Avances en Arquitectura y Tecnología de Computadores

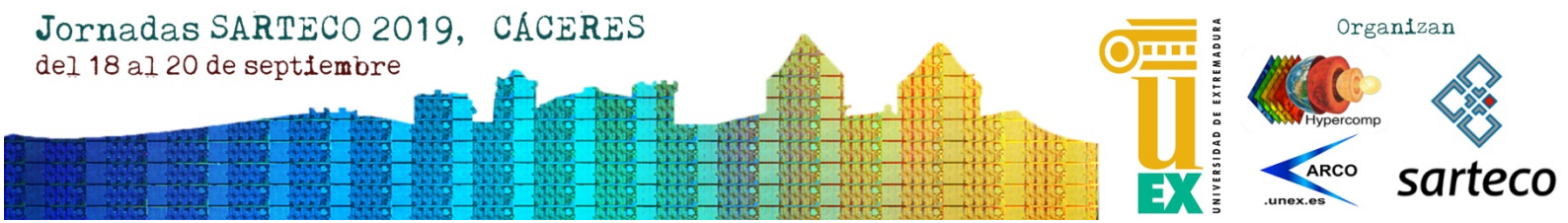
Actas de las Jornadas SARTECO 2019

Cáceres, 18 a 20 de septiembre de 2019

Editado por:

Miguel Ángel Vega Rodríguez

Antonio J. Plaza Miguel



Avances en Arquitectura y Tecnología de Computadores
Actas de las Jornadas SARTECO 2019

Editores: Miguel Ángel Vega Rodríguez y Antonio J. Plaza Miguel

Septiembre 2019, Jornadas SARTECO

ISBN-13: 978-84-09-12127-4

Cáceres, España

Publicado por:

Universidad de Extremadura. Servicio de Publicaciones

Plaza de Caldereros, 2 - Planta 3ª, 10003 Cáceres (España)

Tel.: 927 257 041; Fax: 927 257 046

Correo-e: publicac@unex.es http://www.unex.es/publicaciones

Prólogo

La Sociedad de Arquitectura y Tecnología de Computadores (SARTECO) presenta, una vez más, las Jornadas SARTECO, que integran las XXX Jornadas de Paralelismo (JP2019) y las IV Jornadas de Computación Empotrada y Reconfigurable (JCER2019). Además, en el contexto de las Jornadas SARTECO se celebra también el V Concurso "Tu Tesis en 3 Minutos" (T3M2019), que pretende premiar los mejores trabajos de tesis recientes en el área, y el III Encuentro WSARTECO de investigadoras en TIC.

Este año las Jornadas se celebran en Cáceres, contando con un excelente programa de sesiones técnicas con un total de 70 artículos que se presentarán en las JP2019 y otros 17 artículos en las JCER2019, además de 3 ponencias invitadas. Aunque la mayoría de autores de estos artículos son españoles, también hay coautores con afiliación de otros muchos países (en orden alfabético): Alemania, Argentina, Brasil, Chile, China, Ecuador, Francia, Grecia, Hungría, Italia, Japón, Marruecos, México, Perú, Portugal y Reino Unido. A fecha de edición del presente libro de actas, el número de asistentes (incluyendo conferenciantes invitados y comité de organización) es de unas 150 personas.

En conclusión, la celebración conjunta de estas actividades y eventos de carácter científico-técnico constituye un referente nacional imprescindible para la comunidad científica agrupada en SARTECO. Estas jornadas reúnen a un nutrido grupo de investigadores, procedentes de diferentes universidades y centros de investigación, con el objeto de intercambiar experiencias, presentar y debatir resultados de investigación, facilitar colaboraciones y sinergias entre grupos, y potenciar nuevas oportunidades de transferencia tecnológica a la industria.

Desde la organización de esta nueva edición de las Jornadas SARTECO, os deseamos a todos una placentera y productiva estancia en Cáceres.

Julio de 2019

Comités Organizadores

Comité de Dirección de las Jornadas SARTECO

- Inmaculada García Fernández (UMA) (Presidenta)
- Victor Viñals Yufera (UNIZAR) (Vicepresidente)
- Katzalin Olcoz Herrero (UCM) (Secretaria)
- Francisco Tirado Fernández (Presidente de Honor)

Organizadores Jornadas SARTECO2019 y Comité JP2019

- Antonio J. Plaza Miguel (UEX)
- Miguel A. Vega Rodríguez (UEX)
- Javier Plaza Miguel (UEX)
- José M. Granado Criado (UEX)
- Sergio Santander Jiménez (UEX)
- Juan M. Haut Hurtado (UEX)
- Mercedes E. Paoletti Ávila (UEX)
- Elena Paoletti Ávila (UEX)

Comité de Coordinación JCER2019

- Jesús González Peñalver (UGR)
- Sergio Cuenca Asensi (UA)
- Miguel A. Vega Rodríguez (UEX)
- Miguel Damas Hermoso (UGR)
- Antonio Martínez Álvarez (UA)
- Gustavo Sutter (UAM)
- Ignacio Bravo (UAH)
- José Torres (UV)
- Jordi Carrabina (UAB)
- Juan Suardíaz (UPCT)
- Jesús Barba Romero (UCLM)
- Goiuria Sagardui Mendieta (UMONDRAGON)
- Jorge Portilla Berrueco (UPM)

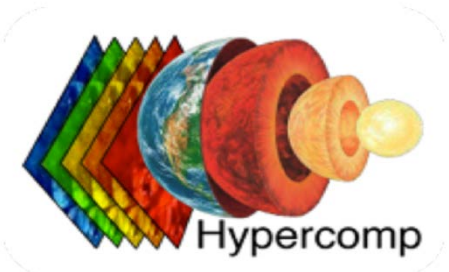
Comité de Programa JCER2019

- Jesús Barba Romero (Universidad de Castilla-La Mancha)
- Marta Beltrán (Universidad Rey Juan Carlos)
- Francisco J. Bonin-Font (Universidad de las Islas Baleares)
- Ignacio Bravo (Universidad de Alcalá)
- David Castells (Universidad Autónoma de Barcelona)
- Javier Castillo (Universidad Rey Juan Carlos)
- Sergio Cuenca Asensi (Universidad de Alicante)
- Juan Carlos Díaz Martín (Universidad de Extremadura)
- Luis Entrena (Universidad Carlos III de Madrid)
- Leire Etxeberria (Universidad de Mondragón)
- Eduard Fernández-Alonso (Recore Systems)
- Rodolfo García-Bermúdez (Universidad Técnica de Manabí, Ecuador)
- Juan Antonio Gómez Pulido (Universidad de Extremadura)
- Jesús González Peñalver (Universidad de Granada)
- José M. Granado Criado (Universidad de Extremadura)
- Juan A. Holgado-Terriza (Universidad de Granada)
- Miren Illarramendi Rezabal (Universidad de Mondragón)
- Antonio Jimeno-Morenilla (Universidad de Alicante)
- Gustavo Marrero Callico (Universidad de las Palmas de Gran Canaria)
- Antonio Martínez Álvarez (Universidad de Alicante)
- Joaquín Olivares (Universidad de Córdoba)
- Gabriel Oliver (Universidad de las Islas Baleares)
- Alberto Ortiz (Universidad de las Islas Baleares)
- Fernando Pardo (Universidad de Valencia)
- Jon Pérez (Ikerlan)
- Jorge Portilla Berrueco (Universidad Politécnica de Madrid)
- Francisco A. Pujol (Universidad de Alicante)
- Francisco Ramos (Schneider Electric)
- Lluis Ribas-Xirgo (Universidad Autónoma de Barcelona)
- José Torres (Universidad de Valencia)
- Miguel A. Vega Rodríguez (Universidad de Extremadura)
- Félix Jesús Villanueva Molina (Universidad de Castilla-La Mancha)

Comité T3M2019

- Inmaculada García Fernández (UMA) (Presidenta)
- Katzalin Olcoz Herrero (UCM) (Secretaria)
- Francisco Tirado Fernández (Presidente de Honor)
- Enrique S. Quintana Ortí (UJI) (Vocal, Junta directiva SARTECO)
- Miquel Moretó Planas (UPC) (Vocal, Junta directiva SARTECO)

Patrocinadores



Escuela Politécnica

TC2
DEPARTAMENTO DE TECNOLOGÍA
COMPUTADORES Y COMUNICACIONES

AYUNTAMIENTO
cáceres

DIPUTACIÓN DE CÁCERES

INSTITUCIÓN CULTURAL
EL BROCENSE
DIPUTACIÓN DE CÁCERES

Índice

Ponencias invitadas

Parte 1 - XXX Jornadas de Paralelismo

Aplicaciones de la computación de altas prestaciones

Computación eficiente de perfiles de difusión para la extracción de información espectral-espacial

Álvaro Acción, Dora B. Heras, y Francisco Argüello[1]

Resumen— **En el ámbito del procesado de imagen multi e hiperespectral es considerado beneficioso el incluir información espacial conjuntamente con la espectral cuando se realiza el procesamiento de dichas imágenes. Los perfiles son transformaciones que extraen información espectral y espacial a diferentes niveles de granularidad en la imagen. Es posible definir perfiles basados en operaciones de difusión anisotrópica por medio de ecuaciones diferenciales parciales no lineales. Su principal ventaja es la de preservar las características morfológicas distintivas de las imágenes, como por ejemplo los bordes, en diferentes escalas. En este documento, reducimos drásticamente el alto coste computacional asociado a la construcción de perfiles de difusión para imágenes hiperespectrales mediante el uso de GPUs. En particular, proponemos un enfoque computacional de bajo coste para esta tarea empleando la arquitectura CUDA.**

I. Introducción

LAS imágenes hiperespectrales están siendo utilizadas cada vez en más contextos de aplicación debido al abaratamiento de los sensores requeridos para obtenerlas, promoviendo so uso recientemente en campos como la agricultura y la monitorización ambiental [1].

Es comúnmente aceptado en el procesamiento de imágenes hiperespectrales o multiespectrales que la extracción de información espectral-espacial mejora los resultados respecto de utilizar solamente información espectral por muy abundante que esta sea. Los perfiles morfológicos (MP) [2] son una de las técnicas más comunes utilizadas para extraer información espectral-espacial de imágenes hiperespectrales. La construcción de MPs se realiza aplicando a cada banda de la imagen transformaciones de apertura y cierre con un elemento estructural (SE) de tamaño creciente sobre las bandas obtenidas.

La difusión anisotrópica es una técnica que puede ser usada para mejorar las imágenes multiespectrales aumentando su relación señal/ruido [3] y, en particular, puede usarse para la generación de perfiles extendidos que sirven como entrada para una posterior etapa de procesado como podría ser la clasificación [4]. La aplicación del filtrado de difusión no lineal ha mostrado un gran potencial [5] en comparación con el filtrado lineal tradicional debido a su capacidad de preservar las características morfológicas distintivas de las imágenes tales como los bordes, o incluso resaltarlas.

Los esquemas utilizados para calcular la difusión no lineal presentan coste computacional elevado. La introducción de los esquemas de *Fast Explicit Diffusion* (FED) [6] ha permitido realizar estos cálculos de forma computacionalmente más eficiente, al tiempo que ofrecen una mayor precisión que los esquemas semi-implícitos existentes previamente.

Las unidades de procesamiento gráfico (GPUs) son plataformas computacionales de alto rendimiento que pueden ser usadas para el procesamiento eficiente de imágenes hiperespectrales alcanzado una ejecución en tiempo real en muchas de las aplicaciones [7], [8], [9]. La disponibilidad de GPUs de consumo capaces de realizar computación paralela permite alcanzar de forma económica cotas de rendimiento antes reservadas a sistemas como clústers o infraestructuras de altas prestaciones.

En este documento, proponemos un algoritmo programado en CUDA para tarjetas de consumo NVIDIA que realiza la extracción eficiente de información espectral-espacial en imágenes hiperespectrales. Se basa en la computación de perfiles construidos mediante filtrado realizado a través de difusión no lineal, a los que llamaremos ADPs (Anisotropic Diffusion Profiles). La unión de ADPs para diferentes bandas o componentes de la imagen producirá el EADP (Extended Anisotropic Diffusion Profile). La difusión no lineal utilizada está basada en FED, debido a su reducido coste computacional y su idoneidad para la computación paralela. Este método presenta la ventaja de que el número de componentes de perfil es bajo, lo que reduce el tiempo de ejecución de una posible clasificación posterior.

El documento está organizado en cuatro secciones. La Sección II presenta una breve introducción a los conceptos de difusión no lineal y al esquema FED utilizado para implementarla. La Sección III presenta los perfiles, sus características y la implementación en CUDA. La Sección IV contiene la evaluación de rendimiento para la clasificación propuesta sobre los conjuntos de datos de test.

II. Filtrado de difusión no lineal

La siguiente sección presentará brevemente el concepto de difusión no lineal.

La formulación clásica para la ecuación de difusión es la siguiente:

[1]Los autores pertenecen al Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, España. (E-mail: alvaro.accion.montes@usc.es, dora.blanco@usc.es, francisco.arguello@usc.es)

$$\frac{\delta L}{\delta t} = \mathrm{div}(\mathrm{c(x,y,t)} \cdot \nabla \mathrm{L}). \quad (1)$$

En la ecuación anterior, div representa el operador de divergencia; ∇, el operador gradiente y c, también llamada función de conductividad, es una función que controla la difusión y la adapta a la estructura de la imagen. Perona y Malik [10] propusieron una función c variable y adaptativa que reduce el suavizado a través de los bordes y se elige en función de la magnitud del gradiente,

$$c(x,y,t) = g(|\nabla L_\sigma(x,y,t)|), \quad (2)$$

donde L_σ representa la imagen original luego de ser suavizada por el *kernel* Gaussiano de media 0 y varianza σ^2. La variable t representa de nuevo el tiempo, pero también puede considerarse como un valor de escala que controla el nivel de detalle de la imagen resultante.

Perona y Malik [10] describieron dos formulaciones diferentes para la función de conductividad, que se muestran en las ecuaciones (3) y (4).

$$g_1 = \exp\left(-\frac{|\nabla \mathrm{L}_\sigma|^2}{\mathrm{k}^2}\right), \quad (3)$$

$$g_2 = \frac{1}{1 + \frac{-|\nabla L_\sigma|^2}{k^2}}, \quad (4)$$

El parámetro k, también llamado parámetro de contraste, sirve como un umbral que permite la difusión hacia atrás. Los valores más altos de k generalmente implican que se suavicen los bordes de bajo contraste.

A. Fast Explicit Diffusion

La difusión explícita rápida (FED) es un esquema numérico eficiente que se utiliza para resolver problemas de ecuaciones parabólicas y elípticas en derivadas parciales.

FED explota el hecho de que el *kernel* Gaussiano se puede aproximar mediante cualquier *kernel* 1-D simétrico con los pesos de los coeficientes w_k que cumplen la condición $w_k = w_{-k}, \forall k \in \{1,\ldots,n\}$ y $\sum_{k=0}^{n} w_k = 1$. FED funciona realizando M ciclos compuestos por n pasos de difusión explícitos. En cada paso, se usa un τ_j variable. El valor de M controla la calidad de la aproximación, produciendo los valores más altos aproximaciones con un error más bajo a costa de una mayor complejidad.

En notación matricial, el ciclo FED se puede definir como:

$$L^{j+1} = (I + \tau_j A(L))\, L^j, \quad j = 0,\ldots,n-1, \quad (5)$$

donde L^{j+1} es la solución al problema de difusión definido por la ec. (2) en un paso dado y A es la matriz de conductividad calculada a partir de c.

Los parámetros FED se pueden expresar como,

$$\tau_j = \frac{\tau_{max}}{2\cos^2(\pi \cdot \frac{2\mathrm{j}+1}{4\mathrm{n}+2})}, \quad (6)$$

$$n = \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{12T}{M\tau_{\mathrm{máx}}}} \right\rceil, \quad (7)$$

donde τ_j es el tamaño de paso de la $j$ésima iteración resultante de la factorización del *box filter* y τ_{max} es el tamaño de paso máximo que no viola la condición de estabilidad del esquema explícito. Al conocer τ_{max}, es posible obtener el número de pasos n usando la ec. (7). Para valores de n grandes, el τ_j resultante puede ser ostensiblemente más grande que la condición de estabilidad [11].

III. Perfiles de difusión anisotrópica extendidos

En esta sección describimos la implementación en CUDA del algoritmo para la generación del perfil extendido (EADP) a partir del apilamiento de perfiles de difusión (ADPs). Se detallan también las técnicas que han permitido obtener una versión eficiente del código en CUDA, así como los resultados experimentales. Se ha analizado el rendimiento para identificar posibles cuellos de botella y decidir las optimizaciones de código que tienen un mayor impacto en el tiempo de ejecución.

Como se puede observar en la Fig. 1, cada ADP se construye aplicando un filtro de difusión no lineal a una imagen en escala de grises. En nuestro caso corresponde a una componente principal resultante de la aplicación del análisis de componentes principales (PCA) a la imagen hiperespectral original.

Una vez que se obtienen los componentes principales de la imagen, el algoritmo de difusión no lineal aplicará C instancias de difusión a cada $\boldsymbol{X}_i$ utilizando C tiempos de proceso T_i diferentes. Esta estrategia generará componentes con un nivel de detalle descendente y, por lo tanto, contendrá diferente información espacial de la imagen. El primer componente del ADP es siempre la componente principal de partida.

Finalmente, cada ADP tendrá $C+1$ componentes donde C es el número de instancias de difusión aplicadas, creando imágenes con niveles de detalle decrecientes. El EADP final tendrá un tamaño de $N \times (C+1)$, donde N es el número de componentes principales retenidos.

Denotando $\mathrm{Diff(X_i, T_i)}$ el proceso de difusión, un ADP se define como:

$$\mathrm{ADP(X_i)} = \{\mathrm{X_i}, \mathrm{Diff(X_i, T_1)}, \ldots, \mathrm{Diff(X_i, T_c)}\} \quad (8)$$

En la Fig. 2 se muestra un ejemplo de un ADP donde se puede apreciar cómo a partir de la componente inicial situada a la izquierda en la figura, el nivel de detalle se reduce de izquierda a derecha, es decir, a medida que se incorporan nuevos componentes de difusión. El EADP es el conjunto de todos

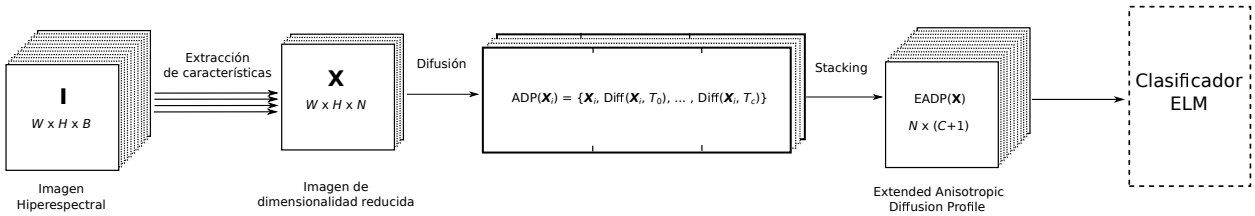


Fig. 1: Esquema propuesto para la clasificación de imágenes hiperespectrales basado en la extracción de información espacial mediante el uso de EADP.

los ADP resultantes de la aplicación del proceso de difusión:

$$\mathrm{EADP}(\mathbf{X}) = \{\mathrm{ADP}(\mathrm{X}_1), ..., \mathrm{ADP}(\mathrm{X}_\mathrm{N})\} \quad (9)$$

Los requisitos de memoria del algoritmo que construye el EADP dependen solo de las dimensiones de la imagen hiperespectral que se procesa y del número de componentes después de la etapa de extracción de características. El uso de memoria permanece constante durante toda la ejecución y requiere 4 *buffers*. La cantidad de memoria requerida es de aproximadamente $4WHN$ palabras.

A. Implementación en GPU

CUDA es un modelo de programación y arquitectura de computación en paralelo desarrollado por NVIDIA que permite la ejecución de funciones relativamente simples, llamadas *kernels*, dentro de una GPU. Se ha desarrollado como primera versión un código para ser ejecutado en una única GPU de consumo.

Para conseguir un código eficiente en CUDA se han aplicado múltiples técnicas de optimización. Dichas técnicas están basadas en una gran parte en optimizar el uso de la jerarquía de memoria: haciendo cálculos *in-place*, tratando de fomentar la reutilización de datos, usando memoria *pinned* para las transferencias entre CPU y GPU, o usando operaciones atómicas en memoria compartida frente a realizarlas en memoria global debido al soporte específico para operaciones atómicas existente desde la arquitectura Maxwell.

Otra técnica para evitar la latencia de acceso a la memoria compartida es realizar operaciones de reducción utilizando primitivas a nivel de *warp*. Se han utilizado también tipos de datos vectoriales para maximizar el paralelismo de instrucciones y optimizar los accesos a memoria. Finalmente, se han usado librerías de alto rendimiento como cuBLAS, cuSOLVER y cuSPARSE para la realización de algunas operaciones.

Tal como se muestra en el pseudocódigo presentado en el algoritmo 1, tras una etapa de extracción de características, que en este caso se ha realizado en GPU mediante el algoritmo PCA [12], se ha procedido al cálculo de la difusividad.

El cálculo de la matriz comienza con la carga inicial de $\boldsymbol{X}_i$ en la memoria de la GPU. Una vez cargada la componente, se realizan dos convoluciones con un

Algorithm 1 Pseudocódigo la construcción de EADP en CUDA

Entrada:
$\boldsymbol{X}_i$: Componente principal, $i = 1..N$.
σ^2: Varianza del filtro Gaussiano.
$\{T_1, \ldots, T_c\}$: lista de tiempos de proceso.
Salida:
EADP($\mathbf{X}$): EADP de la imagen $\mathbf{X}$.

```
1:  EADP ← ∅
2:  for i=1 →N do
3:      ADP_i ← ∅
        Cálculo de matriz de difusividad
4:      X_{i,σ} ← < apply_row_conv >(X_i, G_σ(X_i))        ▷ SM+GM
5:      X_{i,σ} ← < apply_col_conv >(X_{i,σ}, G_σ(X_i))    ▷ SM+GM
6:      ∇X_{i,σ} ← < scharr >(X_{i,σ}, Scharr(X_{i,σ}))    ▷ SM+GM
7:      max ← < reduce_max >(X_{i,σ})                      ▷ REG+GM
8:      histo ← < create_histogram >(∇X_{i,σ}, max)        ▷ SM+GM
9:      k ← get_bin(histo, 0,7)                            ▷ SM+GM
10:     A(∇X_{i,σ}) ← < pm2_coefficients >(∇X_{i,σ}, k)    ▷ GM
        Cálculo de proceso FED
11:     X_i^0 ← X_i
12:     for c = 1 → C do
13:         τ_j → fed_tau_by_process_time(T_c)
14:         X_i^{c,0} ← X_i^{c−1}
15:         for j = 1 → n do
16:             ΔX_i^{c,j} ← < fed_nld_step >(X_i^{c,j−1}, A(∇X_{i,σ}))
                                                           ▷ SM+GM
17:             X_i^{c+1,j+1} ← < fed_nld_update >(X_i^{c,j}, ΔX_i^{c,j})
                                                           ▷ GM
18:         end for
19:         ADP_c ← X^{c,n} ∪ ADP_c
20:     end for
21:     EADP ← ADP_c ∪ EADP
22: end for
```

kernel Gaussiano para suavizar la imagen original (líneas 4-5 en el pseudocódigo). Cada convolución es realizada por un *kernel* específicamente diseñado para explotar la localidad de la memoria dependiendo de la dirección de la operación. A continuación, el *kernel* que calcula las derivadas de la imagen obtiene $\nabla \boldsymbol{X}_{i,\sigma}$ (línea 6). Después de eso, el valor máximo de la imagen suavizada, $G_\sigma(\boldsymbol{X}_i)$, se obtiene con un *kernel* de reducción (línea 7). Dicho máximo se usa en la creación de un histograma.

Un nuevo *kernel* tomará $\nabla \boldsymbol{X}_{i,\sigma}$ y computará un histograma con las distancias entre las derivadas vertical y horizontal (línea 8), almacenándolas atómicamente en la memoria compartida y luego sumándolos atómicamente al histograma en la memoria global. El parámetro de contraste se calcula en CPU (línea 9), debido a la baja complejidad computacional de la tarea. Por último, la matriz de difusividad se calcula aplicando la ec. (4) a cada píxel de la imagen (línea 10).

El cálculo inicial del número de pasos y τ_j se ejecutan en la CPU (línea 13). El proceso de difusión

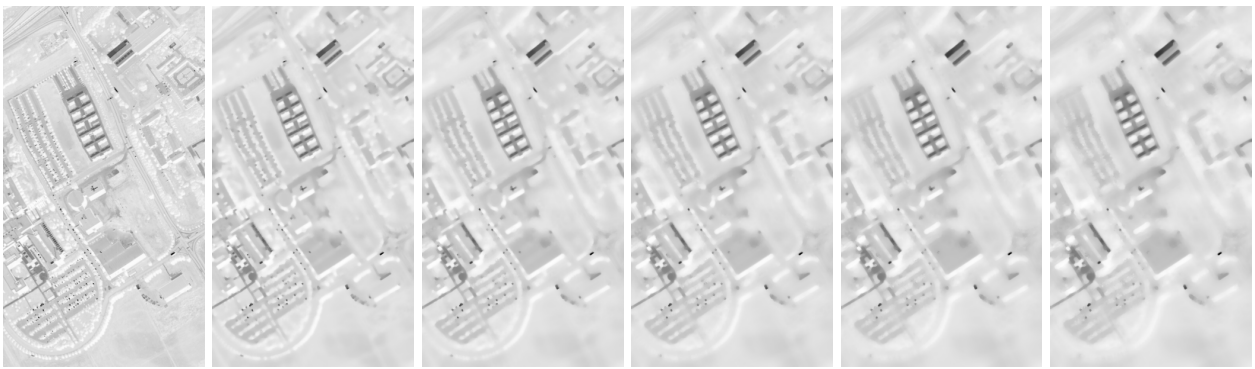


Fig. 2: La imagen representa un ADP para $N = 1$, $C = 5$ y de izquierda a derecha, una componente principal de la imagen seguida de las componentes resultantes de la difusión.

realiza la actualización de $\boldsymbol{X}_i^{c+1,j+1}$ de forma iterativa dentro de un *kernel* (líneas 16-17). Por último se añade cada componente al ADP correspondiente y este al EADP global (líneas 18-21).

IV. Evaluación del rendimiento

Esta sección describe los conjuntos de datos utilizados, explica los parámetros seleccionados para las pruebas y analiza la calidad de la propuesta presentada. Por un lado, se analiza el tiempo de ejecución en comparación con una difusión computada utilizando solamente la CPU multinúcleo de los experimentos. Por otro lado, para demostrar la efectividad del enfoque propuesto, incluimos un análisis de precisión de la clasificación obtenida utilizando el EADP comparando con las técnicas más comunes en la literatura.

Para realizar los experimentos que permiten evaluar esta técnica se han utilizado cuatro imágenes hiperespectrales reales. Por un lado, Indian Pines (IndianP) y Salinas Valley (Salinas), ambas tomadas por el sensor AVIRIS de la NASA. La resolución espacial es de 20 metros/píxel y cubre un rango espectral de 400 a 2500 nm, con 220 bandas espectrales en total. Los datos de referencia disponibles para cada imagen se dividen en dieciséis clases. IndianP y Salinas tienen unas dimensiones de 145 x 145 y 512 x 217 píxeles respectivamente. Por otro lado, Pavia University (PaviaU) y Pavia Center (PaviaC), adquiridas por el sensor ROSIS-03 en la ciudad de Pavia, Italia. Su resolución espacial es de 2,6 metros/píxel y cubre el rango espectral de 430 a 860 nm, con un total de 103 bandas espectrales. Los datos de referencia contienen nueve clases.Las dimensiones espaciales de PaviaU son 610 x 340 píxeles y las de PaviaC de 610 x 610 píxeles.

Los experimentos se llevaron a cabo utilizando una estación de trabajo con CPU Intel Core i5 8400 de 6 núcleos que funciona a 2,80 GHz y 32 GB de RAM. También dispone de una GPU NVIDIA GeForce GTX 1060 de 6 GB. Todos los experimentos se ejecutaron bajo Ubuntu Linux 16.04 de 64 bits y se compilaron con GCC versión 6.4.0 y CUDA toolkit 9.1.

Respecto del esquema FED se seleccionaron los valores de parámetros que respetasen un compromiso razonable entre precisión y velocidad, siendo estos valores $M = 1$, $\tau_{max} = 0{,}25$. Para la clasificación se ha utilizado el clasificador ELM detallado en [13]. La ejecución de ELM requiere fijar el número de muestras seleccionadas de cada clase para entrenar, que fijamos a 200 y el número de neuronas de la capa oculta que ha sido de 250 neuronas.

TABLA I: Comparativa en términos de clasificación con perfiles morfológicos

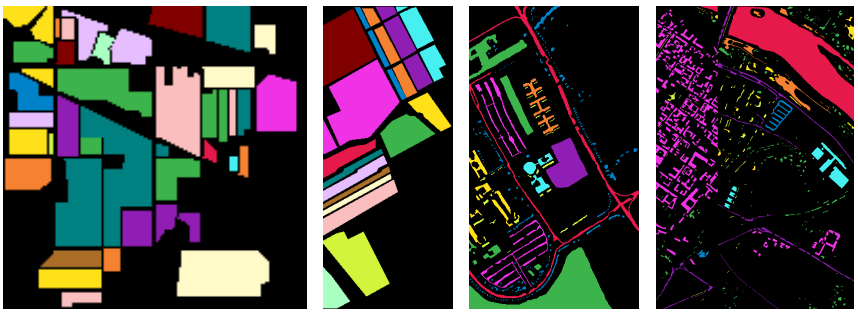
Dataset	Método	**OA**	$\boldsymbol{\kappa}$
Salinas	PCA-EADP	98.40	98.25
	WT-EMP [14]	91.67	90.70
	WTSS-EMP [14]	98.44	98.30
PaviaC	PCA-EADP	99.20	98.85
	WT-EMP [14]	99.54	99.30
	WTSS-EMP [14]	99.86	99.80

A. Precisión de la clasificación

Para evaluar la precisión de la clasificación, se consideraron las medidas estándar: precisión general medida como porcentaje de píxeles en los que se produce acierto en la clasificación (OA) y coeficiente kappa (κ).

Puede observarse, tal y como mostramos en la Fig. 4, que el valor de OA observado al clasificar una imagen usando perfiles de difusión aumenta con el número de componentes principales extraídas de la imagen para un paso de tiempo de proceso fijo. Sin embargo, el aumento del tamaño del perfil que supone este aumento en número de componentes acarrea un mayor uso de recursos en la posterior clasificación, por lo que finalmente se optó por seleccionar para los restantes experimentos 7 componentes principales ($N = 7$). Los valores de los parámetros $N = 7$, $C = 8$, $\sigma = 1$ y $ts = 65$ se utilizaron como un compromiso entre la precisión y el número de componentes del EADP. El tamaño total de cada ADP es, por lo tanto, 9 y el tamaño total de EADP es 63.

La tabla I compara la precisión de la clasificación en términos de OA y $\boldsymbol{\kappa}$ con otras dos propuestas diferentes para extraer información espectral-espacial



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

Fig. 3: Imágenes hiperespectrales utilizadas en los experimientos: (a) IndianP, (b) Salinas, (c) PaviaU, (d) PaviaC

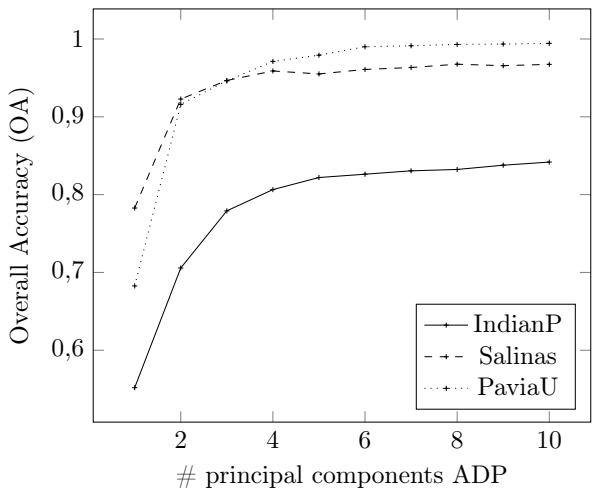


Fig. 4: OA variando el número de componentes principales extraídas para un ADP con ts=5.

que hemos desarrollado en trabajos anteriores. La primera es WT-EMP [15] y consiste en extraer características de la imagen original mediante una wavelet de modo similar a cómo en esta propuesta lo hacemos con PCA, construyendo luego un perfil morfológico extendido al que se le aplica reducción de ruido también mediante wavelets. En el caso de WTSS-EMP [14] el modelo anterior se refina añadiendo al resultado descrito una versión de la imagen original donde se ha eliminado ruido banda a banda. Los resultados muestran que los perfiles de difusión mejoran sustancialmente los resultados anteriores ya que extraen más eficientemente la información espacial.

B. Comparativa de rendimiento de CUDA

Con el fin de evaluar el rendimiento de la implementación en CUDA tomando como referencia una implementación eficiente en CPU, se desarrolló una implementación OpenMP del algoritmo que utiliza los 6 núcleos disponibles.

La Tabla II muestra una comparativa de rendimiento entre la CPU y las implementaciones de GPU. El proceso de difusión se divide en etapas y se muestra el tiempo total agregado (en milisegundos) para todas las iteraciones en cada etapa, así como la aceleración del código CUDA sobre el OpenMP.

Las etapas de *Setup* y *Cleanup* corresponden a la carga inicial del componente principal en la memoria del dispositivo y la transferencia de la imagen resultado del proceso de difusión a la memoria del host (CPU), respectivamente, que resultan ser las más costosas en la implementación en GPU. Las otras etapas hacen referencia a las líneas detalladas en el algoritmo 1.

Se puede observar que el aumento de rendimiento de la implementación en GPU es notable en las etapas de computación. La ganancia máxima se puede observar en el dataset IndianP, con un 10,47× de speedup sobre la implementación en OpenMP.

Podemos comprobar que la implementación de CUDA supera a la CPU en casi un orden de magnitud. Las aceleraciones logradas no muestran correlación con el tamaño de la imagen, con todas las escenas produciendo resultados similares. Es importante recalcar que, debido a los requisitos del algoritmo de difusión, las operaciones deben realizarse con aritmética de doble precisión, que está severamente limitada en las GPU NVIDIA de consumo.

V. Conclusiones

En este artículo se presenta una primera implementación en CUDA de perfiles de difusión anisotrópica (ADP) para extraer información espectral-espacial en imágenes hiperespectrales. Estos se crean aplicando múltiples instancias de difusión no lineal a componentes extraídas mediante análisis en componentes principales (PCA), que posteriormente se apilan para generar un perfil extendido (EADP). La implementación propuesta alcanza un rendimiento de hasta 10,47× para IndianP en comparación con la implementación OpenMP de referencia.

El reducido número de componentes del EADP permite un proceso de clasificación rápido y eficiente utilizando un algoritmo de clasificación supervisada como ELM, alcanzando valores de precisión de clasificación de hasta OA de 99,20 para PaviaC.

Agradecimientos

Este trabajo fue apoyado en parte por la Consellería de Educación, Universidade e Formación Profesional bajo las Subvenciones GRC2014/008, ED431C 2018/2019 y ED431G/08 y el Ministerio de Econo-

TABLA II: Tiempo medio de ejecución OpenMP y CUDA (en milisegundos) para la generación de un EADP de 63 componentes (a partir de 7 componentes principales obtenidas mediante PCA).

Etapa (líneas)	IndianP			Salinas			PaviaU		
	CPU	GPU	Speedup	CPU	GPU	Speedup	CPU	GPU	Speedup
Setup	0.001	0.021	0.005×	0.001	0.08	0.009×	0.001	0.137	0.007×
Matriz de difusividad	3.88	0.33	11.79×	9.73	0.59	16.55×	21.29	0.55	38.64×
Proceso FED	7.58	0.72	10.48×	27.15	3.26	8.32×	41.76	5.31	7.86×
Cleanup	0.001	0.029	0.05×	0.002	0.08	0.24×	0.002	0.137	0.02×
Total	11.46	1.09	10.47×	36.88	4.01	9.21×	63.06	6.14	10.27×

mía y Empresa, Gobierno de España bajo la subvención TIN2016-76373-P. Ambos están cofinanciados por el Fondo Europeo de Desarrollo Regional.

Referencias

[1] Gary A Shaw, "Spectral imaging for remote sensing," *Lincoln Laboratory Journal*, vol. 14, no. 1, pp. 3–28, 2003.

[2] 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*, vol. 46, no. 11, pp. 3804–3814, 2008.

[3] ST Acton and J Landis, "Multi-spectral anisotropic diffusion," *International Journal of Remote Sensing*, vol. 18, no. 13, pp. 2877–2886, 1997.

[4] Fardin Mirzapour and Hassan Ghassemian, "Hyperspectral image classification using profiles based on partial differential equations," in *Electrical Engineering (ICEE), 2015 23rd Iranian Conference on*. IEEE, 2015, pp. 288–292.

[5] Yi Wang, Ruiqing Niu, and Xin Yu, "Anisotropic diffusion for hyperspectral imagery enhancement," *IEEE Sensors Journal*, vol. 10, no. 3, pp. 469–477, 2010.

[6] Sven Grewenig, Joachim Weickert, and Andrés Bruhn, "From box filtering to fast explicit diffusion," in *Joint Pattern Recognition Symposium*. Springer, 2010, pp. 533–542.

[7] 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*, vol. 10, no. 11, pp. 4869–4878, 2017.

[8] Yan Ma, Lajiao Chen, Peng Liu, and Ke Lu, "Parallel programing templates for remote sensing image processing on GPU architectures: design and implementation," *Computing*, vol. 98, no. 1-2, pp. 7–33, 2016.

[9] Sergio Bernabe, Sergio Sanchez, 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*, vol. 6, no. 3, pp. 1386–1398, 2013.

[10] Pietro Perona and Jitendra Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[11] Pascal Gwosdek, Sven Grewenig, Andrés Bruhn, and Joachim Weickert, "Theoretical foundations of gaussian convolution by extended box filtering," in *International Conference on Scale Space and Variational Methods in Computer Vision*. Springer, 2011, pp. 447–458.

[12] Alberto S Garea, Dora B Heras, and Francisco Argüello, "GPU classification of remote-sensing images using kernel ELM and extended morphological profiles," *International Journal of Remote Sensing*, vol. 37, no. 24, pp. 5918–5935, 2016.

[13] 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*, vol. 8, no. 6, pp. 2884–2893, 2015.

[14] Pablo Quesada-Barriuso, Dora B Heras, and Francisco Argüello, "Exploring the impact of wavelet-based denoising in the classification of remote sensing hyperspectral images," in *Image and Signal Processing for Remote Sensing XXII*. International Society for Optics and Photonics, 2016, vol. 10004, p. 100040R.

[15] 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*, vol. 8, no. 6, pp. 2962–2970, 2015.