixel in the segmentation map.
−Atomically add 1 to the corresponding region counter.
−Atomically add each band of the pixel to the corresponding accumulator.
2: <Divide accumulated values by number of pixels in each region> ▷ GM
3: <Propagate the averaged value to each pixel of each region> ▷ GM
▷ GM: Global Memory

Figure 5.4: Pseudocode for the region averaging process in the segmentation of step (I) in Figure 5.1.

Finally, in the third step of the segmentation, the process to average the spectra of the pixels belonging to the same segmentation region starts (see bottom of Figure 5.2 and Figure 5.4). It involves three consecutive kernels (optimization 8 introduced in Section 2.2) that are executed in global memory as the size of the regions is unknown and there is no data reuse. The first kernel accumulates the values of each pixel belonging to a region. Each thread atomically adds the spectral values of a pixel to the corresponding region values and also increments a counter for the number of pixels in the region. The second kernel divides the accumulated pixel values for each region by the number of pixels of the region, thus using as many threads as there are regions in the image. The final kernel propagates the calculated average values to every pixel belonging to each region, needing again as many threads as there are pixels in the image. In the three kernels, each thread computes all the bands of the corresponding pixel. The computation is carried out in-place (optimization technique 1 in Section 2.2).

After the segmentation stage, the images will have no intra-region variability and a increased inter-region variability. This allows us to detect the changes more easily and to do so at object level instead of at pixel level. By assigning the same value to all the pixels in the region, that region (i.e., corresponding to an object or a part of it) will be considered as a whole in the thresholding step.

Alternatively, the segmentation stage can be carried out through an ECAS-II segmenter [65] following the pseudocode in Figure 4.1 and whose GPU implementation details can be read in Section 4.1. The segmentation is based, as shown in Figure 4.1, on the iterative use of gradients with respect to neighboring pixels and an automatically generated ruleset to progressively modify the spectra of the pixels of the image until the spectra of those pixels belonging

Image fusion on GPU.
Input: two segmented images.
Output: difference image.
1: <Apply CVA with SAM distance> ▷ GM
−Accumulate the *product* for each band between correspondent pixels.
−Accumulate the square of each pixel for each band
for both images → (*firstSquare*, *secondSquare*).
−Calculate distance as $(2/\pi) * \arccos(product/(\sqrt{firstSquare}) * \sqrt{secondSquare}))$.
−Denormalize the distance by a scale factor.
▷ GM: Global Memory

Figure 5.5: Pseudocode for the fusion stage corresponding to step (II) in Figure 5.1.

to the same region is homogenized. The iterative process implies a higher computational cost for this method than the watershed calculation; nevertheless, the ECAS-II segmentation provides more accurate segmented images as it works with all the spectral information of the image.

5.2.2 Fusion stage on GPU

The following step combines the segmented images into a difference one (See Figure 5.1). This is usually handled by CVA processing [101] using ED. In this thesis, the difference image is created by using the SAM distance,

$$\alpha_{i,j} = \frac{2}{\pi}\cos^{-1}\left(\frac{S_j \cdot S_i}{\|S_j\|\|S_i\|}\right) \in [0,1], \tag{5.1}$$

where $\alpha_{i,j}$ is the spectral angle between the spectrum at pixel i (S_i) and the one at j (S_j), and i, j are the two pixels under consideration.

The fusion of the two segmented hyperspectral images through CVA is computed by a kernel (Figure 5.5) in which each thread computes the SAM distance between the two pixel-vectors corresponding to the same spatial location in the two segmented images. This distance is then denormalized by applying a scale factor that allows the subsequent computation of a histogram of the difference image. The distance computation is carried out in global memory generating the difference image in Figure 5.2.

5.2.3 Thresholding stage on GPU

A thresholding process is needed to obtain the change map of the images (See Figure 5.1). In the approach presented in this thesis, it is performed following Otsu's algorithm [103]. To do so, a histogram of the previously calculated difference image is required. In order to preserve as much detail as possible, this histogram is created with as many bins as gray levels in the difference image.

This process (see pseudocode in Figure 5.6) starts by calculating the range encompassing the values of the difference image. This can be achieved efficiently through the cuBLAS library [201] (optimization strategy 6 in Section 2.2). After this, the histogram of the data is efficiently calculated through two consecutive kernels (optimizations 6 and 8 in Section 2.2). First, n partial histograms are obtained (n being the number of blocks of threads) as shown in Figure 5.2, in a kernel where each thread processes a pixel of the difference image. In the second kernel, the partial histograms are combined in order to obtain the final histogram, using as many threads as there are bins in the histogram. Thus each thread writes in a different position of the array.

The next task is to calculate the global value of the threshold using Otsu's algorithm. It assumes two classes of pixels in the histogram, and calculates, in an iterative process, the optimal threshold as the one that provides the maximum inter-class variance. This process is intrinsically sequential (it iterates through the possible thresholds and selects the one resulting in the largest inter-class variance), thus it is computed by a single thread. This is not a problem as long as it is an extremely quick step.

Finally, the CD map is directly created through a process in which all the pixels in the difference image whose value is greater than the calculated threshold are identified as change pixels. The same number of threads as pixels in the difference image are launched in this kernel. The output of the thresholding stage (the change map) is stored over the same memory that contains the magnitude of change for each pixel (optimization technique 1 in Section 2.2).

5.2.4 Spatial regularization on GPU

The previous processing can give rise to some unconnected pixels or noise in the CD map. This is corrected by applying an iterative spatial regularization technique similar to the one introduced in [122].

Thresholding on GPU.
Input: difference image.
Output: raw change map.
1: <Search data interval in the difference image> ▷ SM-GM
2: <Calculate partial histograms of the data> ▷ GM
3: <Merge partial histograms> ▷ GM
4: <Calculate threshold by Otsu's method> ▷ GM
5: <Generate binary image regarding the calculated threshold > ▷ GM
▷ GM: Global Memory, SM: Shared Memory

Figure 5.6: Pseudocode for the thresholding stage corresponding to step (III) in Figure 5.1.

Spatial regularization on GPU.
Input: raw change map.
Output: spatially regularized change map.
1: **while** Global flag > 0 **do**
2: <Global flag $= 0$ > ▷ GM
3: <Apply neighborhood-based spatial regularization> ▷ GM
4: **if** Any pixel value changes **then**
5: <Global flag $++$ > ▷ GM
6: **end if**
7: **end while**
▷ GM: Global Memory

Figure 5.7: Pseudocode for the spatial regularization stage corresponding to step (IV) in Figure 5.1.

This stage is computed by a single kernel, as shown in Figure 5.7. The information of the closest neighborhood of each pixel is exploited in an iterative process. The value of each pixel in the CD map is computed by a thread. A global flag is activated if any pixel is updated, indicating that the iterative process must last one more iteration. The process is computed in-place (optimization technique 1 in Section 2.2) in global memory in order to guarantee that the values of the neighbors of a pixel are always correctly updated.

5.2.5 Comparison to the sequential and OpenMP CPU implementations

Some stages of the method present different implementations in the CPU and GPU versions in order to fully exploit each architecture. In particular, the region averaging step of the segmentation stage (Figure 5.4) and the fusion stage (Figure 5.5) are performed in pixel-vector

order (i.e. the image is stored in memory so that the components of a pixel are contiguous) in the CPU implementations. The GPU version works in band-vector order. This aims to maximize the exploitation of the available cache memories in the CPU and maximize the number of threads being executed in parallel in the GPU, respectively (optimization techniques 1 and 2 in Section 2.2). Furthermore, the histogram calculation of the thresholding stage (Figure 5.6) follows a two-step processing in GPU, as explained in Section 5.2.3, while this is not necessary in the CPU version of the code, which computes it in only one step.

5.3 Experimental results of the binary CD scheme

This section presents the accuracy and execution performance results obtained over experimental data for the evaluation of the binary CD scheme. The Santa Barbara and Bay Area datasets presented in Section 2.3.3 were selected for this purpose. The NVIDIA GTX TITAN X introduced in Section 2.3.1 is used for the performance results in GPU. As was explained in Section 2.3.2, the accuracy of the binary CD is evaluated in terms of correctly classified pixels, MAs, FAs, and total error. The computational efficiency of the scheme is evaluated in execution time, speedup, and hardware exploitation.

Table 5.1 (for the Santa barbara dataset) and Table 5.2 (for the Bay Area dataset) show the accuracies obtained using the proposed binary CD scheme (SAM+watershed+Otsu for the configuration based on watershed segmentation and SAM+ECAS-II+Otsu for the one based on ECAS-II segmentation) with CVA based on SAM distance and Otsu's thresholding, along with other configurations included for comparison purposes. In particular, ED+EM can be used as a reference for comparison, as it is frequently done in the bibliography [102]. It includes an ED-based CVA and EM thresholding. ED+Otsu includes CVA with ED and Otsu-based thresholding. SAM+Otsu substitutes the ED by SAM distance. ED+watershed+Otsu adds the spatial processing based on watershed segmentation plus region averaging and a final spatial regularization as explained in Section 5.2.4. SAM+watershed+EM and SAM+watershed+Otsu add the same spatial processing to the configuration that uses the SAM distance. SAM+ECAS-II+EM, SAM+ECAS-II+K-means, and SAM+ECAS-II+Otsu, replace the watershed segmentation + region averaging spatial processing by an ECAS-II segmentation comparing three different thresholding techniques (EM, K-means, and the Otsu one).

The proposed scheme provides very good binary CD results, achieving up to a 97.04% and 97.41% of correctly classified pixels for the Santa Barbara and Bay Area datasets, re-

spectively, when the segmentation by ECAS-II is applied (SAM-ECAS-II+Otsu configuration). The watershed-based configuration of the scheme (SAM+watershed+Otsu) achieves up to 96.96% and 96.94% of correctly classified pixels, respectively. It can be seen that, in general, the configurations based on SAM distance (SAM+Otsu, SAM+watershed+Otsu, SAM+watershed+EM, SAM+ECAS-II+EM, SAM+ECAS-II+K-means, and SAM+ECAS-II+Otsu) greatly improve the accuracy obtained by those using ED based CVA (all the remaining configurations), achieving up to 11 more percentage points. It is also observed that the spatial information included improves the results in both the ED and SAM cases. The configurations replacing Otsu's threshold with the EM technique (ED+EM, SAM+watershed+EM, SAM+ECAS-II+EM) or K-means clustering (SAM+ECAS-II+K-means) show that Otsu's approximation provides a better thresholding of the difference data and, therefore, a better change map.

The change maps achieved by the different configurations are shown in Figs. 5.8 (e-h) and 5.9 (e-h). The figures also include a color composition of the input images (a, b), the reference data of the changes (c) and the map of changes obtained by the best configuration in terms of accuracy with respect to the reference data (d). As it can be seen in Figs. 5.8 (c) and 5.9 (c), the reference data cover regions with different sizes and shapes throughout the image. Figs. 5.8 (d) and 5.9 (d) show that the scheme correctly detects the presence or absence of changes in a large percentage of the reference data (97.04% and 97.41%, respectively, for both test images). Figs. 5.8 (g) and (h), for the Santa Barbara dataset, and 5.9 (g) and (h), for the Bay Area dataset, show that the spatial information added by the new proposals SAM+watershed+Otsu and SAM+ECAS-II+Otsu remove most of the isolated pixels present in Figs. 5.8 (e) and (f) and 5.9 (e) and (f), improving the final accuracy. The removal of isolated pixels improves the CD accuracy as the proposals of this thesis target detection at object level instead of the detection of changes in disconnected pixels.

Regarding the computational efficiency, the execution times obtained for the sequential CPU, the OpenMP CPU using 4 threads, and the CUDA GPU versions of the scheme are detailed in Table 5.3. The OpenMP CPU version of the scheme achieves a reasonable speedup when compared to the sequential one. A speedup of up to 46.5$\times$ is achieved with the GPU implementation as compared to the fastest CPU version (the OpenMP one). It can be seen that all the GPU algorithms in the scheme achieve high speedups. The stages involving a higher computational load (i.e., the gradient calculation, the region averaging step, and the spatial regularization stage) achieve large speedups (65.4$\times$, 32.4$\times$, and 65.6$\times$, respectively,

Configuration	Correct		MAs	FAs	Total Errors	
ED+EM	110065	(83.04%)	7990	14497	22487	(16.96%)
ED+Otsu	110348	(83.25%)	9866	12338	22204	(16.75%)
SAM+Otsu	125598	(94.75%)	4147	2807	6954	(5.25%)
ED+watershed+Otsu	115101	(86.83%)	7842	9609	17451	(13.17%)
SAM+watershed+EM	126389	(95.35%)	**1765**	4398	6163	(4.65%)
SAM+watershed+Otsu	128523	(96.96%)	3011	1018	4029	(3.04%)
SAM+ECAS-II+EM	126045	(95.09%)	1856	4651	6507	(4.91%)
SAM+ECAS-II+K-means	128083	(96.63%)	3938	**531**	4469	(3.37%)
SAM+ECAS-II+Otsu	**128626**	**(97.04%)**	3302	624	**3926**	**(2.96%)**

Table 5.1: Accuracy results for binary CD of the Santa Barbara dataset (correctly classified pixels, MAs, FAs, and total errors), in terms of number of pixels and percentages.

Configuration	Correct		MAs	FAs	Total Errors	
ED+EM	64530	(88.84%)	4245	3861	8106	(11.16%)
ED+Otsu	61745	(85.01%)	8940	1951	10891	(14.99%)
SAM+Otsu	68952	(94.93%)	2272	1412	3684	(5.07%)
ED+watershed+Otsu	62944	(86.66%)	8476	1216	9692	(13.34%)
SAM+watershed+EM	68548	(94.37%)	868	3220	4088	(5.63%)
SAM+watershed+Otsu	70411	(96.94%)	2011	214	2225	(3.06%)
SAM+ECAS-II+EM	67640	(93.12%)	**777**	4219	4996	(6.88%)
SAM+ECAS-II+K-means	69868	(96.19%)	2652	**116**	2768	(3.81%)
SAM+ECAS-II+Otsu	**70757**	**(97.41%)**	1421	458	**1879**	**(2.59%)**

Table 5.2: Accuracy results binary for CD of the Bay Area dataset (correctly classified pixels, MAs, FAs, and total errors), in terms of number of pixels and percentages.

for the largest dataset). The smallest speedups correspond to the fusion and thresholding stages (17.3× and 7.0× for the Santa Barbara dataset and 16.7× and 3.0× for the Bay Area dataset) but these are stages which involve low computational load and, therefore, do not significantly affect the total speedup. Finally, it is worth considering that the CPU-GPU transfer time is 0.917s for the Santa Barbara dataset and 0.353s for the Bay Area dataset. This means that even if the images have to be loaded into the GPU memory only for this calculation, which is not usually the case in real applications, the total execution time of the GPU version would be faster than the best CPU version.

In order to outline the computational requirements of the scheme, it is worth noting that the maximum memory required in GPU at a given instant is approximately twice the size of

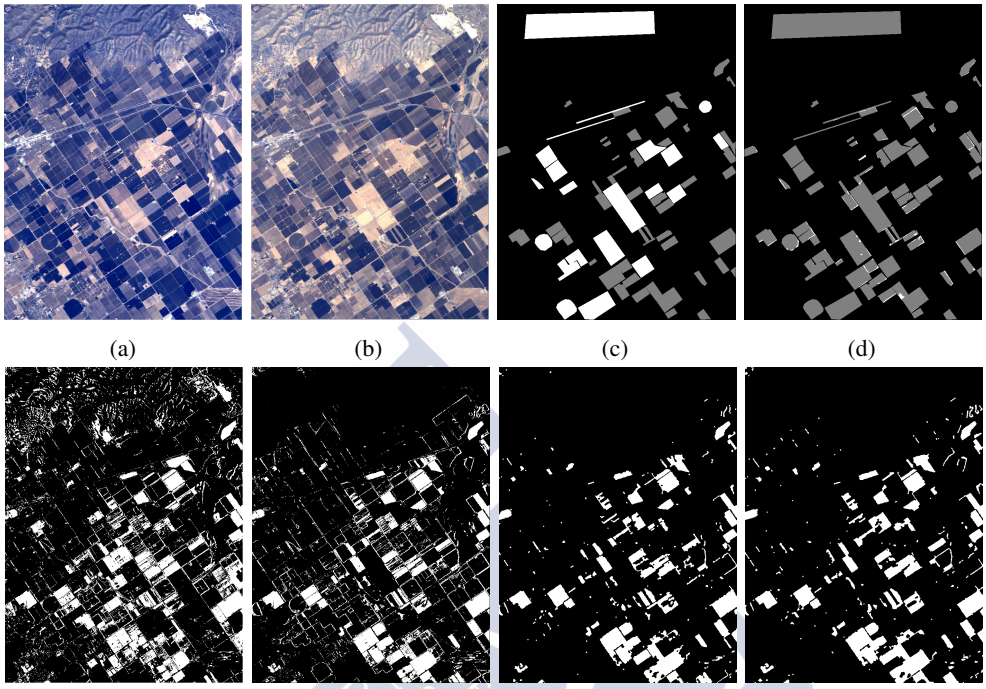


Figure 5.8: CD maps for the Santa Barbara dataset. Color composition of the input images (a, b), reference data of changes {white = no change, gray = change} (c), hit map for SAM+ECAS-II+Otsu {white = miss, gray = hit} (d), ED+Otsu CD map (e), SAM+Otsu CD map (f), SAM+watershed+Otsu CD map (g), and SAM+ECAS-II+Otsu CD map (h).

one of the images of the dataset. This situation corresponds to the step when the regions are averaged and to the fusion stage.

The hardware occupancy in GPU achieved for the most relevant kernels was analyzed using the *nvprof* tool provided by NVIDIA, and it is shown for the Santa Barbara dataset in Table 5.4 together with the GFLOPS achieved by each stage of the algorithm. The most costly kernels in terms of execution time are those involved in the calculation of the region averaging process (See Figure 5.4). These are executed in 512-thread blocks. The second kernel, which divides the accumulated values, achieves 41.2% occupancy as the grid size is too small to hide the operation latency. The kernel for the gradient calculation (See Figure 5.3) uses 2D blocks of threads of size 32×4 and achieves an occupancy of 49.4%, limited by the amount of shared

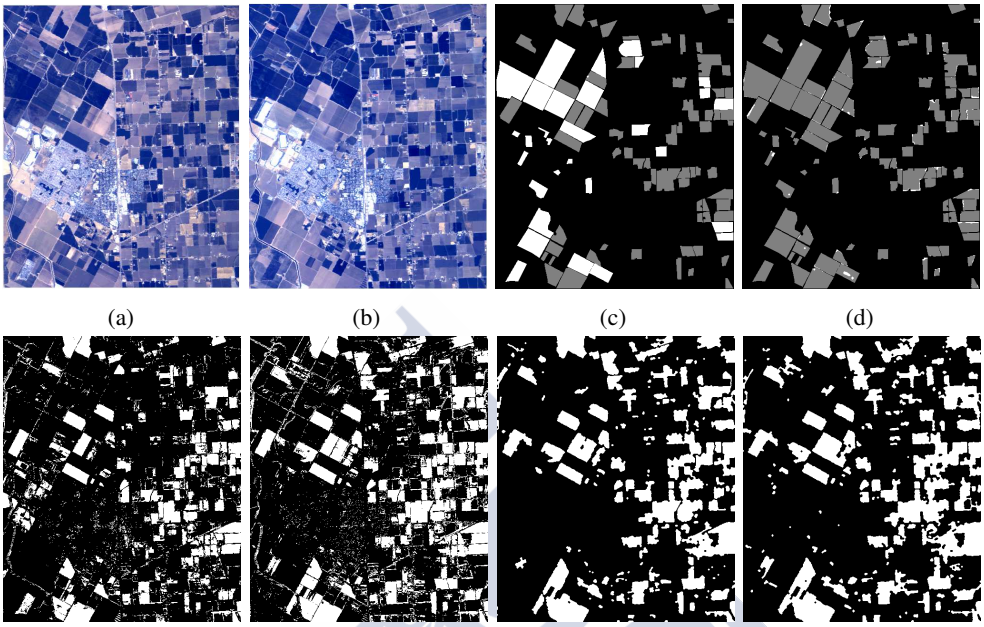


Figure 5.9: CD maps for the Bay Area dataset. Color composition of the input images (a, b), reference data of changes {white = no change, gray = change} (c), hit map for SAM+ECAS-II+Otsu {white = miss, gray = hit} (d), ED+Otsu CD map (e), SAM+Otsu CD map (f), SAM+watershed+Otsu CD map (g), and SAM+ECAS-II+Otsu CD map (h).

Dataset	**Santa Barbara**				**Bay Area**			
Stage	CPU	OpenMP	GPU	Speedup	CPU	OpenMP	GPU	Speedup
Gradients	4.838	1.441	0.022	65.4×	1.887	0.587	0.010	58.7×
Watershed	0.311	0.136	0.004	34.0×	0.229	0.095	0.003	31.6×
Average regions	2.651	0.937	0.029	32.3×	1.068	0.384	0.014	27.4×
Fusion	0.224	0.121	0.007	17.3×	0.094	0.050	0.003	16.7×
Thresholding	0.007	0.007	0.001	7.0×	0.003	0.003	0.001	3.0×
Regularization	1.352	0.984	0.015	65.6×	0.287	0.222	0.004	55.5×
Total	9.383	3.626	0.077	46.5×	3.568	1.341	0.034	38.3×

Table 5.3: Performance results in terms of execution time (in seconds) and speedups of the GPU scheme for CD with the SAM+watershed+Otsu configuration in the TITAN X GPU. CPU stands for a sequential implementation. OpenMP stands for an optimized parallel implementation using 4 threads. Speedup represents the factor by which the GPU version is faster than the OpenMP one.

Stage	Sub-stage	Time	GFLOPS	Memory utilization	Achieved occupancy	Theoretical occupancy
Gradients		0.022	89	35%	49.4%	50%
Watershed		0.004	–	35%	95.8%	100%
	Addition	0.017	–	85%	98.5%	100%
Average regions	Division	0.002	187	25%	41.2%	50%
	Propagation	0.010	–	65%	99.5%	100%
Fusion		0.007	148	85%	97.9%	100%
Thresholding		0.001	–	75%	65.9%	100%
Regularization		0.015	7	95%	82.0%	100%

Table 5.4: Performance results for the Santa Barbara dataset in terms of execution time (in seconds), achieved GFLOPS, memory utilization, and occupancy of the GPU scheme for CD with the SAM+watershed+Otsu configuration in the TITAN X GPU.

memory available (this information is provided by the tool *nvprof*). The next most relevant kernel is the one devoted to the regularization of the change map (See Figure 5.7) for the pseudocode of this stage) as it is an iterative kernel executed 46 times for the Santa Barbara dataset in this example. This kernel uses blocks of 512 threads achieving an occupancy of 82%. The limiting factor for this kernel, obtained by *nvprof*, is the memory bandwidth of the L2 cache memory. The kernel devoted to the fusion (See pseudocode in Figure 5.5) achieves an occupancy of 97.9%.

As can be seen in Table 5.4, most of the kernels are limited by the memory utilization (memory utilization above 75% in the table), which prevents the full exploitation of the compute resources. The kernels with enough computational load and not limited by the memory utilization are those that achieve more GFLOPS (the division kernel in the region averaging and the fusion kernel). It is important to note that some of the kernels (watershed, addition, propagation, and thresholding) execute only integer or single precision operations.

The execution time results for the ECAS-II-based configuration are shown in Table 5.5. It can be seen than the ECAS-II step is the most costly, since it involves the iterative execution of a CA to perform the segmentation, but it is the one achieving the best accuracies. Nevertheless, it is also the stage that achieves larger speedups as it is the one with the biggest computational load.

Dataset	Santa Barbara			Bay Area		
Stage	OpenMP	GPU	Speedup	OpenMP	GPU	Speedup
ECAS-II	14526.480	189.892	76.5×	6013.973	78.924	76.2×
Fusion	0.121	0.007	17.3×	0.050	0.003	16.7×
Thresholding	0.007	0.001	7.0×	0.003	0.001	3.0×
Regularization	0.984	0.015	65.6×	0.222	0.004	55.5×
TOTAL	14527.592	189.915	76.5×	6014.248	78.931	76.2×

Table 5.5: Performance results in terms of execution times (in seconds) and speedups of the GPU scheme for CD for the Santa Barbara and Bay Area datasets with the SAM+ECAS-II+Otsu configuration in the TITAN X GPU.

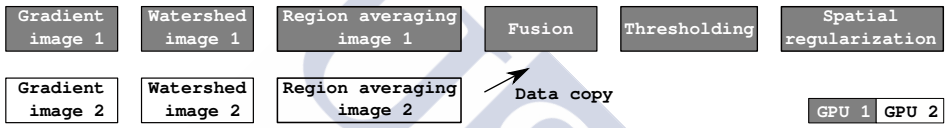


Figure 5.10: Multi-GPU CD flowchart.

5.4 Multi-GPU binary CD

Although the use of expensive and bulky computation infrastructures is beyond the scope of this thesis, multi-GPU systems are currently a constantly increasing trend. It is very realistic to assume the availability of multi-GPU architectures in a large part of commodity hardware in the near future. This can be achieved, for instance, by using technologies such as the Scalable Link Interface (SLI) that interconnects two or more GPUs to distribute the workload among them.

The multi-GPU experiments were carried out in a NVIDIA Tesla K80 from the FT2 cluster introduced in Section 2.3.1. OpenMP is used to manage the two GPUs in parallel. The experiments were run over the Santa Barbara dataset.

The scheme for binary CD proposed in this chapter includes the four stages shown in Figure 5.2. Several of the steps involved in this scheme are performed independently over each one of the input images. This makes it a good candidate to be projected into a multi-GPU system in order to achieve better execution times.

A multi-GPU version of the scheme following the flowchart of Figure 5.10 was developed to test the efficiency of multi-GPU systems for this task. The gradient calculation, watershed segmentation and region averaging processes for both images are executed in parallel

Stage	1 GPU	2 GPUs	Speedup
Gradients	0.076	0.038	2.0×
Watershed	0.014	0.007	2.0×
Average regions	0.053	0.030	1.8×
Memory Movements	–	0.004	–
Fusion	0.014	0.014	1.0×
Thresholding	0.002	0.002	1.0×
Regularization	0.022	0.022	1.0×
Total	0.181	0.117	1.5×

Table 5.6: Single and dual GPU performance results for the Santa Barbara dataset in terms of execution time (in seconds) and speedups of the GPU scheme for CD with the SAM+watershed+Otsu configuration in the NVIDIA Tesla K80 GPU.

in two independent GPUs, whereas the fusion, thesholding and spatial regularization steps are performed in a single GPU. This flowchart was designed aiming to minimize the data movements among GPUs. Once the region-averaged images are generated, the fusion process dramatically reduces the data size, decreasing the computational cost of the last stages of the algorithm. As a consequence, if the code were executed using both GPUs, the GPU computational load would not be enough to hide the access latency to the data. For this reason, the last three processes of CD are performed in only one GPU.

Table 5.6 shows the comparison between the execution times corresponding to a single GPU execution versus a dual GPU execution in the previously presented system, for the Santa Barbara dataset. The single GPU version (1 GPU in Table 5.6) is slower than the one presented in Table 5.3 because the NVIDIA Tesla K80 cores used in this experiment achieve 0.82 GHz, whereas the TITAN X cores used in Table 5.3 achieve 1.4 GHz. Table 5.6 shows that a speedup of 1.5× is achieved using two GPUs. The steps that have been parallelized represent 80% of the computation time in the single GPU version. Furthermore, the data loads are also 1.5× faster when two GPUs are employed as each input image can be loaded in parallel into the GPU memory.

5.5 Discussion

An efficient GPU scheme for the binary CD of multitemporal hyperspectral datasets was introduced in this chapter. The proposed scheme focuses on object-based CD and it incorporates segmentation techniques to achieve this goal. The spatial information of the objects in the

multitemporal data is included through a region averaging process. The region map is obtained through a gradient calculation followed by a watershed segmentation. Alternatively, the spatial processing can be tackled by the ECAS-II algorithm presented in Section 4.1.

After this, a fusion of the multitemporal information is carried out by a CVA processing. In this thesis, the SAM distance is proposed as a better alternative for performing this fusion than the ED used in the traditional CVA method. This process provides an intensity difference image.

The next stage of the technique obtains a binary CD map by performing a thresholding of the difference image following the Otsu method. Finally, a spatial regularization based on the closest neighborhood pixels is performed to improve the binary CD map.

The scheme has been fully projected to GPU in order to achieve efficient execution times allowing the use of the scheme in real time scenarios. The main differences between the GPU and CPU versions of the scheme, in order to fully exploit the capabilities of each architecture have also been detailed.

The scheme has been evaluated with two multitemporal hyperspectral datasets, achieving above 97% of correctly classified pixels in both cases. From the results, it can be seen that the SAM distance outperforms the results achieved for the corresponding scheme with ED in all the cases. Different thresholding alternatives were compared, showing that the Otsu method is an adequate alternative for achieving accurate change/no change discrimination. Finally, the segmentation-based inclusion of the spatial information proposed always improves the results based only on the spectral information.

Regarding the scheme execution times, the complete scheme, with the watershed-based segmentation, can be executed in GPU in less than 0.1 seconds for the considered datasets (0.077s and 0.034s for Santa Barbara and Bay Area, respectively). All the stages achieve large speedups in their GPU projection, except the thresholding stage, whose computational load is too small and, therefore, it does not affect the total execution time or speedup. The version of the scheme with the ECAS-II-based segmentation, also achieves large speedups (up to 76.5$\times$ faster in GPU) but the total execution times are bigger because the ECAS-II step is more computationally costly than the watershed based alternative. Nevertheless, ECAS-II is the best alternative for achieving the best change detection accuracies. Finally, a study of the use of the available GPU capabilities has also been carried out to show how the correct exploitation of the hardware allows us to achieve the best implementation (and therefore execution times) possible.

Finally, the last section of this chapter was devoted to showing the projection of the proposed scheme to a multi-GPU system. The objective is to exploit the capabilities of this kind of systems to further improve the execution time of the scheme. To this end, the stages of the scheme that are computed independently in the two input images are executed in parallel in two GPUs, these being the most computationally costly stages. The partial results of the second GPU are then moved to the first one, which executes the remaining stages. This multi-GPU scheme allows us to achieve an additional 1.5$\times$ speedup over the single GPU version of the scheme.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

MULTICLASS CHANGE DETECTION FOR MULTITEMPORAL DATASETS ON GPU

This chapter presents the multiclass CD scheme for hyperspectral images developed in this thesis. The proposed scheme is based on the combination of the binary CD scheme proposed in Chapter 5 with a direct multidate supervised classification technique that performs the multiclass discrimination.

The first section of the chapter is devoted to the introduction of the proposed scheme. The multiclass detection relies on the use of SAEs to deal with the extraction of the change features of the multitemporal dataset. Other alternatives, such as PCA or NWFE, are also considered for the FE of the change information. The scheme also considers the spatial information through the use of EMP. The efficient ELM classifier in GPU presented in Chapter 3 is used to perform the final *from-to* transition classification stage, and the classification results thereof are also compared to those obtained by an SVM. The next section of the chapter introduces the GPU implementations applied in this scheme that have not previously been introduced in other schemes presented in this thesis.

The remaining sections of the chapter introduce the achieved experimental results. Results for two different multiclass CD hyperspectral scenarios are provided, along with the application of the scheme to a multispectral dataset extensively used in the literature. Finally, the robustness of the scheme in noisy scenarios is also tested.

6.1 Multiclass CD scheme

The complete scheme for the multiclass CD in multitemporal hyperspectral datasets is introduced in this section. The stages of this scheme are shown in Figure 6.1. It combines the binary CD detailed in the previous chapter (at the top of Figure 6.1) with a direct multidate supervised classification performed over an image with enhanced change features (at the bottom of Figure 6.1) which performs the multiclass detection. The combination of the two branches is performed through the filtering of the enhanced change feature image with the binary CD map (binary filtering in Figure 6.1).

The binary branch of the multiclass CD is efficiently performed through the scheme explained in Section 5.2. The binary CD map obtained through this processing is used here as a filtering mask to select the change pixels to be classified in the last stage of the scheme.

Regarding the multiclass branch of the CD scheme (bottom part of Figure 6.1), it consists of a change feature processing followed by a direct multidate classification. First, the multitemporal information of the two input images is combined through the fusion of the two images. This fusion can be performed in different ways. The most simple options are calculating the point-by-point difference between the input images or stacking all the multitemporal information in a single image. In this last case, the spectral size will be the sum of the spectral sizes of the input images. The stack option also allows us to perform the CD in datasets whose images do not share the same spectral size.

Once the multitemporal information is combined into a single image, an FE technique is computed to extract the relevant features of the resulting image, reducing its dimensionality. FE can be tackled with different techniques [62], for instance, PCA [153], K-PCA [75], Transfer Component Analysis (TCA) [76] (an FE method specifically designed to domain adaptation problems), ICA [222], NWFE [79], or SAE [86, 85]. Recently, the use of deep learning techniques has proven to be a good alternative for reducing the dimensionality of hyperspectral datasets while highlighting the relevant features [81, 82, 85], so the use of SAE to perform FE for CD is proposed in this thesis.

The next stage aims to introduce the spatial information contained in the dataset into the scheme. An EMP is calculated to this end (See Section 1.5). The EMP is effective in extracting spatial information for classification purposes [66]. In particular, it extracts information about the contrast and size of the structures present in the multidimensional image. SEs of increasing size are required in order to construct the profile. After applying the EMP to the

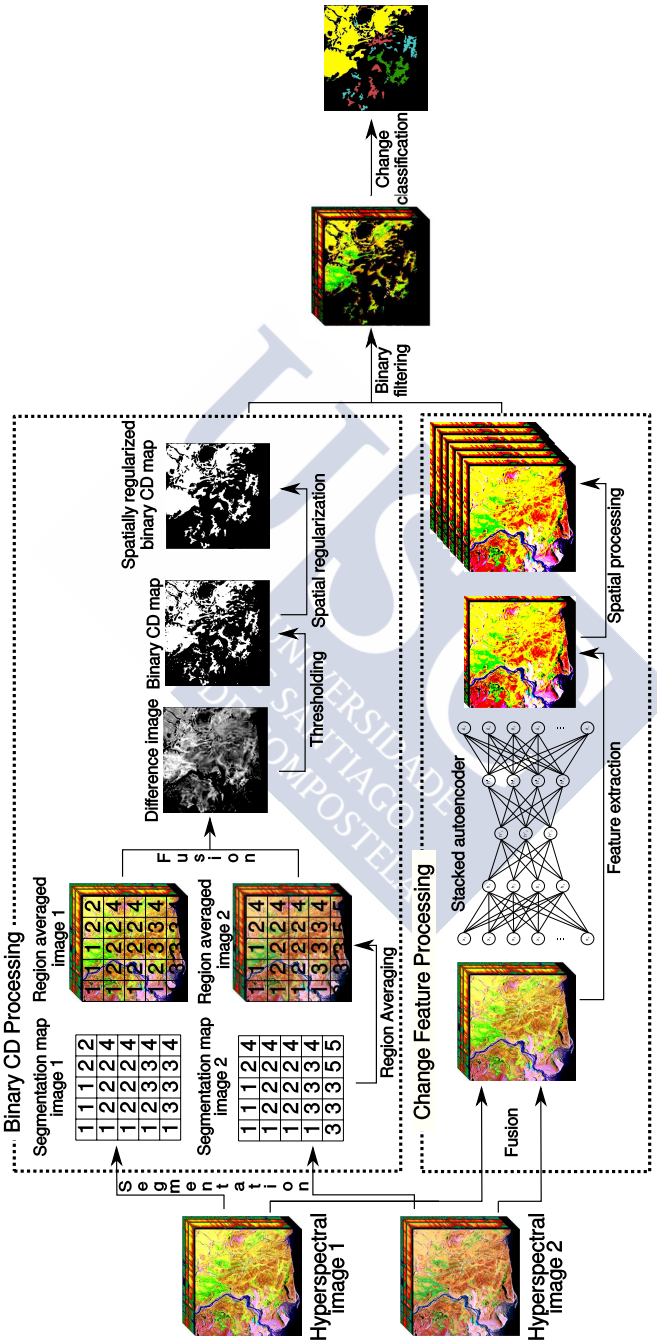


Figure 6.1: Multiclass CD flowchart.

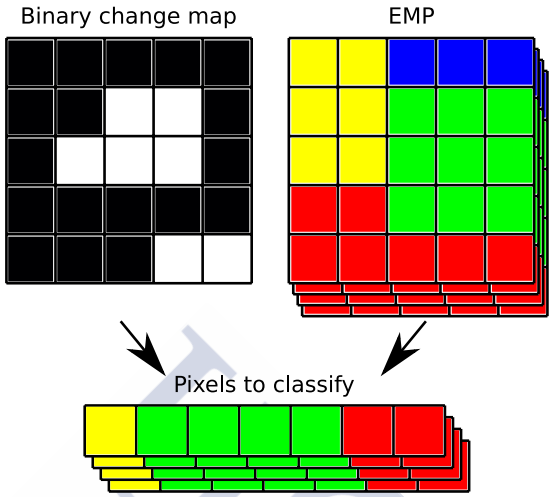


Figure 6.2: Multiclass CD binary filtering.

reduced dimensionality image, an image with $(2k+1)$ components for each band preserved, where k is the number of SE sizes considered, is obtained.

Once the change feature processing is done, and the binary CD map is available, the results are combined through a filtering process, as shown in Figure 6.2, whereby all the pixels that have been considered as no change pixels in the binary CD (black color) are removed from the image. This filtering provides advantages both in the learning of the later classifier and in terms of computational efficiency: The filtering allows us to avoid the learning of the no change class, which is the one with a greater variance in its characteristics, and leads to a smaller number of pixels to classify.

Finally, a supervised classification by ELM is applied to the filtered image in order to obtain the final multiclass change classification map. An SVM classification is also used for comparison purposes. The ELM classifier is the one introduced in Chapter 3 for the efficient classification of hyperspectral datasets in GPU.

6.2 Multiclass CD scheme on GPU

This section is devoted to the introduction of the GPU implementations of those techniques introduced in some stage of the proposed multiclass CD scheme that have not previously been introduced in this thesis. In particular, the PCA, SAE, and EMP implementations are

detailed. Some pseudocodes are used to introduce these implementations, where each process executed in GPU is placed between <> symbols and may involve one or more kernels. The pseudocodes also include the GM and SM acronyms to indicate kernels executed in global memory or shared memory, respectively.

6.2.1 FE by PCA on GPU

Figure 6.3 shows the main steps needed to efficiently compute the PCA on GPU following the EVD approximation. First, a pre-processing stage (lines 1-3 on the pseudocode) is needed where the dataset ($\mathbf{X}$) is centered by subtracting from each pixel the mean of the coefficients of the corresponding band. For this purpose, a vector $\mathbf{y}$ is created, with all its components equal to 1, and whose size is equal to the number of pixels of the image. Then, the *cublasSgemv* CUBLAS function is used to perform the matrix-vector product $\mathbf{z} = \mathbf{X^T} \times \mathbf{y}$, obtaining a vector in which each value corresponds to the sum of all pixels of one band of the original data. Finally, the values of $\mathbf{z}$ are used in a kernel that performs the centering of the data, where each thread computes the value of a pixel for all the bands.

The PCA stage (lines 4-7) starts by calculating the correlation matrix of $\mathbf{X}$ with the *cublasSsyrk* function that performs the $\mathbf{XX^T}$ operation. The function only computes the upper triangle of the result, as the bottom triangle is symmetric. So, the next kernel is devoted to completing this result, after which the EVD is computed with the *culaDeviceSgesvd* CULA function. Finally, the PCA is obtained through the *cublasSgemm* function, which calculates the product between $\mathbf{X}$ and $\mathbf{V}$.

6.2.2 FE by SAE on GPU

The SAE FE is efficiently computed in GPU thanks to the use of the 1.0.0-rc3 version of the *Caffe* framework [204]. The main characteristics of the structure of an SAE were introduced in Section 1.3.4. A pseudocode with the main stages and kernels involved in this computation can be seen in Figure 6.4. It consists of an iterative process that alternates *Forward* and *Backward* propagation. The *Forward* propagation generates a new compressed representation of the data with the current weight values through the computation of inner products and activation functions (lines 2-5 in Figure 6.4) and the *Backward* propagation aims to reconstruct the original data performing the same operations in opposite order (lines 9-12) . After each iteration, the weight values are updated (line 13) taking into account the value returned by

EVD-PCA on GPU.
Input: Hyperspectral image **X**.
Output: PCA FE dimensionality reduced image.
Dataset centering stage
1: <Create the vector **y** of 1's> ▷ GM
2: <Obtain the sum of all pixels in each band of dataset> ▷ SM + GM
3: <Dataset centering> ▷ SM
PCA stage
4: <Calculate the covariance matrix $\mathbf{XX^T}$ of centered dataset> ▷ SM + GM
5: <Complete the lower triangular values of the covariance matrix > ▷ GM
6: <Calculate the matrix of eigenvectors **V** > ▷ SM + GM
7: <Obtain the PCs by multiplying **XV** > ▷ SM + GM
▷ GM: Global Memory, SM: Shared Memory.

Figure 6.3: Pseudocode for the PCA FE on GPU.

the loss function (lines 6-8 in Figure 6.4). The *Forward* and *Backward* processes will loop in each iteration as many times as the number of layers selected for the SAE.

6.2.3 EMP on GPU

The mathematical operators used for the construction of the EMP are the opening and closing by reconstruction. Each MP keeps the spatial structures contained in the image if the corresponding SE fits within them, otherwise the structures are removed. The details of the EMP were explained in Section 1.5. The EMP is efficiently computed in GPU as presented in [200]. The erosion and dilation operations (for the calculus of the *infimum* and *supremum* intensity value of the pixels covered by a SE, respectively) are computed by exploiting the capabilities of the NPP library (optimization technique 5 in Section 2.2). Moreover, a block-asynchronous updating process is applied to the EMP to speedup the morphological reconstruction (optimization technique 3 in Section 2.2). The pseudocode of this implementation is displayed in Figure 6.5.

As can be seen in the pseudocode, each band of the input image (the enhanced change features, dimensionality reduced image) is independently processed to create the corresponding MP. Each component of each MP is computed with a series of consecutive kernels (lines 3-9 in Figure 6.5) and stacked to form the final EMP of the image. The kernels for the erosion and dilation of each band are computed with the NPP library and followed by reconstruction kernels in shared memory to produce the opening and closing operators. Kernels in GM to

SAE on GPU.
Input: Hyperspectral image.
Output: SAE FE dimensionality reduced image.
1: **for each** iteration **do**
Forward
2: **for each** layer *n* of **SAE do**
nth Inner product calculation
3: InnerProductLayer::Forward_gpu() → <caffe_gpu_gemm()> ▷ SM + GM
nth Inner activation
4: CuDNNSigmoidLayer::Forward_gpu() → <cudnnActivationForward()> ▷ GM
5: **end for**
SoftMax with Loss
6: CuDNNSoftmaxLayer::Forward_gpu() → <cudnnSoftmaxForward()> ▷ SM + GM
7: SoftmaxLossForwardGPU() → <SoftmaxLossForwardGPU> ▷ GM

Backward
8: SoftmaxLossBackwardGPU() → <SoftmaxLossBackwardGPU> ▷ GM
9: **for each** layer *n* of **SAE do**
nth Inner Activation
10: CuDNNSigmoidLayer::Backward_gpu() → <cudnnActivationBackward()> ▷ GM
nth Inner product calculation
11: InnerProductLayer::Backward_gpu() → <caffe_gpu_gemm()> ▷ SM + GM
12: **end for**
Weights update
13: caffe::SGDSolver() → <SGDUpdate> ▷ GM
14: **end for**
▷ GM: Global Memory, SM: Shared Memory.

Figure 6.4: Pseudocode for the SAE FE on GPU.

calculate the complement of the corresponding image are interleaved during the computation in order to use the appropriate input for each kernel in the process, because the computation of the closing is performed with the same kernels used for the opening but applied to the complement of the original input.

6.3 Multiclass hyperspespectral CD results

The results obtained by the application of the proposed scheme in two hyperspectral datasets are detailed in this section. The Hyperion and Synthetic 1 hyperspectral datasets introduced in Section 2.3.3 are used for the evaluation of the multiclass CD scheme.

EMP on GPU.
Input: Hyperspectral image $\mathbf{X}$.
Output: EMP.
for each band k of $\mathbf{X}$ **do**
  for each size s of SE **do**
    <Erode $\mathbf{X}_k$ with a SE of size s > ▷ NPP
    <Reconstruction (opening by reconstruction)> ▷ SM
    <Complement $\mathbf{X}_k$ > ▷ GM
    <Dilate $\mathbf{X}_k$ with a SE of size s > ▷ NPP
    <Complement the output of the dilation > ▷ GM
    <Reconstruction (closing by reconstruction)> ▷ SM
    <Complement the output of the reconstruction > ▷ GM
  end for
end for
▷ GM: Global Memory, SM: Shared Memory, NPP: NVIDIA Performance Primitive library.

Figure 6.5: Pseudocode for the EMP calculation on on GPU.

Among the different configurations of the binary branch of the scheme presented in Section 5.2.1, the one based on fusion of the information by calculating the difference by means of SAM distance [104] and Otsu's thresholding [103], including the spatial information through a watershed segmentation [159], is selected for the experiments. As shown in the previous section, this configuration is the one that achieves the best trade off solution between accuracy of the results and moderate execution time (See Tables 5.1 and 5.2).

For the multiclass branch of the CD, different configurations of the scheme are compared. The difference and stack of the input images are evaluated as fusion methods of the multitemporal information for both datasets in order to compare the discrimination capability of the change features obtained by both methods. The FE by SAEs is configured to reduce the dimensionality of the dataset to 12 bands and it is compared to the FE obtained by NWFE and PCA, retaining the same number of components.

Thus, the SAE has been configured with two layers that reduce the dimensionality of the data from the original spectral size to 12 features. 20% of the pixels were randomly chosen and used for training the SAE with the following configuration of parameters, that was experimentally selected. A batch of 64 pixels per iteration and a limit of 300,000 iterations. The back propagation process uses a Stochastic Gradient Descent (SGD) and the *'inv'* learning rate policy corresponding to

$$inv = base\_lr * (1 + \gamma * i)^{(-power)}, \tag{6.1}$$

where i is the iteration number and with a base learning rate (*base_lr*) of 0.01, and values for the parameters γ and *power* of 0.0001 and 0.75 respectively.

The effect of calculating an EMP to the CD accuracies is also studied in all the cases. An EMP size of 180 is selected, corresponding to the application of 7 SE sizes to each one of the 12 bands obtained by FE. This EMP size was selected as the one that maximizes the accuracies with a lower computational load after an exploration of the accuracies achieved with different EMP sizes.

The ELM and SVM classifiers were trained with 5% of the reference data available for each class in all the cases. This can be considered a low number of training samples [223]. The samples selected for training are randomly chosen in each run and the remaining labeled samples of the change reference map are used for testing. The classification by SVM is carried out using the LIBSVM library and the Gaussian RBF as the activation function. The parameters of each classifier (C and σ for SVM, and number of neurons in the hidden layer for the ELM, as explained in Sections 1.6.1 and 1.6.2) have been optimized independently for each different configuration of the multiclass CD scheme considered in order to maximize the achieved accuracies.

6.3.1 Accuracy results for the Hermiston dataset

Figure 6.6 shows the Hyperion dataset as well as the results obtained by using different configurations of the CD technique. The graphical results will be explained throughout this section.

Tables 6.1 and 6.2 show the accuracy results of the proposed binary branch of the CD scheme for the Hermiston dataset. 98.75% of the total pixels of the image were correctly discriminated as change or no change. The number of no change pixels that pass the binary filtering, and therefore are classified as changes, corresponds with the FAs number in Table 6.1. The MAs produced in the binary CD are distributed between the 5 classes of change as shown in Table 6.2. The last column of Table 6.2 indicates the percentage of change pixels of each class that were misclassified in the binary CD process. Finally, it can be checked that the total number of filtered pixels (9942) is equal to the real number of change pixels (9986) plus the FAs (467) minus the MAs (511).

Figure 6.6 (a) and (b) are color compositions of the two input images, while Figure 6.6 (c) shows the change reference map as described in Section 2.3.3. As can be seen in Figure 6.6 (c) and (d), the binary CD map (c) corresponds at object level with the changes represented in the reference data (d). Most of the errors in this branch of the CD processing correspond

Configuration	Correct		MAs	FAs	Total Error	
SAM+watershed+Otsu	77022	(98.75%)	511	467	978	(1.25%)

Table 6.1: Accuracy results for binary CD (correctly classified pixels, MAs, FAs, and total error), in terms of number of pixels and percentages, for the Hermiston dataset with the SAM+watershed+Otsu configuration.

Class	Number of pixels		Filtered pixels		MAs	% of total MAs	% MAs per class
0	68014	(87.20%)	467	(4.70%)	–	–	–
1	5558	(7.13%)	5246	(52.77%)	312	61.06%	5.61%
2	1331	(1.71%)	1245	(12.52%)	86	16.83%	6.46%
3	79	(0.10%)	79	(0.79%)	0	0.00%	0.00%
4	1557	(2.00%)	1545	(15.54%)	12	2.35%	0.77%
5	1461	(1.87%)	1360	(13.68%)	101	19.77%	6.91%
TOTAL	78000	(100%)	9942	(100%)	511	100%	–

Table 6.2: Binary filtering details for the Hermiston dataset (binary CD with SAM+watershed+Otsu configuration).

to misclassified pixels on the edges between objects as can be checked in Figure 6.6 (e). This is reasonable because the possible errors in isolated pixels are avoided as a consequence of the averaging process performed during the binary CD stage (See Section 5.3, in particular Figures 5.8 (e), (f), (g), and (h) and 5.9 (e), (f), (g), and (h)) thus avoiding errors in those pixels. This explains why the percentage of MAs shown in Table 6.2 is larger in those classes that include a bigger number of objects in the datasets (i.e., the classes that include a bigger number of border pixels).

Table 6.3 shows the accuracies achieved in the complete multiclass CD of the Hermiston dataset. Only those pixels that passed the binary filtering are considered in this stage (corresponding to the 'Correct' column in Table 6.1). As can be observed in the table, the SVM and ELM classifiers perform similarly in all the configurations of the CD scheme, with the ELM classification being slightly better in most of the cases. Regarding the different fusion approaches, it can be seen that for the same configurations of the remaining stages of the scheme, the fusion through the stack of the input images performs better than the fusion through the difference of those images. This makes sense as the stack maintains all the original information while equal difference values can be achieved from different input signatures.

Table 6.3 also shows that, regarding the FE technique, the SAE outperforms NWFE and PCA in all the considered cases, achieving accuracy results of up to 3 percentage points better

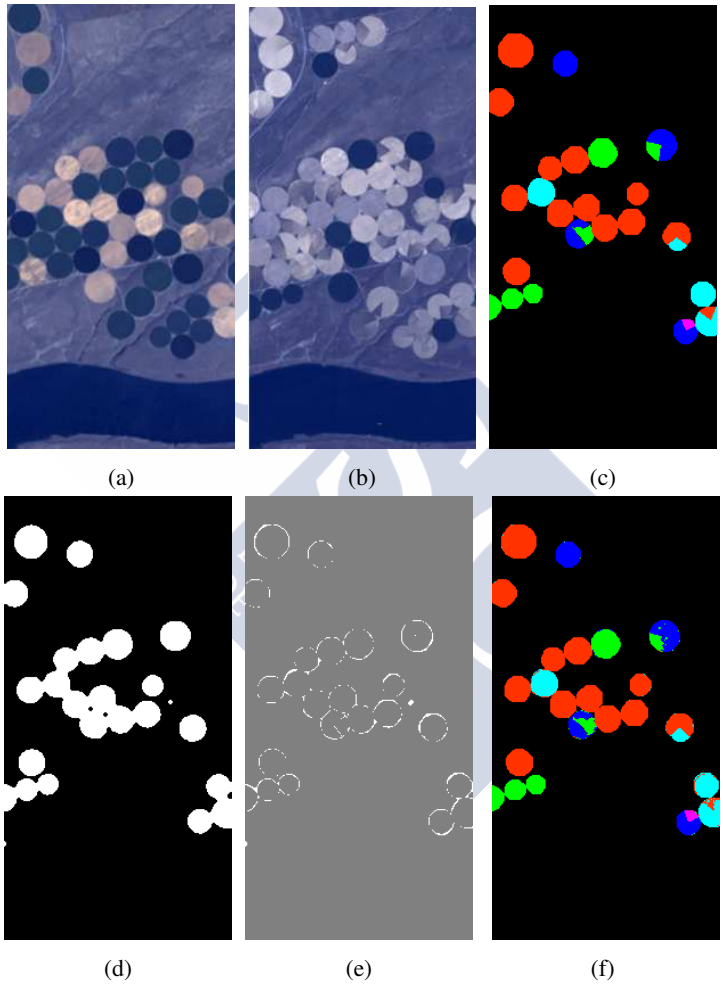


Figure 6.6: Binary and multiclass CD maps for the Hermiston dataset. Color composition of the input images (a, b), reference data of changes (c), binary CD map {white = change, black = no change} (d) , hit map for SAM+watershed+Otsu {white = miss, gray = hit} (e), and best ELM multiclass CD map (Stack+SAE+EMP) (f).

Fusion	FE	Spatial processing	SVM					ELM			
			Parameters		OA	AA	k	Parameters	OA	AA	k
			C	γ				N			
Difference	PCA	–	32	32	92.24	71.76	87.64	120	92.27	76.78	87.69
	NWFE	–	32768	8	91.01	82.97	85.79	130	92.30	79.18	87.77
	SAE	–	32	4	92.91	87.98	88.68	80	95.75	92.09	93.25
Stack	PCA	–	256	32	93.29	85.19	89.47	160	94.13	85.28	90.66
	NWFE	–	2048	32	93.46	85.10	89.59	130	94.19	86.19	90.77
	SAE	–	32	0.5	96.32	92.30	94.10	80	96.63	88.99	94.63
Difference	PCA	EMP	256	1	94.20	81.10	90.80	170	94.74	78.47	91.64
	NWFE	EMP	32	16	93.96	86.50	90.40	180	95.13	81.71	92.25
	SAE	EMP	1024	0.03125	97.17	97.40	95.51	110	97.72	97.39	96.39
Stack	PCA	EMP	64	16	95.51	87.50	92.80	160	95.31	86.22	92.54
	NWFE	EMP	64	0.0625	95.05	86.22	92.19	180	95.57	86.98	92.95
	SAE	EMP	64	0.03125	97.44	**97.60**	95.95	160	**97.89**	97.08	**96.66**

Table 6.3: Multiclass CD accuracies for the Hermiston dataset.

in terms of OA. Furthermore, the larger improvement of the SAE FE is in terms of AA, improving the other FE methods by up to 16 percentage points, allows us to conclude that SAE performs a better FE for this problem.

Finally, the application of EMP to include the spatial information in this branch of the CD scheme improves the accuracy of the configurations without EMP in all the cases. This improvement is greater when the baseline accuracy is lower.

The best multiclass results in the Table are achieved for the Hermiston dataset with the Stack + SAE + EMP configuration and this is also shown in Figure 6.6 (f). If a comparison of the obtained map with the reference data available in Figure 6.6 (c) is performed, it can be seen that all the objects detected as changes have been assigned to the correct class and the small amount of misclassified pixels mainly corresponds to edges among regions.

6.3.2 Accuracy results for the Synthetic 1 dataset

The same analysis applied to the Hermiston dataset is also carried out for the Synthetic 1 dataset. This dataset is useful for testing the proposed multiclass CD scheme in a scenario with a higher number of change classes. Moreover, the synthetic dataset is also conducive to the availability of an accurate reference data of changes, so that the presence of wrongly

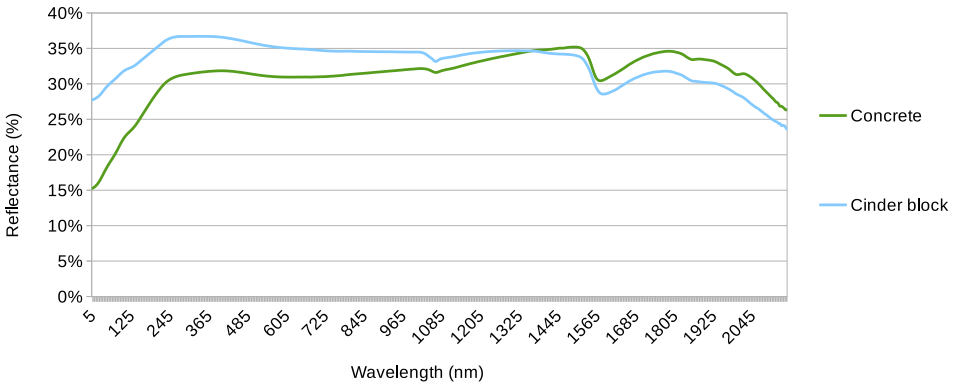


Figure 6.7: Spectral signatures of cinder blocks and concrete, provided by the USGS [133] spectral library *splib06a* [134].

Configuration	Correct		MAs	FAs	Total Error	
SAM+watershed+Otsu	752509	(95.56%)	20940	14007	34947	(4.44%)

Table 6.4: Accuracy results for binary CD (correctly classified pixels, MAs, FAs, and total error), in terms of number of pixels and percentages, for the Synthetic 1 dataset with the SAM+watershed+Otsu configuration.

misclassified pixels in border regions due to failures in the precision of the reference data is avoided.

Tables 6.4 and 6.5 show the detailed results of the binary branch of the CD scheme for the Synthetic 1 dataset. In this case, the binary CD achieves 95.56% of correctly classified pixels. An analysis of the per-class binary filtering results of Table 6.5 shows that most of the misclassified pixels correspond to 4 classes whose pixels are are almost entirely MAs (classes 2, 19, 20, and 23). These classes, as introduced in Table 2.11, correspond to transitions between objects composed by the same or very similar materials, which explains the occurrence of a misclassification when the discrimination is based on the spectral signature similarity of the objects. For instance, class 19 represents the transition of an area from cinder blocks to concrete, two classes with similar spectral signatures, as shown in Figure 6.7.

Figure 6.8 shows the same results than 6.6, for the Synthetic 1 dataset. Figure 6.8 (a) and (b) show the color composition of the input images. Figure 6.8 (c) shows the change reference information. Figure 6.8 (d) shows the binary CD map obtained, with most of the objects in the

Class	Number of pixels		Filtered pixels		MAs	% of total MAs	% MAs per class
0	650009	(82.55%)	14007	(10.73%)	–	–	–
1	2441	(0.31%)	2296	(1.76%)	145	0.69%	5.94%
2	4761	(0.60%)	0	(0.00%)	4761	22.74%	100.00%
3	9072	(1.15%)	7939	(6.08%)	1133	5.41%	12.49%
4	2250	(0.29%)	2210	(1.69%)	40	0.19%	1.78%
5	9086	(1.15%)	8866	(6.79%)	220	1.05%	2.42%
6	4493	(0.57%)	4455	(3.41%)	38	0.18%	0.85%
7	2758	(0.35%)	2648	(2.03%)	110	0.53%	3.99%
8	2288	(0.29%)	2239	(1.72%)	49	0.23%	2.14%
9	3587	(0.46%)	3412	(2.61%)	175	0.84%	4.88%
10	2436	(0.31%)	2262	(1.73%)	174	0.83%	7.14%
11	7935	(1.01%)	7663	(5.87%)	272	1.30%	3.43%
12	2479	(0.31%)	2363	(1.81%)	116	0.55%	4.68%
13	2408	(0.31%)	2198	(1.68%)	210	1.00%	8.72%
14	5917	(0.75%)	5912	(4.53%)	5	0.02%	0.08%
15	2759	(0.35%)	2704	(2.07%)	55	0.26%	1.99%
16	2714	(0.34%)	2612	(2.00%)	102	0.49%	3.76%
17	5290	(0.67%)	5159	(3.95%)	131	0.63%	2.48%
18	3629	(0.46%)	3598	(2.76%)	31	0.15%	0.85%
19	2363	(0.30%)	0	(0.00%)	2363	11.28%	100.00%
20	2571	(0.33%)	6	(0.00%)	2565	12.25%	99.77%
21	2545	(0.32%)	2510	(1.92%)	35	0.17%	1.38%
22	11125	(1.41%)	10417	(7.98%)	708	3.38%	6.36%
23	5915	(0.75%)	233	(0.18%)	5682	27.13%	96.06%
24	5780	(0.73%)	5630	(4.31%)	150	0.72%	2.60%
25	3336	(0.42%)	3291	(2.52%)	45	0.21%	1.35%
26	2750	(0.35%)	2685	(2.06%)	65	0.31%	2.36%
27	4543	(0.58%)	4444	(3.40%)	99	0.47%	2.18%
28	3209	(0.41%)	2533	(1.94%)	676	3.23%	21.07%
29	2415	(0.31%)	2310	(1.77%)	105	0.50%	4.35%
30	4043	(0.51%)	3681	(2.82%)	362	1.73%	8.95%
31	5296	(0.67%)	5229	(4.01%)	67	0.32%	1.27%
32	5253	(0.67%)	5002	(3.83%)	251	1.20%	4.78%
TOTAL	787456	(100%)	130514	(100%)	20940	100%	–

Table 6.5: Binary filtering details for the Synthetic 1 dataset (binary CD with SAM+watershed+Otsu configuration).

reference data of Figure 6.8 (c) correctly detected. As can be seen in Figure 6.8 (e), the errors in the binary CD stage correspond mainly to 5 large areas. The 4 areas that are located in the central part of the image correspond to classes 2, 19, 20, and 23, as explained in the previous paragraph. The remaining one is a rectangle area near the right-hand border of the image that comprises most of the FAs. The spatial regularization based on the closest neighborhood used to remove disconnected pixels in the binary CD, produces some misclassified border pixels, but it improves the general accuracy of the CD map. The multiclass CD map shown in Figure 6.8 (f) displays how all the areas selected as change are correctly assigned in the multiclass branch of the CD scheme to the correct transition between classes regarding the available reference data (Figure 6.8 (c)).

The multiclass CD accuracies achieved with the proposed CD scheme for the Synthetic 1 dataset are displayed in Table 6.6. Note that the percentages in this table are calculated only over the pixels that have passed the binary CD stage. In this case, the accuracy values are very high in all the configurations, which makes it more difficult to establish an order of quality among the different techniques used in each stage of the CD scheme. Nevertheless, in general, the same conclusions extracted for the Hermiston dataset can be applied here. The fusion through the stack of the multitemporal information achieves better CD results than the difference between the input images. The use of SAEs for FE achieves competitive results as compared with PCA and NWFE. The application of EMPs to introduce the spatial information in the multiclass CD improves the accuracy results. Finally, the ELM classifier is a valid alternative to the traditional SVM in terms of accuracy for the supervised classification of this kind of images.

6.3.3 Execution time and speedup results on GPU

The NVIDIA GTX TITAN X introduced in Section 2.3.1 is used to measure the performance results in GPU. Table 6.7 shows, in execution time and speedups, the effects of the application of the binary filtering before the multiclass CD for both images. The selected configuration for this purpose is as follows: the binary branch of the CD scheme is performed through region averaging based on watershed segmentation, SAM distance computation, and Otsu's thresholding followed by a spatial regularization. The change feature processing is performed through feature fusion as the pixel by pixel difference of the input images, PCA FE and EMP computation. The FE by means of PCA has been selected here in order to show the fastest configuration of the multiclass CD scheme, as the PCA processing is faster than the SAE FE.

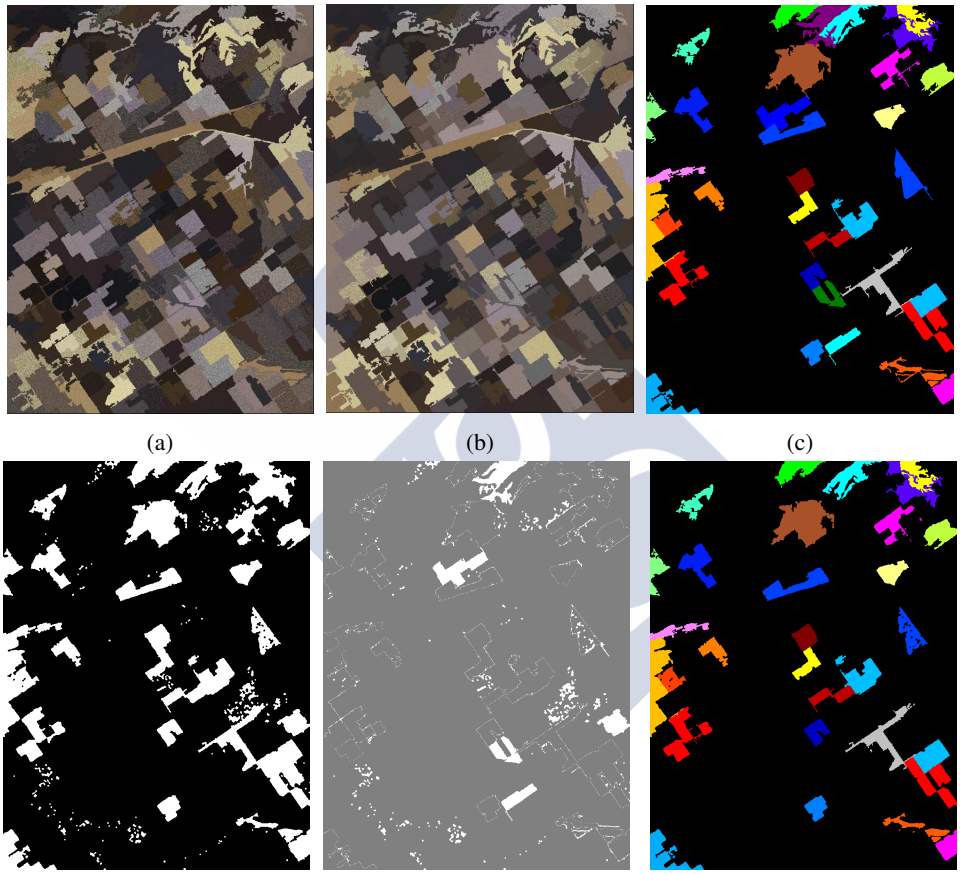


Figure 6.8: Binary and multiclass CD maps for the Synthetic 1 dataset. Color composition of the input images (a, b), reference data of changes (c), binary CD map {white = change, black = no change} (d) , hit map for SAM+watershed+Otsu {white = miss, gray = hit} (e), and best ELM multiclass CD map (f).

Fusion	FE	Spatial processing	SVM				ELM			
			Parameters		OA	AA	k	Parameters	OA	AA	k
			C	γ				N			
Difference	PCA	–	64	32	98.66	97.10	98.60	410	95.84	88.10	95.63
	NWFE	–	32	32	99.48	99.40	99.49	400	98.49	93.80	98.41
	SAE	–	65536	2	97.15	95.00	97.00	290	97.04	90.85	96.90
Stack	PCA	–	32	1	99.95	96.60	99.90	450	99.97	98.83	99.97
	NWFE	–	64	1	99.99	100.00	100.00	340	99.98	99.44	99.98
	SAE	–	32	1	99.97	98.70	100.00	760	99.98	98.01	99.97
Difference	PCA	EMP	256	1	99.99	98.60	100.00	800	99.96	96.50	99.96
	NWFE	EMP	128	0.125	99.98	100.00	100.00	720	99.98	99.96	99.98
	SAE	EMP	32	0.125	99.98	98.60	100.00	630	99.99	98.17	100.00
Stack	PCA	EMP	32	1	**100.00**	**100.00**	**100.00**	500	**100.00**	**100.00**	**100.00**
	NWFE	EMP	32	1	**100.00**	**100.00**	**100.00**	320	**100.00**	**100.00**	**100.00**
	SAE	EMP	32	1	**100.00**	98.00	**100.00**	750	**100.00**	99.60	**100.00**

Table 6.6: Multiclass CD accuracies for the Synthetic 1 dataset.

Finally, the filtered feature enhanced image is classified with an ELM classifier. When the filtering with the map obtained by the binary CD is applied, the execution time of the ELM is greatly improved as the number of pixels to classify is largely reduced. In the most common scenarios, the number of change pixels will always be smaller than the number of no change pixels. Thus, the ELM is computed 4.3× and 4.8× faster for the Hermiston and Synthetic 1 datasets in CPU, respectively, when the filtering is performed. The speedup is sufficient to hide the computational load of the binary CD branch of the scheme introduced in Figure 6.1 along with the binary filtering itself. Regarding the execution times in GPU, the speedup of the ELM classification of the filtered pixels with respect to the ELM classification of the entire image is smaller, given that a smaller number of pixels to classify implies less exploitation of the GPU resources. Nevertheless, it achieves speedups of 1.2× and 2.6× for the Hermiston and Synthetic 1 datasets, respectively, in the total time needed to compute the binary CD and filtered ELM classification.

Table 6.8 shows the execution time of the proposed multiclass CD scheme for the Hermiston and Synthetic 1 datasets, along with the speedups achieved by the CUDA GPU implementation as compared to an OpenMP CPU implementation with four threads. The total execution time of the scheme in GPU takes only 0.73 and 2.08 seconds for the considered datasets, while the multicore CPU execution times are up to 31 seconds. It can be seen that

	Binary CD	Filtering	ELM	Total
Hermiston in CPU				
Without binary filtering	–	–	0.3811	0.3811
With binary filtering	0.3629	0.0044	0.0802	0.4475
Speedup	–	–	**4.8**×	**0.9**×
Hermiston in GPU				
Without binary filtering	–	–	0.0534	0.0534
With binary filtering	0.0195	0.0003	0.0249	0.0447
Speedup	–	–	**2.1**×	**1.2**×
Synthetic 1 in CPU				
Without binary filtering	–	–	13.1564	13.1564
With binary filtering	5.7217	0.2355	3.0562	9.0134
Speedup	–	–	**4.3**×	**1.5**×
Synthetic 1 in GPU				
Without binary filtering	–	–	1.6806	1.6806
With binary filtering	0.0733	0.0015	0.5719	0.6466
Speedup	–	–	**2.9**×	**2.6**×

Table 6.7: Impact of the use of the binary CD (SAM+watershed+Otsu) in the execution time (OpenMP CPU and CUDA GPU, in seconds) and the speedup of the classification stage (ELM applied to enhanced change features image (difference+PCA+EMP).

	Binary CD	Fusion	PCA	EMP	Filtering	ELM	Total
Hermiston							
OpenMP CPU	0.3629	0.0327	0.1672	1.8127	0.0044	0.0802	2.4601
CUDA GPU	0.0195	0.0014	0.0512	0.6280	0.0003	0.0249	0.7253
Speedup	**18.6**×	**24.0**×	**3.3**×	**2.9**×	**13.3**×	**3.2**×	**3.4**×
Synthetic 1							
OpenMP CPU	5.7217	0.3335	1.7128	20.8336	0.2355	3.0562	31.8933
CUDA GPU	0.0733	0.0107	0.0947	1.3233	0.0015	0.5719	2.0753
Speedup	**78.1**×	**31.2**×	**18.1**×	**15.7**×	**161.0**×	**5.3**×	**15.4**×

Table 6.8: Multiclass CD execution times for the Hermiston and Synthetic 1 datasets. SAM+watershed+Otsu configuration in the binary branch, difference+PCA+EMP configuration in the multiclass branch of the CD, and ELM classifier.

every single stage of the processing achieves a positive speedup when projected to GPU, achieving a total speedup of 3.4× and 15.4×, respectively. For all the stages, the speedup is larger in the Synthetic 1 dataset as the data to be computed are enough to fully exploit the computational capabilities of the TITAN X GPU used in the experiments.

Method	Correct		MAs	FAs	Total Error	
C^2VA	117990	(95.46%)	2583	3027	5610	(4.54%)
Euclidean+Otsu	118002	(95.48%)	2586	3012	5598	(4.53%)
SAM+Otsu	**121387**	**(98.21%)**	**1062**	1151	**2213**	**(1.79%)**
Euclidean+watershed+Otsu	117055	(94.70%)	2763	3782	6545	(5.30%)
SAM+watershed+Otsu	121292	(98.13%)	1087	1221	2308	(1.87%)
SAM+Otsu+reg	121064	(97.95%)	1614	**922**	2536	(2.05%)
Euclidean+watershed+Otsu+reg	118946	(96.23%)	3429	1225	4654	(3.77%)
SAM+watershed+Otsu+reg	121039	(97.93%)	1523	1038	2561	(2.07%)

Table 6.9: Binary CD accuracy results for the Sardinia dataset (Correctly classified pixels, MAs, FAs, and total error), in terms of number of pixels and percentage. 'reg' stands for the application of spatial regularization to the CD map.

6.4 Multiclass CD results in a multispectral dataset

It is interesting to show the behavior of the schemes for datasets with different spectral and spatial dimensions, as well as different spatial resolutions, as the different real applications and sensors will need to process images in a broad range of dimensionality. The Sardinia dataset, introduced in Section 2.3.3, has been employed for the evaluation of multitemporal CD methods in several papers from the literature [192, 193, 224]. The results obtained through the application of the developed schemes introduced in this chapter to the multispectral dataset of Sardinia are detailed in this section. This dataset has lower dimensionality (only 6 spectral bands) than the other datasets used in this thesis and introduced in Section 2.3.3. The changes between the two images of this dataset are not related with the replacement of objects or areas but with the modification of the shapes of relevant objects in the images, as explained in Section 2.3.3. The edges between regions are also sharper than for the other datasets.

Table 6.9 shows the accuracies obtained in the binary CD of the Sardinia dataset with different configurations of the binary scheme presented in this thesis. The configuration that achieves the highest accuracy results is the one based on the SAM distance and Otsu's thresholding, without including any additional stage.

The CD maps achieved with the different configurations of the binary CD scheme introduced in Table 6.9 are shown in Figure 6.9. A binary CD map corresponding to the C^2VA approach is also included for comparison purposes (Figure 6.9(e)). C^2VA, as introduced in Section 1.8.1, is a method based on the computation of distances and angles between the input images, where the ED is used to decide whether a pixel has changed and the angle is used

Class	Number of pixels		Filtered pixels		MAs	% of total MAs	% MAs per class
0	113492	(91.82%)	1151	(11.290%)	–	–	–
1	214	(0.17%)	3	(0.037%)	211	19.87%	98.60%
2	2414	(1.95%)	1622	(15.91%)	792	74.58%	32.81%
3	7480	(6.05%)	7421	(72.78%)	59	5.56%	0.78%
TOTAL	78000	(100%)	9942	(100%)	1062	100%	–

Table 6.10: Binary filtering details for the Sardinia dataset (binary CD with SAM+Otsu configuration).

to cluster different types of changes. C^2VA is a technique used in several references in the literature [37, 193]. It can be seen in Figure 6.9 that the maps including spatial information by using segmentation algorithms (Figure 6.9(h-l)) remove disconnected misclassified pixels but they also loose the shape of the objects corresponding to the changes to be detected. For the case of this dataset, due to the sharp edges between regions presented, the drawback of loosing the correct shape of the changed areas is bigger than the advantages of removing the disconnected misclassified pixels.

For the configurations without spatial information (Figure 6.9(e-g)), the approach based on SAM distance achieves a cleaner CD map, which results in the better accuracy shown in Table 6.9. It also detects the change related with the simulated burned area (green in Figure 6.9(c)) Nevertheless, the small change related with the enlargement of the open quarry (gray in (Figure 6.9(c) and class 1 in Table 6.10) is not detected by the SAM distance. An analysis of different pixels of this area in both images from the Sardinia dataset shows that most of the pixels with this change in the available reference data maintain the same spectral signature in both images, with the differences between corresponding pixel-vectors only being related with changes of scale. An example of this is shown in Figure 6.10. As stated before in this thesis, the SAM distance is invariant to changes of scale, usually related to different illumination conditions. Therefore, it is appropriate that this area is not labeled as change as the spectral signatures of the pixels of the area suggest the presence of the same materials in both images of the dataset.

The detailed results of the binary filtering process for the Sardinia dataset are displayed in Figure 6.10. As has previously been explained, the changes of class 1 (gray in (Figure 6.9(c)), located in the bottom right part of the image, are mostly MAs. The filtering of the changes of class 3, the class including more pixels and whose changes are not simulated or related to changes in the illumination, correctly passes more than the 99% of the change pixels of the class.

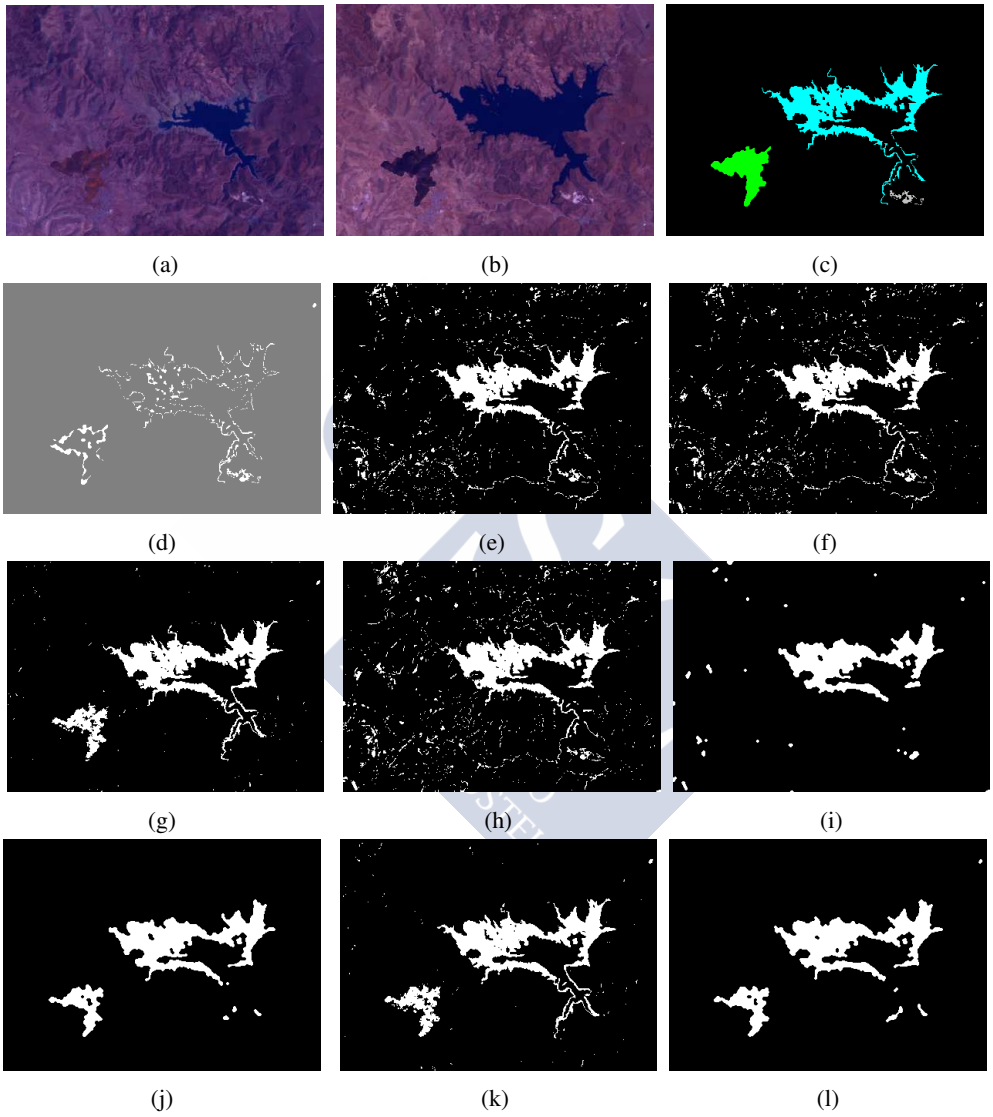


Figure 6.9: Color composition of the input images (a, b), reference data of changes {white = no change, gray = change} (c), hit map for SAM+Otsu {white = miss, gray = hit} (d), C^2VA CD map (e), ED+Otsu CD map (f), SAM+Otsu CD map (g), ED+watershed+Otsu CD map (h), ED+watershed+Otsu+reg CD map (i), SAM+Otsu+reg CD map (j), SAM+watershed+Otsu CD map (k), and SAM+watershed+Otsu+reg CD map (l) for the Sardinia dataset.

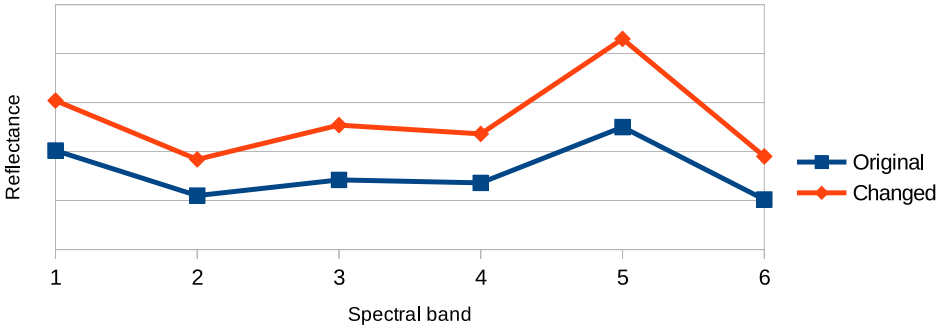


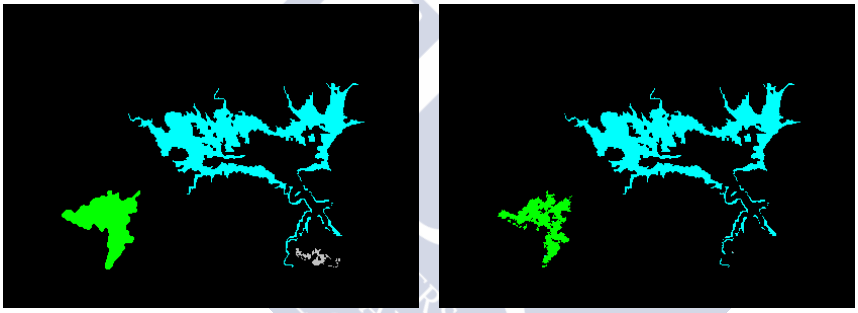
Figure 6.10: Sardinia dataset: Example of spectral signatures of the same pixel in both images of the dataset. The pixel is labelled as type 1 change.

Finally, Table 6.11 shows the multiclass CD accuracies for the Sardinia dataset. As this is a dataset with a much lower dimensionality, the configurations explored are different from those of previous datasets. The difference and stack of the multitemporal information are explored as fusion techniques, and the fused information of dimensionality 6 and 12, respectively, directly passed to the classifiers without building an EMP. The PCA and NWFE FE methods are evaluated retaining 3 components in all the cases. Due to the type of changes present in this dataset, with sharp and irregular edges, the spatial processing through EMPs is not efficient in this case, so it is removed from the method.

As can be seen, all the different configurations considered achieve very high accuracy in terms of OA and k, above 99.9% and 99.5%, respectively. Nevertheless, for the case of the AA accuracy, it must be taken into account that one of the three considered classes of change only passes 3 pixels after the filtering and that is not sufficient for the suitable training and testing of a supervised classifier. So, the AA are around a value of 66.6% in most of the configurations because the pixels of this class are not correctly classified. Figure 6.11 shows the best CD classification map achieved for the Sardinia dataset. It can be seen that all the pixels selected as change are classified to the corresponding change class regarding the available reference data.

Fusion	FE	SVM					ELM			
		Parameters		OA	AA	k	Parameters	OA	AA	k
		C	γ				N			
		32	1	99.94	66.66	99.80	20	99.95	74.95	99.84
Difference	PCA	32	1	99.92	66.66	99.70	10	99.92	69.91	99.72
	NWFE	32	1	99.93	66.66	99.80	10	99.92	69.90	99.72
		32	1	99.92	**100.00**	99.70	50	99.97	91.65	**99.90**
Stack	PCA	32	0.125	99.95	66.66	99.80	10	99.88	76.51	99.57
	NWFE	32	1	**99.97**	**100.00**	**99.90**	10	99.94	76.61	99.79

Table 6.11: Multiclass CD accuracies for the Sardinia dataset.



(a) (b)

Figure 6.11: Reference data of changes (a) and best ELM multiclass CD map (b) for the Sardinia dataset.

6.5 Multiclass CD results in the presence of noise

The capture of the hyperspectral images often produces noise that modifies the original spectral signatures. This is the reason why it is important for a remote sensing scheme to be robust in the presence of noise. In this section, the behavior of the multiclass CD scheme proposed in this thesis facing different types and levels of noise is evaluated.

The experiments were carried out on the Sardinia dataset. Two different types of noise have been added to the original images: AWGN and speckle noise, which were introduced in Section 2.3.2. The effects of different levels of AWGN and speckle noise in the binary CD of the Sardinia dataset are shown in Tables 6.12 and 6.13. Table 6.12 shows the results of the binary branch of the scheme in terms of correctly classified pixels, MA, FA, and total error. Table 6.13 details the accuracies achieved for the change and no change classes. Three

different variance levels (σ) have been considered for each kind of noise, with the input images being scaled within the range $[0-1]$. The PSNR obtained after the application of each considered noise is also shown in the tables as a measure of the level of degradation of the input images.

Table 6.12 shows the binary results for the SAM+Otsu binary approach considered in this thesis and also for the C^2VA approach for the Sardinia dataset. It can be seen that for both techniques and both types of noise, the greater the degradation of the signal the worse the CD accuracies obtained. In terms of total correctly classified pixels, the C^2VA outperforms the SAM+Otsu technique for the three levels of AWGN noise and the larger level of speckle noise. Nevertheless, a careful analysis of the MAs and FAs shows that the SAM+Otsu technique achieves better MAs results in the six considered scenarios. This is important given that the MAs values are directly related with the amount of changes that the technique is able to detect.

As can be observed in Table 6.13, the proposed SAM+Otsu technique maintains better accuracy in the change pixels in the six scenarios. The C^2VA achieves a better overall result in some cases in Table 6.12 as it tends towards a smaller number of FAs and the percentage of no change pixels in the image is much larger than the one of change pixels.

In fact, in five of the six noise cases considered, the proposed technique maintains an accuracy for the pixels of the change class similar than the one achieved in the dataset without noise, as can be seen in Table 6.13. The accuracy for this class only decreases drastically in the case of AWGN noise with σ=0.04 where the PSNR is only 15 decibels and the amount of correctly detected pixels of both methods is around 56%.

Regarding the multiclass accuracy (calculated over the pixels selected as change by the binary stage), as can be seen in Table 6.14, the proposed multiclass CD method achieves better results in the presence of noise than the C^2VA in five of the six considered noise scenarios. The proposed method performs better, although the FE techniques are not applied in these datasets due to its small dimensionality, independently of the fusion method (difference or stack of the input images) and the classification algorithm considered (SVM or ELM). The C^2VA achieves a slightly better result in the case of AWGN with σ=0.04; however, it must be taken into account that for this scenario only the 57% of the change pixels were correctly filtered in the binary stage, while with the CD method proposed in this thesis this value increases up to 71%, as shown in Table 6.13.

From the analysis of the behavior of the proposed multiclass CD scheme in the presence of noise, it can be extracted that it is sensitive to noisy conditions, particularly in the binary

Noise			C^{2}VA						SAM+Otsu					
Type	σ	**PSNR**	**Correct**		**MAs**	**FAs**	**Total error**		**Correct**		**MAs**	**FAs**	**Total error**	
–	–	–	117990	(95.46%)	2583	3027	5610	(4.54%)	**121387**	**(98.21%)**	**1062**	**1151**	**2213**	**(1.79%)**
	0.005	23db	**102335**	**(82.80%)**	1798	**19467**	**21265**	**(17.20%)**	91446	(73.99%)	**988**	31166	32154	(26.01%)
AWGN	0.01	20db	**84117**	**(68.06%)**	1977	**37506**	**39483**	**(31.94%)**	73841	(59.74%)	**1063**	48696	49759	(40.26%)
	0.04	15db	**69964**	**(56.61%)**	4366	**49270**	**53636**	**(43.39%)**	69159	(55.95%)	**2884**	51557	54441	(44.05%)
	005	35db	120085	(97.16%)	2612	**903**	3515	(2.84%)	**121282**	**(98.12%)**	**1349**	969	**2318**	**(1.88%)**
Speckle	0.01	32db	119788	(96.92%)	2620	**1192**	3812	(3.08%)	**121001**	**(97.90%)**	**1261**	1338	**2599**	**(2.10%)**
	0.04	26db	**109161**	**(88.32%)**	2172	**12267**	**14439**	**(11.68%)**	107901	(87.30%)	**1099**	14600	15699	(12.70%)

Table 6.12: Binary CD accuracies for the Sardinia dataset in the presence of noise.

Noise			C^2VA		SAM+Otsu	
Type	σ	**PSNR**	**Change**	**No change**	**Change**	**No change**
–	–	–	74.45	97.33	**89.49**	**98.99**
AWGN	0.005	23db	82.21	**82.84**	**90.22**	72.54
	0.01	20db	80.44	**66.95**	**89.48**	57.09
	0.04	15db	56.81	**56.58**	**71.47**	54.57
Speckle	0.005	35db	74.16	**99.20**	**86.65**	99.15
	0.01	32db	74.08	**98.94**	**87.52**	98.82
	0.04	26db	78.51	**89.19**	**89.13**	87.14

Table 6.13: Detailed per class binary CD accuracies, as percentages, for the Sardinia dataset in the presence of noise.

Noise			C^2VA	Difference		Stack	
Type	σ	**PSNR**		**SVM**	**ELM**	**SVM**	**ELM**
–	–	–	99.30	99.94	99.95	99.92	**99.97**
AWGN	0.005	23db	93.56	97.13	97.19	97.91	**98.05**
	0.01	20db	91.17	94.65	94.17	**95.83**	95.28
	0.04	15db	**86.45**	85.53	86.02	86.12	86.41
Speckle	0.005	35db	99.20	99.92	99.93	**99.98**	99.96
	0.01	32db	98.89	99.91	99.92	99.93	**99.94**
	0.04	26db	96.81	99.74	99.77	**99.90**	99.88

Table 6.14: Multiclass CD accuracies in terms of OA for the Sardinia dataset in the presence of noise.

branch of the CD. Therefore, the introduction of denoising techniques as an initial stage of the CD scheme, or in its binary branch, should be considered to achieve a more robust multiclass CD scheme for multitemporal images.

6.6 Discussion

The complete multiclass CD scheme for multitemporal hyperspectral dataset targeted as the final goal of this thesis has been presented in this chapter.

The proposed scheme combines binary and multiclass CD techniques in order to obtain a detailed change classification map including *from-to* transition information. The combination of the binary CD with the multiclass one, also allows us to obtain a more efficient scheme, reducing the computational cost of the multiclass branch of the scheme.

The scheme relies on the use of SAE to perform FE of the multitemporal information obtained in the dataset, which is previously merged as the pixel-by-pixel difference or stack of the input images. The spatial information is then taken into account by the computation of an EMP of the extracted features. In this way, a change feature enhanced image is constructed.

The EMP is filtered by the binary CD map obtained by the binary CD branch of the method. So only the change pixels are computed in the final classification stage, where an efficient ELM classifier is used to obtain the final change classification map.

The main advantage of this scheme over others presented in the literature is that it provides detailed transition information for each pixel of the dataset. The GPU projection of all the algorithms involved in the computation of the scheme, along with the scheme design itself, allows us to compute the scheme efficiently in commodity hardware.

The proposed scheme has been evaluated in two hyperspectral datasets, both in terms of accuracy and time execution performance. In accuracy terms, the scheme correctly detects up to the 98.75% of the changes present in the Hermiston dataset, with an accuracy up to 97.89% in the subsequent change classification. In both cases, all the changes were correctly detected and classified at the object level, and the errors only correspond to disconnected pixels. The scheme performs similar in the Synthetic 1 dataset, created to obtain a more difficult challenge, only producing incorrect change detections when the transitions are over very similar materials and achieving a 100% of correct change classification with different configurations. Regarding the performance in terms of execution time, the proposed binary and multiclass CD combination allows a speedup for the scheme of up to 2.6× in the considered datasets. The proposed filtering provides better speedups the larger the size of the dataset to be processed, assuming that the amount of change pixels is usually smaller than the amount of no change ones. The complete scheme can be computed in under one second in real hyperspectral images (0.72 seconds for the Hermiston dataset) and the GPU implementation achieves a total speedup up to 15.4× in the execution of the scheme as compared to a 4-thread OpenMP multicore version of the same scheme, in this case, for the Synthetic 1 dataset, which is the one which is sufficiently large to fully exploit the computational capabilities of the GPU.

The CD scheme was also evaluated on a multispectral dataset from the literature, showing that it is suitable for use with in lower dimensionality images, although the particular properties of the changes present on the considered dataset make it necessary to select a different configuration of the scheme from the one proposed to hyperspectral images. In particular, FE was not applied, because of the small original dimensionality. The spatial processing was also

removed due to the fact that the changes in the considered dataset are related with modifications of the shapes of relevant objects with sharp edges instead of the replacement of existent objects or areas.

The behavior of the proposed CD scheme has been tested in the presence of different types (AWGN and Speckle) and magnitudes of noise in the input datasets. The scheme detects more changes than an alternative C^2VA from the literature under these circumstances. Nevertheless, the number of FAs increases when the magnitude of the noise is elevated. Regarding the robustness in the change classification in noisy environments, the scheme maintains accuracies very similar to that achieved without noise in all the considered scenarios, being more robust than the C^2VA alternative from the literature.

Conclusions

This thesis addresses the efficient multiclass Change Detection (CD) of multitemporal hyperspectral remote sensing datasets. The main focus is set on the proposal of new CD schemes which allow us to improve the accuracies of the schemes in the literature and are also suitable for their efficient parallel implementation in commodity hardware, such as the Graphics Processing Units (GPUs). The following paragraphs detail the main contributions of this thesis and also explain how the target objectives have been accomplished.

1. The main characteristics of the hyperspectral imagery have been reviewed. The fundamentals of the different processing techniques for hyperspectral images used in this thesis have also been studied, including Feature Extraction (FE), segmentation, Mathematical Morphology (MM), supervised classification, registration, and, especially, CD for multitemporal datasets.

2. Two different *parallel programming models* have been studied, in order to be used for the achievement of efficient parallel implementations of the schemes proposed in this thesis: the OpenMP Application Program Interface (API), for multi-threaded applications in shared memory architectures in Central Processing Unit (CPU), and the Compute Unified Device Architecture (CUDA), for general purpose parallel computation on NVIDIA GPUs. Several techniques to be applied in the GPU implementations with the aim of fully exploiting the architecture capabilities have also been introduced and explained in detail. The introduced techniques are used throughout the thesis.

3. *Efficient schemes for the classification of hyperspectral datasets through Extreme Learning Machine (ELM) based classifiers* have been proposed. The first GPU implementation of the ELM classifier for hyperspectral images in the literature has been developed

in this thesis. Two spectral-spatial schemes combining the ELM classifier with the spatial information obtained by using segmentation techniques have been also introduced: one is based on the use of the watershed transform, which operates over the Robust Color Morphological Gradient (RCMG) of the original dataset, and the other is based on the use of the Really Quick Shift (RQS) segmentation. In both cases the combination of the spectral and spatial information is performed by a Majority Vote (MV) process of the pixel-wise classification inside each segmentation region. The proposed schemes provide advantages both in the quality of the classification maps achieved, as compared with alternatives from the literature, such as the Support Vector Machine (SVM), and in the efficiency in terms of computation time. Overall Accuracy (OA) values up to 96.66% have been achieved for the Pavia University dataset, with the ELM-based classifications being up to 8.8× faster than the ones performed with analogous schemes based on SVM for the same image.

4. *The first GPU projection of the Evolutionary Cellular Automata Segmentation (ECAS-II) segmenter* has been developed. The main characteristic that defines this segmentation is the fact that it maintains all the original information of the dataset instead of performing dimensionality reduction. The proposed implementation allows us to considerably reduce the execution time of this segmentation algorithm, making it eligible for inclusion in efficient spectral-spatial schemes. In this way, the presented GPU projection improves the execution time of ECAS-II by up to 21×, as compared with an efficient OpenMP implementation of the same algorithm for the Salinas dataset. ECAS-II has also been combined with the ELM classifier in a new spectral-spatial classification scheme. Experiments show that the proposed spectral-spatial classification scheme based on ECAS-II segmentation outperforms the classification accuracies of other schemes based on segmentation techniques from the literature, achieving OA values above 98% for the datasets considered in the experiments.

5. New efficient CD schemes for multitemporal hyperspectral datasets have been proposed.

    - A *scheme focused on binary CD at the object level*. This scheme uses a region averaging process based on watershed segmentation to efficiently include, in terms of computational load, the spatial information at object level. The scheme is based on Change Vector Analysis (CVA), improving the computation of the differences

between the images by using the Spectral Angle Mapper (SAM) distance. The Otsu thresholding method has been proposed in the scheme to obtain the binary CD map with time efficiency. A final spatial regularization processing of the CD map has also been introduced to improve the accuracy of the scheme. The experiments show that the proposed scheme achieves very good CD accuracy results (above 97% of OA) when applied to multitemporal remote sensing hyperspectral datasets. The proposed scheme is fully projected onto GPU, which allows its incorporation in real-time processing scenarios, with the required computation time for the two datasets considered in the experiments (Santa Barbara and Bay Area) being 0.077 and 0.034 seconds, respectively. An implementation of the scheme for multi-GPU architectures has also been presented in this thesis, achieving speedups of $2\times$ in the segmentation stage of the scheme, which is performed in a separate GPU for each image.

- A *multiclass CD scheme combining binary and multiclass CD*. The objective is not only the detection of changes at object level, but also the classification of the type of changes. The binary branch of the scheme is computed with the previously presented binary CD scheme. A direct multidate classification is selected to deal with the multiclass processing. A final classification stage is included to provide detailed *from-to* transitions in the CD map. The classification stage is performed over an image with enhanced change information through three consecutive stages.

  a) The change enhancing processing proposed starts with the fusion of the multitemporal information through the difference or stack of the input images, followed by an FE by means of Stacked Autoencoders (SAEs) or Principal Component Analysis (PCA). This combination provides a compressed (and therefore, computationally efficient) representation of the dataset, where the changes are highlighted to facilitate their posterior classification. An Extended Morphological Profile (EMP) of this image is then computed, to include the spatial information of the datasets in this processing.

  b) Once both the binary and multiclass branches of the scheme have been computed, a filtering process is used to combine the binary and multiclass branches of the scheme. In this way, only the pixels selected as changes in the binary CD are retained from the change feature processing to be used as the input to

the final classification stage. This binary filtering allows us to greatly reduce the computational load of the classification stage of the scheme.

c) Finally, the efficient ELM proposed in this thesis is used to obtain the final multiclass CD map.

The scheme is fully projected to GPU to reduce the computation time required to perform the CD. The scheme achieves up to 98.8% of correctly change detection, with equally high change classification accuracies. The proposed scheme allows the total CD computation to be performed in 0.7 seconds for the Hermiston dataset. Finally, the robustness of the proposed scheme in noisy conditions has also been evaluated with positive results.

Several hyperspectral and multispectral datasets acquired from remote sensing platforms, such as the AVIRIS, ROSIS-03, Hyperion, and Thematic Mapper sensors, have been used for the evaluation of the schemes and techniques proposed in this thesis. The accuracy of the proposed schemes is compared with similar schemes under the same conditions in all the cases. The execution time performance has been evaluated by comparing the proposed GPU implementations with efficient OpenMP implementations developed in this thesis for a fair comparison.

Different commodity hardware architectures have also been used in the experiments. An Intel Core i5-3470 CPU was used for the OpenMP implementations while different GPUs corresponding to different generations of CUDA capable devices were used for the GPU projections. In particular a GTX TITAN belonging to the Kepler family and a GTX TITAN X from the Pascal generation were used for the single GPU implementations. The multi-GPU experiments were carried out in a NVIDIA Tesla K80 device, also belonging to the Kepler family.

Considering the results introduced in this thesis, the main targeted objectives have been successfully accomplished. It has been proven that it is possible to design a multiclass CD method improving the efficiency of those in the literature, and with a reduced computational cost, projected to commodity GPU hardware.

Taking the research carried out in this thesis as a starting point, there are several possible lines for further work. For instance, the CD schemes proposed in this thesis may be improved, adding stages devoted to the denoising of the input images, in order to achieve schemes more robust to the noise produced during the acquisition of the data. Additionally, the projection of the processing schemes to miniaturized embedded systems, such as the Tegra X1 mobile

processor based on ARM, may result in the ability to perform on-board remote sensing processing in real-time in platforms, such as drones.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] Freek D. Van der Meer, Harald van der Werff, Frank J.A. van Ruitenbeek, Chris A. Hecker, Wim H. Bakker, Marleen F. Noomen, Mark van der Meijde, E. John M. Carranza, J. Smeth, and Tsehaie Woldai. Multi-and hyperspectral geologic remote sensing: A review. *International Journal of Applied Earth Observation and Geoinformation*, 14(1):112–128, 2012.

[2] J. Bioucas-Dias, Antonio Plaza, Gustavo Camps-Valls, Paul Scheunders, N. Nasrabadi, and Jocelyn Chanussot. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geosci. Remote Sens. Mag.*, 1(2):6–36, 2013.

[3] Mustafa Teke, Hüsne Seda Deveci, Onur Haliloğlu, Sevgi Zübeyde Gürbüz, and Ufuk Sakarya. A short survey of hyperspectral remote sensing applications in agriculture. In *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*, pages 171–176. IEEE, 2013.

[4] J. Solomon and B. Rock. Imaging spectrometry for earth remote sensing. *Science*, 228(4704):1147–1152, 1985.

[5] Mathieu Fauvel, Yuliya Tarabalka, Jón Atli Benediktsson, Jocelyn Chanussot, and James C. Tilton. Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675, 2013.

[6] Yuliya Tarabalka, Jón Atli Benediktsson, Jocelyn Chanussot, and James C. Tilton. Multiple spectral–spatial classification approach for hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4122–4132, 2010.

[7] Thomas Eugene Avery and G.L. Berlin. Fundamentals of remote sensing and airphoto interpretation. 5th. *Prentice Hall, London*, 1992.

[8] John A. Richards and Xiuping Jia. *Remote sensing digital image analysis*, volume 3. Springer, 1999.

[9] 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.

[10] Robert O. Green, Michael L. Eastwood, Charles M. Sarture, Thomas G. Chrien, Mikael Aronsson, Bruce J. Chippendale, Jessica A. Faust, Betina E. Pavri, Christopher J. Chovit, Manuel Solis, et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). *Remote sensing of environment*, 65(3):227–248, 1998.

[11] A.A. Müller, Andrea Hausold, Peter Strobl, and M. Ehlers. Hysens-dais/rosis imaging spectrometers at dlr. In *Proc. SPIE*, volume 4545, page 226, 2001.

[12] Francesca Bovolo and Lorenzo Bruzzone. The time variable in data fusion: A change detection perspective. *Geoscience and Remote Sensing Magazine, IEEE*, 3(3):8–26, 2015.

[13] Ashbindu Singh. Review article digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6):989–1003, 1989.

[14] Sicong Liu, Lorenzo Bruzzone, Francesca Bovolo, and Peijun Du. Hierarchical unsupervised change detection in multitemporal hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 53(1):244–260, 2015.

[15] Mauro Dalla Mura, Saurabh Prasad, Fabio Pacifici, Paulo Gamba, Jocelyn Chanussot, and Jón Atli Benediktsson. Challenges and opportunities of multimodality and data fusion in remote sensing. *Proceedings of the IEEE*, 103(9):1585–1601, 2015.

[16] Edward J. Kaiser, David R. Godschalk, and F. Stuart Chapin. *Urban land use planning*, volume 4. University of Illinois Press Urbana, IL, 1995.

[17] X. Liu and R.G. Lathrop Jr. Urban change detection based on an artificial neural network. *International Journal of Remote Sensing*, 23(12):2513–2518, 2002.

[18] David M Tralli, Ronald G Blom, Victor Zlotnicki, Andrea Donnellan, and Diane L. Evans. Satellite remote sensing of earthquake, volcano, flood, landslide and coastal inundation hazards. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(4):185–198, 2005.

[19] P.A. Brivio, R. Colombo, M. Maggi, and R. Tomasoni. Integration of remote sensing data and GIS for accurate mapping of flooded areas. *International Journal of Remote Sensing*, 23(3):429–441, 2002.

[20] Glenn B. Stracher and Tammy P. Taylor. Coal fires burning out of control around the world: thermodynamic recipe for environmental catastrophe. *International Journal of Coal Geology*, 59(1):7–17, 2004.

[21] Paul C. Doraiswamy, Sophie Moulin, Paul W Cook, and Alan Stern. Crop yield assessment from remote sensing. *Photogrammetric engineering & remote sensing*, 69(6):665–674, 2003.

[22] S. Moulin, A. Bondeau, and R. Delecolle. Combining agricultural crop models and satellite observations: from field to regional scales. *International Journal of Remote Sensing*, 19(6):1021–1036, 1998.

[23] Marie Launay and Martine Guerif. Assimilating remote sensing data into a crop model to improve predictive performance for spatial applications. *Agriculture, ecosystems & environment*, 111(1):321–339, 2005.

[24] D.A. Stow. Reducing the effects of misregistration on pixel-level change detection. *International Journal of Remote Sensing*, 20(12):2477–2483, 1999.

[25] Volker Walter. Object-based classification of remote sensing data for change detection. *ISPRS Journal of photogrammetry and remote sensing*, 58(3):225–238, 2004.

[26] Allan Aasbjerg Nielsen. The regularized iteratively reweighted MAD method for change detection in multi-and hyperspectral data. *IEEE Transactions on Image processing*, 16(2):463–478, 2007.

[27] Baudouin Desclée, Patrick Bogaert, and Pierre Defourny. Forest change detection by statistical object-based method. *Remote Sensing of Environment*, 102(1):1–11, 2006.

[28] Massimo Zanetti and Lorenzo Bruzzone. A generalized statistical model for binary change detection in multispectral images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 3378–3381. IEEE, 2016.

[29] Masroor Hussain, Dongmei Chen, Angela Cheng, Hui Wei, and David Stanley. Change detection from remotely sensed images: From pixel-based to object-based approaches. *ISPRS Journal of Photogrammetry and Remote Sensing*, 80:91–106, 2013.

[30] Hassiba Nemmour and Youcef Chibani. Support vector machines for automatic multiclass change detection in algerian capital using Landsat TM imagery. *Journal of the Indian Society of Remote Sensing*, 38(4):585–591, 2010.

[31] Hassiba Nemmour and Youcef Chibani. Multiple support vector machines for land cover change detection: An application for mapping urban extensions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(2):125–133, 2006.

[32] A. Bannari, D. Morin, F. Bonn, and A.R. Huete. A review of vegetation indices. *Remote sensing reviews*, 13(1-2):95–120, 1995.

[33] William A. Malila. Change vector analysis: an approach for detecting forest changes with Landsat. In *LARS Symposia*, page 385, 1980.

[34] Sankar K. Pal. Fuzzy image processing and recognition: Uncertainty handling and applications. *International Journal of Image and Graphics*, 1(02):169–195, 2001.

[35] Ashish Ghosh, Badri Narayan Subudhi, and Lorenzo Bruzzone. Integration of gibbs markov random field and hopfield-type neural networks for unsupervised change detection in remotely sensed multitemporal images. *Image Processing, IEEE Transactions on*, 22(8):3087–3096, 2013.

[36] Ming Hao, Wenzhong Shi, Hua Zhang, and Chang Li. Unsupervised change detection with expectation-maximization-based level set. *Geoscience and Remote Sensing Letters, IEEE*, 11(1):210–214, 2014.

[37] Sicong Liu, Lorenzo Bruzzone, Francesca Bovolo, Massimo Zanetti, and Peijun Du. Sequential spectral change vector analysis for iteratively discovering and detecting multiple changes in hyperspectral images. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(8):4363–4378, 2015.

[38] Feng Ling, Wenbo Li, Yun Du, and Xiaodong Li. Land cover change mapping at the subpixel scale with different spatial-resolution remotely sensed imagery. *Geoscience and Remote Sensing Letters, IEEE*, 8(1):182–186, 2011.

[39] Pieter Kempeneers, Fernando Sedano, Peter Strobl, Daniel O. McInerney, and Jesús San-Miguel-Ayanz. Increasing robustness of postclassification change detection using time series of land cover maps. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(9):3327–3339, 2012.

[40] Xia Li and AGO Yeh. Principal component analysis of stacked multi-temporal images for the monitoring of rapid urban expansion in the Pearl River Delta. *International Journal of Remote Sensing*, 19(8):1501–1518, 1998.

[41] X.L. Dai and Slamak Khorram. Remotely sensed change detection based on artificial neural networks. *Photogrammetric engineering and remote sensing*, 65:1187–1194, 1999.

[42] Michele Volpi, Devis Tuia, Francesca Bovolo, Mikhail Kanevski, and Lorenzo Bruzzone. Supervised change detection in VHR images using contextual information and support vector machines. *International Journal of Applied Earth Observation and Geoinformation*, 20:77–85, 2013.

[43] Farid Melgani and Lorenzo Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, 2004.

[44] Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3):247–259, 2011.

[45] Mahesh Pal and P.M. Mather. Support vector machines for classification in remote sensing. *International Journal of Remote Sensing*, 26(5):1007–1011, 2005.

[46] Yakoub Bazi and Farid Melgani. Toward an optimal SVM classification system for hyperspectral remote sensing images. *IEEE Transactions on geoscience and remote sensing*, 44(11):3374–3385, 2006.

[47] Guang-Bin Huang, Dian Hui Wang, and Yuan Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.

[48] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(2):513–529, 2012.

[49] Nan Liu and Han Wang. Ensemble based extreme learning machine. *IEEE Signal Processing Letters*, 17(8):754–757, 2010.

[50] Jiuwen Cao, Zhiping Lin, Guang-Bin Huang, and Nan Liu. Voting based extreme learning machine. *Information Sciences*, 185(1):66–77, 2012.

[51] Mahesh Pal. Extreme-learning-machine-based land cover classification. *International Journal of Remote Sensing*, 30(14):3835–3841, 2009.

[52] Mahesh Pal, Aaron E. Maxwell, and Timothy A. Warner. Kernel-based extreme learning machine for remote-sensing image classification. *Remote Sensing Letters*, 4(9):853–862, 2013.

[53] Dora B. Heras, Francisco Argüello, and Pablo Quesada-Barriuso. Exploring ELM-based spatial–spectral classification of hyperspectral images. *International Journal of Remote Sensing*, 35(2):401–423, 2014.

[54] Ramón Moreno, Francesco Corona, Amaury Lendasse, Manuel Graña, and Lênio S. Galvão. Extreme learning machines for soybean classification in remote sensing hyperspectral images. *Neurocomputing*, 128:207–216, 2014.

[55] Borja Ayerdi, Ion Marqués, and Manuel Graña. Spatially regularized semisupervised ensembles of extreme learning machines for hyperspectral image segmentation. *Neurocomputing*, 2014.

[56] Mark van Heeswijk, Yoan Miche, Erkki Oja, and Amaury Lendasse. GPU-accelerated and parallelized ELM ensembles for large-scale regression. *Neurocomputing*, 74(16):2430–2437, 2011.

[57] Qiangqiang Yuan, Liangpei Zhang, and Huanfeng Shen. Hyperspectral image denoising employing a spectral–spatial adaptive total variation model. *IEEE Transactions on Geoscience and Remote Sensing*, 50(10):3660–3677, 2012.

[58] Yong-Qiang Zhao and Jingxiang Yang. Hyperspectral image denoising via sparse representation and low-rank constraint. *IEEE Transactions on Geoscience and Remote Sensing*, 53(1):296–308, 2015.

[59] Mathieu Fauvel, Jocelyn Chanussot, and Jón Atli Benediktsson. A spatial–spectral kernel-based approach for the classification of remote-sensing images. *Pattern Recognition*, 45(1):381–392, 2012.

[60] Qian Yu, Peng Gong, Nick Clinton, Greg Biging, Maggi Kelly, and Dave Schirokauer. Object-based detailed vegetation classification with airborne high spatial resolution remote sensing imagery. *Photogrammetric Engineering & Remote Sensing*, 72(7):799–811, 2006.

[61] Mathieu Fauvel, Jón Atli Benediktsson, Jocelyn Chanussot, and Johannes R. Sveinsson. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 46(11):3804–3814, 2008.

[62] Antonio Plaza, Jón 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.

[63] Yuliya Tarabalka, Jocelyn Chanussot, and Jón Atli Benediktsson. Segmentation and classification of hyperspectral images using watershed transformation. *Pattern Recognition*, 43(7):2367–2379, 2010.

[64] Brian Fulkerson and Stefano Soatto. Really Quick Shift: Image segmentation on a GPU. In *ECCV Workshops (1)*, pages 350–358, 2010.

[65] B. Priego, F. Bellas, and R. J. Duro. Evolving cellular automata to segment hyperspectral images using low dimensional images for training. *Bioinspired Computation in Artificial Systems*, 9108:117–126, 2015.

[66] Jón Atli Benediktsson, Jón Aevar Palmason, and Johannes R. Sveinsson. Classification of hyperspectral data from urban areas based on extended morphological profiles. *Geoscience and Remote Sensing, IEEE Transactions on*, 43(3):480–491, 2005.

[67] Jon Aevar Palmason, Jón Atli Benediktsson, Johannes R. Sveinsson, and Jocelyn Chanussot. Classification of hyperspectral data from urban areas using morpholgical preprocessing and independent component analysis. In *International Geoscience And Remote Sensing Symposium*, volume 1, page 176, 2005.

[68] Xin Huang, Xuehua Guan, Jón Atli Benediktsson, Liangpei Zhang, Jun Li, Antonio Plaza, and Mauro Dalla Mura. Multiple morphological profiles from multicomponent-base images for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12):4653–4669, 2014.

[69] Yanfeng Gu, Tianzhu Liu, Xiuping Jia, Jón Atli Benediktsson, and Jocelyn Chanussot. Nonlinear multiple kernel learning with multiple-structure-element extended morphological profiles for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3235–3247, 2016.

[70] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[71] Pierre Soille and Martino Pesaresi. Advances in mathematical morphology applied to geoscience and remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 40(9):2042–2055, 2002.

[72] Jón Atli Benediktsson, Martino Pesaresi, and Kolbeinn Amason. Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):1940–1949, 2003.

[73] 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.

[74] Robert A. Schowengerdt. *Remote sensing: models and methods for image processing*. Academic press, 2006.

[75] Mathieu Fauvel, Jocelyn Chanussot, and Jon Atli Benediktsson. Kernel principal component analysis for the classification of hyperspectral remote sensing data over urban areas. *EURASIP Journal on Advances in Signal Processing*, 2009(1):783194, 2009.

[76] Giona Matasci, Michele Volpi, Mikhail Kanevski, Lorenzo Bruzzone, and Devis Tuia. Semisupervised transfer component analysis for domain adaptation in remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(7):3550–3564, 2015.

[77] Chi Hau Chen and Xiaohui Zhang. Independent component analysis for remote sensing study. In *Proc. SPIE*, volume 3871, pages 150–158, 1999.

[78] Silvia Marchesi and Lorenzo Bruzzone. ICA and kernel ICA for change detection in multispectral remote sensing images. In *Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009*, volume 2, pages II–980. IEEE, 2009.

[79] Bor-Chen Kuo and David A. Landgrebe. A robust classification procedure based on mixture classifiers and nonparametric weighted feature extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 40(11):2486–2494, 2002.

[80] Pablo Quesada-Barriuso, Francisco Argüello, and Dora B. Heras. Computing efficiently spectral-spatial classification of hyperspectral images on commodity GPUs. In *Recent Advances in Knowledge-Based Paradigms and Applications*, pages 19–42. Springer, 2014.

[81] Liangpei Zhang, Lefei Zhang, and Bo Du. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):22–40, 2016.

[82] John E Ball, Derek T Anderson, and Chee Seng Chan. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):042609, 2017.

[83] Karl Pearson. Principal components analysis. *The London, Edinburgh and Dublin Philosophical Magazine and Journal*, 6(2):566, 1901.

[84] Lingyan Ran, Yanning Zhang, Wei Wei, and Tao Yang. Bands sensitive convolutional network for hyperspectral image classification. In *Proceedings of the International*

Conference on Internet Multimedia Computing and Service, pages 268–272. ACM, 2016.

[85] Jaime Zabalza, Jinchang Ren, Jiangbin Zheng, Huimin Zhao, Chunmei Qing, Zhijing Yang, Peijun Du, and Stephen Marshall. Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, 185:1–10, 2016.

[86] 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.

[87] Dou Quan, Shuang Wang, Mengdan Ning, Tao Xiong, and Licheng Jiao. Using deep neural networks for synthetic aperture radar image registration. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 2799–2802. IEEE, 2016.

[88] Fahim Irfan Alam, Jun Zhou, Alan Wee-Chung Liew, and Xiuping Jia. CRF learning with CNN features for hyperspectral image segmentation. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 6890–6893. IEEE, 2016.

[89] Essa Basaeed, Harish Bhaskar, Paul Hill, Mohammed Al-Mualla, and David Bull. A supervised hierarchical segmentation of remote-sensing images using a committee of multi-scale convolutional neural networks. *International Journal of Remote Sensing*, 37(7):1671–1691, 2016.

[90] Jun Wang, Qiming Qin, Zhoujing Li, Xin Ye, Jianhua Wang, Xiucheng Yang, and Xuebin Qin. Deep hierarchical representation and segmentation of high resolution remote sensing images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, pages 4320–4323. IEEE, 2015.

[91] Saptarshi Pal, Srija Chowdhury, and Soumya K. Ghosh. Dcap: A deep convolution architecture for prediction of urban growth. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 1812–1815. IEEE, 2016.

[92] Huaizhong Zhang, Pablo Casaseca-de-la Higuera, Chunbo Luo, Qi Wang, Matthew Kitchin, Andrew Parmley, and Jesus Monge-Alvarez. Systematic infrared image quality improvement using deep learning based techniques. In *Remote Sensing Technologies and Applications in Urban Environments*, volume 10008, page 100080P. International Society for Optics and Photonics, 2016.

[93] 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.

[94] Pedram Ghamisi, Yushi Chen, and Xiao Xiang Zhu. A self-improving convolution neural network for the classification of hyperspectral data. *IEEE Geoscience and Remote Sensing Letters*, 13(10):1537–1541, 2016.

[95] Weijia Li, Haohuan Fu, Le Yu, Peng Gong, Duole Feng, Congcong Li, and Nicholas Clinton. Stacked autoencoder-based deep learning for remote-sensing image classification: a case study of african land-cover mapping. *International Journal of Remote Sensing*, 37(23):5632–5646, 2016.

[96] Hongying Liu, Qiang Min, Chen Sun, Jin Zhao, Shuyuan Yang, Biao Hou, Jie Feng, and Licheng Jiao. Terrain classification with polarimetric SAR based on deep sparse filtering network. In *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*, pages 64–67. IEEE, 2016.

[97] Maoguo Gong, Jiaojiao Zhao, Jia Liu, Qiguang Miao, and Licheng Jiao. Change detection in synthetic aperture radar images based on deep neural networks. *IEEE transactions on neural networks and learning systems*, 27(1):125–138, 2016.

[98] Pablo F. Alcantarilla, Simon Stent, German Ros, Roberto Arroyo, and Riccardo Gherardi. Street-view change detection with deconvolutional networks. In *Robotics: Science and Systems*, 2016.

[99] Jiaojiao Zhao, Maoguo Gong, Jia Liu, and Licheng Jiao. Deep learning to classify difference image for image change detection. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 411–417. IEEE, 2014.

[100] Linzhi Su, Jiao Shi, Puzhao Zhang, Zhao Wang, and Maoguo Gong. Detecting multiple changes from multi-temporal images by using stacked denosing autoencoder based

change vector analysis. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 1269–1276. IEEE, 2016.

[101] Sushil Singh and Rajneesh Talwar. Review on different change vector analysis algorithms based change detection techniques. In *Image Information Processing (ICIIP), 2013 IEEE Second International Conference on*, pages 136–141. IEEE, 2013.

[102] Lorenzo Bruzzone and Diego Fernández Prieto. Automatic analysis of the difference image for unsupervised change detection. *Geoscience and Remote Sensing, IEEE Transactions on*, 38(3):1171–1182, 2000.

[103] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[104] Philip E. Dennison, Kerry Q. Halligan, and Dar A. Roberts. A comparison of error metrics and constraints for multiple endmember spectral mixture analysis and spectral angle mapper. *Remote Sensing of Environment*, 93(3):359–367, 2004.

[105] Nirmal Keshava. Distance metrics and band selection in hyperspectral processing with applications to material identification and spectral libraries. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(7):1552–1565, 2004.

[106] Moses Azong Cho, Pravesh Debba, Renaud Mathieu, Laven Naidoo, Jan Van Aardt, and Gregory P. Asner. Improving discrimination of savanna tree species through a multiple-endmember spectral angle mapper approach: Canopy-level analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4133–4142, 2010.

[107] Behnood Rasti, Paul Scheunders, Pedram Ghamisi, Giorgio Licciardi, and Jocelyn Chanussot. Noise reduction in hyperspectral imagery: Overview and application. *Remote Sensing*, 10(3):482, 2018.

[108] 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.

[109] Emmanuel Christophe, Julien Michel, and Jordi Inglada. Remote sensing processing: From multicore to GPU. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):643–652, 2011.

[110] Yan Ma, Haiping Wu, Lizhe Wang, Bormin Huang, Rajiv Ranjan, Albert Zomaya, and Wei Jie. Remote sensing big data computing: Challenges and opportunities. *Future Generation Computer Systems*, 51:47–60, 2015.

[111] Carlos Gonzalez, Sergio Sánchez, Abel Paz, Javier Resano, Daniel Mozos, and Antonio Plaza. Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing. *INTEGRATION, the VLSI journal*, 46(2):89–103, 2013.

[112] Bin Yang, Minhua Yang, Antonio Plaza, Lianru Gao, and Bing Zhang. Dual-mode FPGA implementation of target and anomaly detection algorithms for real-time hyperspectral imaging. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2950–2961, 2015.

[113] Wentai Zhang, Guojie Luo, Li Shen, Thomas Page, Peng Li, Ming Jiang, Peter Maass, and Jason Cong. FPGA acceleration by asynchronous parallelization for simultaneous image reconstruction and segmentation based on the mumford-shah regularization. In *SPIE Optical Engineering+ Applications*, pages 96000H–96000H. International Society for Optics and Photonics, 2015.

[114] Carlos González, Sergio Bernabé, Daniel Mozos, and Antonio Plaza. FPGA implementation of an algorithm for automatically detecting targets in remotely sensed hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(9):4334–4343, 2016.

[115] Peng Liu, Tao Yuan, Yan Ma, Lizhe Wang, Dingsheng Liu, Shasha Yue, and Joanna Kołodziej. Parallel processing of massive remote sensing images in a GPU architecture. *Computing and Informatics*, 33(1):197–217, 2014.

[116] Sergio Sánchez, Rui Ramalho, Leonel Sousa, and Antonio Plaza. Real-time implementation of remotely sensed hyperspectral image unmixing on GPUs. *Journal of Real-Time Image Processing*, 10(3):469–483, 2015.

[117] Ujwala Bhangale, Surya S. Durbha, Roger L. King, Nicolas H. Younan, and Rangaraju Vatsavai. High performance GPU computing based approaches for oil spill detection from multi-temporal remote sensing data. *Remote Sensing of Environment*, 2017.

[118] Sergio Sánchez, Abel Paz, Gabriel Martín, and Antonio Plaza. Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units. *Concurrency and Computation: Practice and Experience*, 23(13):1538–1557, 2011.

[119] Alexander Agathos, Jun Li, Dana Petcu, and Antonio Plaza. Multi-GPU implementation of the minimum volume simplex analysis algorithm for hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2281–2296, 2014.

[120] Jarno Mielikainen, Bormin Huang, and Hung-Lung Allen Huang. GPU-accelerated multi-profile radiative transfer model for the infrared atmospheric sounding interferometer. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 4(3):691–700, 2011.

[121] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Efficient classification of hyperspectral images on commodity GPUs using ELM-based techniques. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 1, 2014. ISBN:1-60132-282-8.

[122] Javier López-Fandiño, Pablo Quesada-Barriuso, Dora B. Heras, and Francisco Argüello. Efficient ELM-based techniques for the classification of hyperspectral remote sensing images on commodity GPUs. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, 8(6):2884 – 2893, 2015. DOI:10.1109/JSTARS.2014.2384133.

[123] Javier López-Fandiño, Blanca Priego, Dora B. Heras, Francisco Argüello, and Richard J. Duro. GPU projection of ECAS-II segmenter for hyperspectral images based on cellular automata. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(1):20–28, 2017. DOI:10.1109/JSTARS.2016.2588530.

[124] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Mauro Dalla Mura. GPU framework for change detection in multitemporal hyperspectral images. In *10th International Symposium on High-Level Parallel Programming and Applications (HLPP17)*, pages 115 – 132, 2017.

[125] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Mauro Dalla Mura. GPU framework for change detection in multitemporal hyperspectral images. *International Journal of Parallel Programming (IJPP)*, pages 1–21, 2017. DOI: 10.1007/s10766-017-0547-5.

[126] Javier López-Fandiño, Dora B. Heras, Francisco Argüello, and Richard J. Duro. CUDA multiclass change detection for remote sensing hyperspectral images using extended morphological profiles. In *2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, pages 404 – 409. IEEE, 2017. DOI: 10.1109/IDAACS.2017.8095113.

[127] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[128] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Efficient GPU multiclass change detection for multidimensional images in the presence of noise. In *SPIE Remote Sensing*. International Society for Optics and Photonics, 2018. Accepted.

[129] Javier López-Fandiño, Alberto S. Garea, Dora B. Heras, and Francisco Argüello. Stacked autoencoders for multiclass change detection in hyperspectral images. In *Geoscience and Remote Sensing Symposium (IGARSS), 2018 IEEE International*. IEEE, 2018. Accepted.

[130] Blanca Priego, Richard J. Duro, Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Evolutionary cellular automata based approach to high-dimensional image segmentation for GPU projection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1593–1600. IEEE, 2016. DOI: 10.1109/IJCNN.2016.7727388.

[131] Javier López-Fandiño, Dora B. Heras, and Francisco Argüello. Clasificación supervisada de imágenes de sensado remoto en GPU. In *XXV Jornadas de Paralelismo*, pages 319–325, 2014. ISBN: 978-84-697-0329-3.

[132] Alberto S. Garea, Álvaro Ordóñez, Dora B. Heras, and Francisco Argüello. HypeRvieW: an open source desktop application for hyperspectral remote-sensing data processing. *International Journal of Remote Sensing*, 37(23):5533–5550, 2016.

[133] USGS. Earth explorer. Available at: https://earthexplorer.usgs.gov/, Accessed February 2018.

[134] Roger N. Clark, Gregg A. Swayze, Richard Wise, K. Eric Livo, T. Hoefen, Raymond F. Kokaly, and Steve J. Sutley. USGS digital spectral library splib06a. *US geological survey, digital data series*, 231:2007, 2007.

[135] Prasad S. Thenkabail and John G. Lyon. *Hyperspectral remote sensing of vegetation*. CRC Press, 2016.

[136] Lalit Saxena and Leisa Armstrong. A survey of image processing techniques for agriculture. 2014.

[137] W. Pan, K. Qin, and Y. Chen. An adaptable-multilayer fractional Fourier transform approach for image registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(3):400–414, March 2009.

[138] Mahmudul Hasan, Mark Pickering, Antonio Robles-Kelly, Jun Zhou, and Xiuping Jia. Regisration of hyperspectral and trichromatic images via cross cumulative residual entropy maximisation. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2329–2332. IEEE, 2010.

[139] Lori Mann Bruce, Cliff H Koger, and Jiang Li. Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction. *Geoscience and Remote Sensing, IEEE Transactions on*, 40(10):2331–2338, 2002.

[140] Gordon Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory*, 14(1):55–63, 1968.

[141] Giorgio Lopez, Ettore Napoli, and Antonio G.M. Strollo. FPGA implementation of the CCSDS-123.0-b-1 lossless hyperspectral image compression algorithm prediction stage. In *Circuits & Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on*, pages 1–4. IEEE, 2015.

[142] S Kargozar Nahavandi, P. Ghamisi, L. Kumar, and M.S. Couceiro. A novel adaptive compression technique for dealing with corrupt bands and high levels of band correlations in hyperspectral images based on binary hybrid GA-PSO for big data compression. *International Journal of Computer Applications*, 109(8):18–25, 2015.

[143] Dora B. Heras, F. Argüello, J. Lopez Gomez, J.A. Becerra, and Richard J. Duro. Towards real-time hyperspectral image processing, a GP-GPU implementation of target identification. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, volume 1, pages 316–321. IEEE, 2011.

[144] Sergio Bernabe, Sebastián López, Antonio Plaza, and Roberto Sarmiento. GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis. *Geoscience and Remote Sensing Letters, IEEE*, 10(2):221–225, 2013.

[145] Sankar K. Pal, A. Ghosh, and B. Uma Shankar. Segmentation of remotely sensed images with fuzzy thresholding, and quantitative evaluation. *International Journal of Remote Sensing*, 21(11):2269–2300, 2000.

[146] James C. Tilton, Yuliya Tarabalka, Paul M. Montesano, and Emanuel Gofman. Best merge region-growing segmentation with integrated nonadjacent region object aggregation. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(11):4454–4467, 2012.

[147] Kevin Bernard, Yuliya Tarabalka, Jesus Angulo, Jocelyn Chanussot, and Jón Atli Benediktsson. Spectral–spatial classification of hyperspectral data based on a stochastic minimum spanning forest approach. *Image Processing, IEEE Transactions on*, 21(4):2008–2021, 2012.

[148] Claudio Persello and Lorenzo Bruzzone. Active learning for domain adaptation in the supervised classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 50(11):4468–4483, 2012.

[149] Zhuo Sun, Cheng Wang, Hanyun Wang, and Jonathan Li. Learn multiple-kernel svms for domain adaptation in hyperspectral data. *IEEE Geoscience and Remote Sensing Letters*, 10(5):1224–1228, 2013.

[150] Sergio Bernabé, Santiago Sánchez, Antonio Plaza, Sebastián López, Jón Atli Benediktsson, and Roberto Sarmiento. Hyperspectral unmixing on GPUs and multi-core processors: A comparison. *IEEE Journal of Selected topics in applied earth observations and remote sensing*, 6(3):1386–1398, 2013.

[151] Mauro Dalla Mura, Jón Atli Benediktsson, Francesca Bovolo, and Lorenzo Bruzzone. An unsupervised technique based on morphological filters for change detection in very high resolution images. *Geoscience and Remote Sensing Letters, IEEE*, 5(3):433–437, 2008.

[152] Nicola Falco, Mauro Dalla Mura, Francesca Bovolo, Jón Atli Benediktsson, and Lorenzo Bruzzone. Change detection in VHR images based on morphological attribute profiles. *Geoscience and Remote Sensing Letters, IEEE*, 10(3):636–640, 2013.

[153] Turgay Celik. Unsupervised change detection in satellite images using principal component analysis and K-means clustering. *Geoscience and Remote Sensing Letters, IEEE*, 6(4):772–776, 2009.

[154] Freek van der Meer. The effectiveness of spectral similarity measures for the analysis of hyperspectral imagery. *International journal of applied earth observation and geoinformation*, 8(1):3–17, 2006.

[155] Chein-I. Chang. Spectral information divergence for hyperspectral image analysis. In *Geoscience and Remote Sensing Symposium, 1999. IGARSS'99 Proceedings. IEEE 1999 International*, volume 1, pages 509–511. IEEE, 1999.

[156] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[157] J.-F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, 1998.

[158] Adrian N Evans and Xin U. Liu. A morphological gradient approach to color edge detection. *IEEE Transactions on Image Processing*, 15(6):1454–1463, 2006.

[159] P. Solle and L. Vincent. Watershed in digital spaces, an efficient algorithm based on immersion simulation. *IEEE Trans. Pattern Anal. Mach. Intell*, 13(6):583–598, 1991.

[160] Fernand Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, 1994.

[161] Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1, 2):187–228, 2000.

[162] John Von Neumann and Arthur Walter Burks. *Theory of self-reproducing automata*. University of Illinois Press Urbana, 1996.

[163] Pablo Quesada-Barriuso, Dora B. Heras, and Francisco Arguello. Efficient GPU asynchronous implementation of a watershed algorithm based on cellular automata. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 79–86. IEEE, 2012.

[164] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[165] Chen Chen, Junjun Jiang, Baochang Zhang, Wankou Yang, and Jianzhong Guo. Hyperspectral image classification using gradient local auto-correlations. In *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, pages 454–458. IEEE, 2015.

[166] Jean Serra. *Image analysis and mathematical morphology, v. 1*. Academic press, 1982.

[167] Martino Pesaresi and Jón Atli Benediktsson. A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE transactions on Geoscience and Remote Sensing*, 39(2):309–320, 2001.

[168] John A. Hartigan and Manchek A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[169] Geoffrey H Ball and David J. Hall. ISODATA, a novel method of data analysis and pattern classification. Technical report, Stanford research inst Menlo Park CA, 1965.

[170] Yuliya Tarabalka, Jocelyn Chanussot, and Jón Atli Benediktsson. Segmentation and classification of hyperspectral images using minimum spanning forest grown from automatically selected markers. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 40(5):1267–1279, 2010.

[171] Mathieu Fauvel. Spectral and spatial methods for the classification of urban remote sensing data. *Institut Technologique de Grenoble–Université d'Islande, Thèse de Doctorat*, 2007.

[172] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[173] Pedram Ghamisi, Jón Atli Benediktsson, and Magnus Orn Ulfarsson. Spectral–spatial classification of hyperspectral images based on hidden markov random fields. 2014.

[174] Alim Samat, Peijun Du, Sicong Liu, Jun Li, and Liang Cheng. E2LMs: Ensemble extreme learning machines for hyperspectral image classification. *IEEE Journal of selected topics in applied earth observations and remote sensing*, 7(4):1060–1069, 2014.

[175] M. Lichman. UCI machine learning repository, Accessed Februrary 2018. Available at: http://archive.ics.uci.edu/ml.

[176] Shigeo Abe. *Support vector machines for pattern classification*, volume 53. Springer, 2005.

[177] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[178] Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.

[179] Pierre Courrieu. Fast computation of moore-penrose inverse matrices. *arXiv preprint arXiv:0804.4809*, 2008.

[180] William K. Pratt. Correlation techniques of image registration. *IEEE transactions on Aerospace and Electronic Systems*, (3):353–358, 1974.

[181] William M Wells, Paul Viola, Hideki Atsumi, Shin Nakajima, and Ron Kikinis. Multimodal volume registration by maximization of mutual information. *Medical image analysis*, 1(1):35–51, 1996.

[182] Álvaro Ordóñez, Francisco Argüello, and Dora B. Heras. Fourier–Mellin registration of two hyperspectral images. *International Journal of Remote Sensing*, 38(11):3253–3273, 2017.

[183] Álvaro Ordóñez, Francisco Argüello, and Dora B. Heras. GPU accelerated FFT-based registration of hyperspectral scenes. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(11):4869–4878, 2017.

[184] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[185] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[186] Allan A. Nielsen. Kernel based orthogonalization for change detection in hyperspectral image data. In *6th EARSeL Workshop on Imaging Spectroscopy*, 2013.

[187] John B. Collins and Curtis E. Woodcock. An assessment of several linear change detection techniques for mapping forest mortality using multitemporal Landsat TM data. *Remote sensing of environment*, 56(1):66–77, 1996.

[188] Dengsheng Lu, P. Mausel, E. Brondizio, and Emilio Moran. Change detection techniques. *International Journal of Remote Sensing*, 25(12):2365–2401, 2004.

[189] J.S. Deng, K. Wang, Y.H. Deng, and G.J. Qi. PCA-based land-use change detection and analysis using multitemporal and multisensor satellite data. *International Journal of Remote Sensing*, 29(16):4823–4838, 2008.

[190] Jeffrey L. Michalek, Thomas W. Wagner, Joseph J. Luczkovich, and Richard W. Stoffle. Multispectral change vector analysis for monitoring coastal marine environments. *Photogrammetric Engineering & Remote Sensing*, 59(3):381–384, 1993.

[191] Jin Chen, Peng Gong, Chunyang He, Ruiliang Pu, and Peijun Shi. Land-use/land-cover change detection using improved change-vector analysis. *Photogrammetric Engineering & Remote Sensing*, 69(4):369–379, 2003.

[192] Francesca Bovolo, Silvia Marchesi, and Lorenzo Bruzzone. A framework for automatic and unsupervised detection of multiple changes in multitemporal images. *IEEE Transactions on Geoscience and Remote Sensing*, 50(6):2196–2212, 2012.

[193] Sicong Liu, Qian Du, Xiaohua Tong, Alim Samat, Lorenzo Bruzzone, and Francesca Bovolo. Multiscale morphological compressed change vector analysis for unsupervised multiple change detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(9):4124–4137, 2017.

[194] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[195] John A. Hartigan. Clustering algorithms. 1975.

[196] Arthur P. Dempster, Nan M. Laird, Donald B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[197] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[198] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607. ACM, 2002.

[199] Mark Harris. Optimizing cuda. *SC07: High Performance Computing With CUDA*, 2007.

[200] Pablo Quesada-Barriuso, Francisco Argüello, Dora B. Heras, , and Jón Atli Benediktsson. Wavelet based classification of hyperspectral images using extended morphological profiles on graphics processing units. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2962 – 2970, 2015.

[201] NVIDIA. *CUBLAS Library User Guide*, 2013.

[202] Stanimire Tomov, Rajib Nath, Peng Du, and Jack Dongarra. MAGMA users' guide. *ICL, UTK (November 2009)*, 2011.

[203] NVIDIA. CULA tools. Available at: http://www.culatools.com/, Accessed Februrary 2018.

[204] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[205] NVIDIA. The NVIDIA CUDA deep neural network library (cuDNN), Accessed Februrary 2018. Available at: https://developer.nvidia.com/cudnn.

[206] NVIDIA. *CUDA Profiler Users Guide*, Accessed March 2018. Available at: http://docs.nvidia.com/cuda/profiler-users-guide/.

[207] CESGA. *Finis Terrae II Quick start guide.*, Accessed Februrary 2018. Available at: https://cesga.es/en/paginas/descargaDocumento/id/231.

[208] Russell G. Congalton. A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, 37(1):35–46, 1991.

[209] Danilo De Donno, Alessandra Esposito, Luciano Tarricone, and Luca Catarinucci. Introduction to GPU computing and CUDA programming: A case study on FDTD [EM programmer's notebook]. *IEEE Antennas and Propagation Magazine*, 52(3):116–122, 2010.

[210] Brandt Tso and Paul M. Mather. *Classification methods for remotely sensed data*. Number LC-0431. CRC press, 2009.

[211] Center for Remote Sensing (CRS). ROSIS sensor website, Accessed Februrary 2018. Available at: https://crs.hi.is/?page_id=877.

[212] California Institute of Technology Jet Propulsion Laboratory. AVIRIS sensor website, Accessed Februrary 2018. Available at: https://aviris.jpl.nasa.gov/.

[213] United States Geological Survey. Hyperion sensor website, Accessed Februrary 2018. Available at: https://eo1.usgs.gov/sensors/hyperion.

[214] United States Geological Survey. Thematic mapper sensor website, Accessed Februrary 2018. Available at: https://lta.cr.usgs.gov/TM.

[215] Computational Intelligence Group from the Basque University (UPV/EHU). Hyperspectral remote sensing scenes, Accessed February 2018. Available at: http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes.

[216] Sicong Liu, Qian Du, Xiaohua Tong, Alim Samat, Haiyan Pan, and Xiaolong Ma. Band selection-based dimensionality reduction for change detection in multi-temporal hyperspectral images. *Remote Sensing*, 9(10):1008, 2017.

[217] Francesca Bovolo and Lorenzo Bruzzone. A theoretical framework for unsupervised change detection based on change vector analysis in the polar domain. *IEEE Transactions on Geoscience and Remote Sensing*, 45(1):218–236, 2007.

[218] Louisa Lam and Ching Y. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 27(5):553–568, 1997.

[219] Pablo Quesada-Barriuso, Francisco Argüello, and Dora B. Heras. Efficient segmentation of hyperspectral images on commodity GPUs. In *KES*, volume 243, pages 2130–2139, 2012.

[220] Pablo Quesada-Barriuso, Dora B. Heras, and Francisco Argüello. Efficient 2D and 3D watershed on graphics processing unit: block-asynchronous approaches based on cellular automata. *Computers & Electrical Engineering*, 39(8):2638–2655, 2013.

[221] Yuliya Tarabalka, Jón Atli Benediktsson, and Jocelyn Chanussot. Spectral–spatial classification of hyperspectral imagery based on partitional clustering techniques. *IEEE Transactions on Geoscience and Remote Sensing*, 47(8):2973–2987, 2009.

[222] Jing Wang and Chein-I. Chang. Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. *IEEE transactions on geoscience and remote sensing*, 44(6):1586–1600, 2006.

[223] Hui Kong, Lei Wang, Eam Khwang Teoh, J.-G. Wang, and Ronda Venkateswarlu. A framework of 2D fisher discriminant analysis: application to face recognition with small number of training samples. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1083–1088. IEEE, 2005.

[224] Begüm Demir, Francesca Bovolo, and Lorenzo Bruzzone. Detection of land-cover transitions in multitemporal remote sensing images with active-learning-based compound classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(5):1930–1941, 2012.

List of Figures

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables