Nueva Implementación Paralela en GPUs para la Extración de Firmas Espectrales Puras con Información Espacial y Espectral

Luis Ignacio Jiménez, Javier Plaza , Antonio Plaza[1] , Sergio Sánchez [2] y Gabriel Martín [3]

Resumen—
La identificación de firmas espectrales puras o endmembers en imágenes hiperespectrales ha estado tradicionalmente sujeta exclusivamente al aprovechamiento de la información espectral. Recientemente, algunas técnicas tales como *spatial-spectral endmember extraction* (SSEE) incorporan tanto la información espectral como la información espacial disponible en la imagen. Debido a que la imágenes hiperespectrales contienen abundante información en ambos dominios, la integración de ambas fuentes de información normalmente supone un incremento de la complejidad y del coste computacional. En este trabajo, hemos desarrollado una nueva y eficiente implementación para el algoritmo SSEE utilizando tarjetas gráficas programables (GPUs), dispositivos compactos y de bajo costeque nos permiten obtener un factor de aceleración significativo si los recursos de la arquitectura son usados apropiadamente. La validaciń experimental se centra en la evaluación de los píxeles candidatos seleccionados por el algoritmo SSEE y en el rendimiento computacional de la implementación paralela. Dicha validación confirma que se puede aprovechar el algoritmo SSEE de una forma computacionalmente eficiente.

Palabras clave— **Análisis hiperespectral, spatial-spectral endmember extraction (SSEE), tarjetas gráficas programables (GPUs).**

I. Introducción

El desmezclado espectral [1] is una técnica muy importante para el aprovechamiento de las imágenes hiperespectrales en teledetección. En los últimos años, muchos métodos han sido desarrollados con el objetivo de identificar las firmas espectralmente puras que constituyen la imagen (llamadas endmembers) y su correspondiente fracción de abundacia a nivel de píxel [2]. Un problema recurrente es cómo identificar automáticamente los endmembers que son representativos de la información espacial y espectral contenida en la imagen. Por ejemplo, normalmente es complicado obtener píxeles espacialmente representativos, teniendo en cuenta que normalmente los algoritmos de identificación de endmembers se centran únicamente en la información espectral, lo que conlleva sensibilidad al ruido, a los valores atípicos y a endmembers anómalos [3].

Para abordar este asunto, varias estrategias han sido propuestas con la intención de guiar la identificación de endmembers hacia áreas espacialmente homogéneas, de las que se espera contengan las firmas espectrales más puras en la imagen [4], [5], [6]. Con este objetivo, han sido varios los métodos espaciales/espectrales desarrollados para la identificación de endmembers en datos hiperespectrales.

Uno de los primeros algoritmos diseñados con este propósito en la literatura fue *automatic morphological endmember extraction* (AMEE) [4], el cual usa las operaciones morfológicas de erosión y dilatación para discriminar los endmembers que son suficientemente puros, espectralmente hablando, cubriendo un área más amplia en términos espaciales. Otro método comúnmente utilizado es el *spatial-spectral endmember extraction* (SSEE) [5], que utiliza restricciones espaciales para mejorar el contraste espectral relativo entre endmembers con información espectral mínima y única mejorando así el potencial de estos elementos potencialmente importantes. Con el SSEE, la información espacial de la imagen es utilizada para incrementar el contraste espectral entre endmembers espectralmente similares pero espacialmente indendientes.

Por último, otros algoritmos de preprocesado espacial han sido usados para la identificación de endmembers [7], [8], [9]. Estos métodos fueron desarrollados con la idea de ser combinados con otros algoritmos basados en información espectral para la extracción de endmembers. *Spatial preprocessing* (SPP) [7] es un algoritmo que introduce la información espacial en el proceso de extracción de los endmembers, de tal manera que se puede acoplar con los métodos clásicos de identificación de endmembers [10]. La idea principal de este preprocesado es la de estimar, para cada píxel de la imagen de entrada, un valor escalar relativo a la similaridad espacial entre dicho píxel y su vecindario, definido por una ventana espacial alrededor del píxel. Una extensión de este concepto fue presentada en [8], donde el uso de vecindarios espaciales fijos en el SPP fue sustituido por la incorporación de regiones pensadas para una mejor caracterización del contexto espacial. Sin embargo, el RBSPP depende enormemente del algoritmo de crecimiento de regiones que realiza antes que el procedimiento haciédolo especialmente sensible a la técnica seleccionada. Otra técnica más reciente es el *spatial and spectral preprocessing* (SSPP) desarrollado en [9], donde se integran ambos tipos de información, espacial y espectral, a nivel de preprocesado. El proceso se divide en cuatro etapas: 1) filtrado gaussiano multiescalar, donde la imagen orig-

[1]Dpto. de Arquitectura de Tecnología de los Computadores y las Comunicaciones, Univ. de Extremadura, Cáceres, España. e-mail: {luijimenez, jplaza, aplaza}@unex.es
[2]Masdar Institute of Science and Technology, Abu Dhabi, UAE. e-mail: smartinez@masdar.ac.ae.
[3]Instituto de Telecomunicações, Lisboa, Portugal. e-mail: salmeron@per.edu.

inal es filtrada para remarcar el contexto espacial, 2) el cálculo de la homogeneidad espacial, donde la imagen filtrada es procesada para obtener un índice de homogeneidad espacial para píxel de la imagen original, 3) el cálculo de un índice de pureza espectral, que en primer lugar reduce la imagen hiperespectral usando el análisis de componentes principales (PCA) [11] y crea un índice de pureza espectral usando un método parecido al adoptado por el algoritmo *pixel purity index* (PPI) [12], y 4) un proceso de fusión de ambos tipos de información donde los únicos píxeles seleccionados son aquellos que son al mismo tiempo espectralmente puros y espacialmente homogéneos.

Todas las técnicas anteriormente menionadas para la identificación de endmembers basadas en información espacial y espectral se caracterizan por tener un coste computacional alto, debido a la necesidad de procesar ambos tipos de información contenidos en las imágenes hiperespectrales. Sin embargo, debido a que los cálculos son similares, muchas de estas técnicas han sido eficientemente implementadas utilizando tarjetas de procesamiento gráfico o GPUs. Por ejemplo el algoritmo SPP fue implementado para GPUs en [13]. El SSPP también ha sido implementado para GPUs en [14], mientras que el RBSPP también es subceptible de ser paralelizado (con la principal dificultad del proceso de segmentación que generalmente resulta en una distribución irregular in versiones paralelas). El algoritmo AMEE ha sido tambien implementado para GPUs en [15]. Aunque, el SSEE (un método bastante representativo y exitoso para la extracción de endmembers utilizando información espacial y espectral) aÃºn no ha sido implementado sobre aceleradores gráficos.

En este trabajo de ha desarrollado una implementación del algoritmo SSEE [5] para GPUs. En este método, la imagen hiperespectral es procesada con la intención de buscar un conjunto de firmas que se consideran candidatas a ser endmembers de entre las cuales que procederá a la extracción de la firmas espectrales puras finales. En consecuencia, el SSEE se puede considerar un algoritmo de preprocesado espacial. Reduciendo el número de cadidatos a ser endmembers se puede reducir considerablemente el tiempo computacional el posterior algoritmo de extracción de endmembers, por lo que el SSEE puede ser combinado con algoritmos clásicos de extracción de endmembers como si fuera otro algoritmo de preprocesado tal y como el SPP, el SSPP o el RBSPP. Aunque el algoritmo SSEE se caracteriza por una complejidad y unos tiempos de computación muy elevados, se han implementado una serie de optimizaciones que permiten una ejecución eficiente del método en GPUs utilizando *Arquitectura Unificada de Dispositivos de Cómputo* (CUDA)[1] de NVidia. La implementación propuesta ha sido probada en dos arquitecturas NVidia distintas usando imágenes reales. Se han realizado comparaciones de los nuevos métodos propuestos contra las implementaciones paralelas disponibles de los

[1]https://developer.nvidia.com/cuda-zone

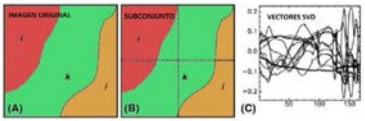

Fig. 1. La primera etapa del algoritmo SSEE (cálculo de autovectores). La imagen (A) contiene tres componentes diferentes (i, j, y k) y es dividida en cuatro subconjuntos (B) del mismo tamaño y disjuntos. La SVD es utilizada para obtener los autovectores (C).

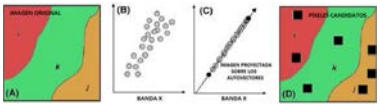

Fig. 2. El segundo paso del algoritmo SSEE (la proyección de los datos). La imagen original (A) representada en el espacio caracterizado como se indica en (B), es proyectada sobre los autovectores obtenidos en la Figura 1(C), identificando los valores de las proyecciones máxima y mínima para seleccionar los píxeles candidatos (D).

algoritmos SPP y SSPP, revelando que estos algoritmos pueden ser eficientemente aprovechados en GPUs al incluir la información espacial en el proceso de extracción de endmembers.

El resto del artículo se organiza como se detalla a continuación. La Sección II describe el algoritmo SSEE en general, así como las distintas versiones optimizadas. La Sección III presenta la implementación GPU del algoritmo SSEE. Los experimentos llevados a cabo para evaluar la precisión de las firmas extraídas y el rendimiento conseguido por las implementaciones paralelas propuestas se especifican en la Sección IV. Por último, la Sección V finaliza este trabajo con una puntualizaciones y posibles líneas de investigación futuras.

II. Spatial-Spectral Endmember Extraction (SSEE) e Implementaciones

El algoritmo SSEE original puede resumirse en los siguiente pasos [5]:

1. En primer lugar, la imagen hiperespectral original es dividida en partes más pequeñas y se utiliza la descomposión de evalores singulares (SVD) para obtener los autovectores que describen la varianza espectral de cada uno de subconjuntos. Esta etapa del algoritmo es la que más tiempo consume. La implementación original del algoritmo SSEE define que los subconjuntos de la imagen no pueden solaparse entre ellos (ver la Figura 1).
2. Toda la imagen es proyectada sobre los mencionados autovectores obteneiendo los valores de la proyección máxima y mínima que determinan los píxeles candidatos iniciales (ver la Figura 2).
3. En la tercera etapa (la expansión y promedio de los píxeles candidatos), el algoritmo SSEE

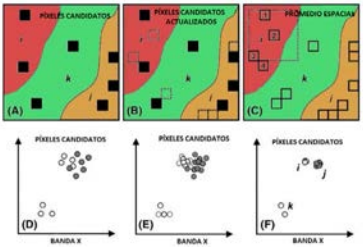

Fig. 3. Tercera etapa del algoritmo SSEE. Después de seleccionar los píxeles candidatos iniciales distribuidos espacialmente con respecto a las clases i, j y k (A), el proceso de expansión comienza alrededor de cada píxel que es espacialmente cercano y espectralmente similar a los seleccionados anteriormente en (B). Un proceso de promedio espectral (utilizando una ventana fija alrededor de cada pixel candidato) es realizado en (C). Por ejemplo en (D), los píxeles candidatos no están promediados. El promediado espacial-espectral comienza en (E) para cada clase. Esta etapa continua por un número de iteraciones determinado por el usuario, agrupando los pixeles candidatos en clases espectrales separadas (F).

utiliza restricciones espaciales para combinar y promediar los píxeles candidatos que son espectralmente similares. En otras palabras, del conjunto de píxeles candidatos obtenidos en el paso anterior es extendido en base a la similaridad espectral entre candidatos dentro de una ventana alrededor de cada píxel, seguido de un promediado en otra ventana del mismo tamaño que la primera. Este proceso separa los endmembers por clases en el espacio espectral, como se lustra en la Figura 3. En este punto, el algoritmo SSEE ha obtenido un conjunto de píxeles candidatos que contendrá a los endmembers finales. En consecuencia, hasta este punto el algoritmo SSEE produce unos resultados del mismo tipo que otros algoritmos de preprocesado tales como el SPP o el SSPP.

4. En el último paso el algoritmo SSEE realiza un listado de los píxeles candidatos basado en la distancia de cada uno de ellos respecto al primero de la lista. A partir de ahí, como se menciona en el artículo original [5], se puede usar un procdimiento manual o técnicas de extracción de endmembers basadas en información espectral clásicas como *orthogonal subspace projection* (OSP) [16].

El algoritmo SSEE descrito anteriormente está caracterizado por un alto coste computacional. La mayoría del tiempo de computación (alrededor de un 99%) se gasta en la selección de candidatos, debido principalmente al coste computacional de la operación SVD y las posteriores proyecciones usando la imagen completa. Antes de explorar la propuesta de implementación para GPUs, se va a describir la optimización principal del método usada para reducir su complejidad. Esta modificación está centrada en el método de obtención de los autovectores.

La SVD no es única técnica de proyección que se puede usar para determinar los autovectores necesarios para el algoritmo SSE. En este trabajo se estudia la posibilidad de usar una implementación rápida de la PCA [17] con esta intención. Esta técnica tiene la ventaja de reducir la complejidad del cálculo de la proyecciones. Además, la PCA es altamente paralelizable con muchas implementaciones disponibles.

III. Implementación GPU

La implementación GPU para el algoritmo SSEE ha sido desarrollada usando el lenguaje de programación CUDA de NVidia, con algunas llamadas a funciones disponibles en las librerías cuBLAS[2] y BLAS/LAPACK[3]. A continuación se describe la arquitectura general de las GPUs y la implementación específica de nuestra propuesta.

A. Arquitectura GPU

Las GPUs se pueden considerar en términos de un modelo de flujo de datos. La Figura 4 nos muestra la arquitectura de una GPU, la cual puede ser considereada como un conjunto de multiprocesadores (MPs). Cada multiprocesador es definido por un módulo SIMD (*single instruction multiple data*). Cada procesador tiene acceso a la memoria local compartida y también a las memorias caché en el multiprocesador, mientras que el multiprocesador tiene acceso a la memoria global de la GPU. Los algoritmos están constituidos por funciones, llamadas *kernels* que operan con flujos de datos completos y son ejecutados por el multiprocesador, usando uno o más flujos como entradas y produciendo uno o más flujos como salidas. De este modo, el paralelismo a nivel de datos está expuesto al hardware, y los kernels pueden ser ejecutados concurrentemente sin ningún tipo de sincronización. Los kernels puede realizar un tipo de procesamiento por lotes dispuestos en forma de malla (*grid*) de bloques (*blocks*) (ver la Figura 5), donde cada bloque se compone de varios hilos (*threads*) que comparten los datos a través de la memoria compartida local y sincronizan su ejecución mediante coordenadas de acceso a la memoria. En consecuencia, hay diferentes niveles de memoria en la GPU para hilo, bloque y malla (ver la Figura 6). También hay un número máximo de hilos que un bloque puede contener pero el número de hilos que se ejecutan concurrentemente puede ser mayor (bloques ejecutados por el mismo kernel pueden ser manejados concurrentemente, a expensas de reducir la cooperación entre hilos ya que pertenecen a distintos bloques de la misma malla y no pueden sincronizar entre ellos).

B. Implementación GPU del Algoritmo SSEE

La implementación GPU para las distintas etapas del SSEE pueden resumirse en:

[2]https://developer.nvidia.com/cublas
[3]http://www.netlib.org/lapack/

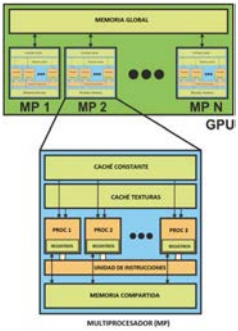

Fig. 4. Esquema general de una aquitectura GPU, vista como un conjunto de multiprocesadores (MPs).

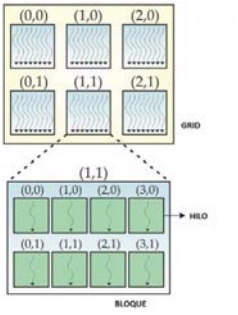

Fig. 5. Procesamiento por lotes en la GPU: grids de bloques de hilos, donde cada bloque está compuesto por un grupo de hilos.

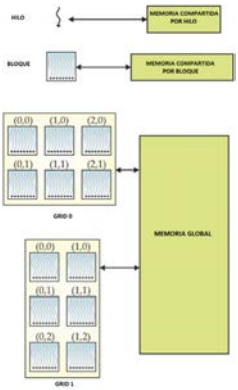

Fig. 6. Diferentes niveles de memoria en la GPU en relación con los conceptos de hilo, bloque y grid.

1. *Cálculo de autovectores.* Una parte significativa del tiempo de ejecución del algoritmo SSEE es utilizado para el cálculo de los autovectores. Este paso puede realizarse usando la SVD o la PCA. La SVD ha sido implementada llamando a la función *dgesvd* disponible en la librería BLAS/LAPACK, la cual ha sido evaluada experimentalmente para ser computacionalmente muy rápida. Para la implementacón GPU de la PCA, se realizan los siguientes pasos. Primero, un kernel normaliza el subconjunto de la imagen restando el píxel medio de dicho subconjunto. Este kernel, llamado *avgXCUDA*, establece una malla unidimensional cuyo tamaño es igual al número de bandas de la imagen y el tamaño de bloque es igual al número de hilos disponible en la GPU. A continuación se llama a la fución de cuBLAS *cublasDgemm* que realiza la multiplicación del subconjunto de la imagen por su transpuesta. Finalmente la transformación SVD es calculada en la CPU, ya que experimentalmente se ha comprobado que la paralelización no es necesaria para obtener unos resultados mejores a nivel de rendimiento.
2. *Proyección de datos.* Una vez los autovectores han sido calculados y tranferidos a la memoria de la GPU, se utiliza la función *cublasDgemm* de nuevo para realizar la proyección de la imagen. Este paso es muy imprtante en términos de coste computacional mayormente por el incremento de complejidad al aumentar el tamaño de la ventana utilizada. Para este paso, se usa un kernel llamado *maxminProjection* que ha sido diseñado para seleccionar los valores de las proyecciones máxima y mínima de cada banda de la imagen, utilizando una operación de reducción correspondiente al número de bandas que representan el porcentaje del total definido por un valor umbral, *svd_th*, introducido como parámetro. Aquí el número de bloques se establece como el número de bandas y el número de hilos se fija en el máximo permitido por el dispositivo.
3. *Expansión y promedio de los píxeles candidatos.* Para realizar este paso lo primero es calcular la norma euclidea de cada píxel de la imagen utilizando un kernel llamado *euclideanNorm*. El número de hilos para este kernel se fija al máximo permitido por la GPU, y el número de bloques se establece como la relación entre el número de píxels y el tamaño del bloque más uno. Un segundo kernel llamado *expandCandidatePixels* es desarrollado para realizar la expansión del conjunto de candidatos dentro del rango que marca el tamaño de ventana *ws*. Finalmente otro kernel llamado *averageStep* calcula el promedio espectral de los cadidatos dentro del mismo rango definido por la ventana alrededor de cada uno de ellos. Ambos kernels definen una malla bidimensional en la cual el número de filas y columnas son definidos por el

TABLA I

PROMEDIO DE DISTANCIA ESPECTRAL (EN GRADOS) ENTRE EL CONJUNTO DE PÍXELS OBTENIDOS POR LOS DISTINTOS MÉTODOS (SSEE, SPP Y SSPP) Y EL CONJUNTO DE REFERENCIA DE FIRMAS SELECCIONADAS DE LA LIBRERÍA USGS EN LA IMAGEN AVIRIS CUPRITE. EN EL CASO DE LOS ALGORITMOS SSEE Y SSPP VARIOS TAMAÑOS DE VENTANA HAN SIDO CONSIDERADOS. LOS RESULTADOS CORRESPONDEN A LOS OBTENIDOS TRAS 100 EJECUCIONES DEL ALGORITMO DE EMPAREJAMIENTO. LOS MEJORES RESULTADOS (ÁNGULOS BAJOS) SE HAN RESALTADO EN NEGRITA.

	SSEE-SVD				**SSEE-PCA**				**SPP**				**SSPP**
Ventana	35x35	50x50	70x70	175x175	35x35	50x50	70x70	175x175	3x3	5x5	7x7	9x9	
Candidatos	2416	1795	1310	403	204	140	77	28	122500	122500	122500	122500	3653
Alunite GDS84 Na03	7,467	7,088	7,472	8,756	**5,735**	5,965	6,361	9,241	5,819	5,942	6,041	6,041	6,863
Alunite GDS83 Na63	7,660	7,676	8,165	9,398	6,886	**6,660**	7,796	10,204	6,876	6,955	7,033	7,035	9,449
Alunite GDS82 Na82	5,089	5,077	5,572	6,871	4,669	**4,107**	5,223	8,165	4,302	4,395	4,491	4,474	6,958
Alunite AL706 Na	5,713	5,847	6,644	7,848	**4,268**	5,229	6,286	8,328	4,794	4,820	4,852	4,844	7,320
Alunite HS295.3B	15,044	14,865	15,465	17,039	11,398	11,202	11,724	17,443	12,196	12,486	12,725	12,737	**11,002**
Alunite SUSTDA-20	6,153	6,195	6,748	7,916	5,427	**5,352**	6,385	8,810	5,434	5,501	5,562	5,562	7,763
Buddingtonite GDS85 D-206	3,793	3,932	3,904	4,854	3,924	3,918	4,049	5,553	3,781	3,778	**3,777**	**3,777**	3,863
Calcite WS272	7,237	7,440	8,341	9,543	**5,052**	7,035	7,948	10,066	5,252	5,307	5,383	5,380	8,647
Calcite HS48.3B	7,761	7,976	8,863	10,082	**5,503**	7,644	8,563	10,787	5,559	5,624	5,699	5,700	8,703
Chalcedony CU91-6A	4,242	4,425	5,194	6,275	**2,918**	4,666	5,019	7,139	2,993	3,035	3,061	3,057	5,717
Chlorite HS179.3B	15,833	16,155	17,932	22,978	15,545	15,192	17,528	22,276	14,796	15,001	15,129	15,146	**13,868**
Chlorite SMR-13.a 104-150	20,676	21,056	22,675	27,574	20,506	20,018	22,767	27,151	19,695	19,893	20,021	20,018	**18,821**
Dickite NMNH106242	7,221	7,272	7,769	8,819	**6,563**	6,664	7,490	9,592	6,618	6,652	6,704	6,699	8,664
Halloysite NMNH106236	12,106	11,733	12,097	13,587	**9,376**	9,548	9,823	13,546	10,026	10,237	10,388	10,385	10,816
Jarosite GDS99 K-y 200C	8,637	8,607	8,827	11,015	8,531	8,845	9,838	10,630	8,357	8,380	8,392	8,392	**8,245**
Kaolinite KGa-1 (wxyl)	5,296	5,455	6,246	7,460	**3,755**	5,082	5,928	8,138	4,199	4,249	4,285	4,297	6,848
Kaolinite KGa-2 (pxyl)	4,393	4,573	5,360	6,489	**3,052**	4,466	5,110	7,005	3,386	3,435	3,455	3,461	5,611
Kaolin/Smect KLF506 95%K	3,835	3,619	4,249	6,946	4,137	5,363	5,885	6,984	**2,799**	2,853	2,867	2,885	3,080
Montmorillonite SWy-1	5,922	6,113	6,937	8,045	**4,234**	6,006	6,681	8,812	4,507	4,550	4,594	4,597	7,232
Montmorillonite SAz-1	8,733	8,688	9,016	10,159	8,348	**7,867**	8,800	11,163	7,987	8,064	8,142	8,150	9,807
Muscovite GDS107	5,191	5,084	5,682	8,118	5,571	6,314	6,628	8,312	4,092	4,055	4,133	4,142	**4,047**
Muscovite HS146.3B	8,205	8,003	8,953	12,232	8,090	9,335	10,274	12,119	7,716	7,843	7,902	7,896	**7,415**
Muscovite HS24.3	6,213	6,026	7,039	10,765	6,138	7,422	8,304	10,275	5,788	5,818	5,833	5,831	**5,365**
Nontronite GDS41	14,366	14,833	16,429	21,576	14,205	13,372	16,609	21,054	13,365	13,583	13,741	13,727	**12,362**
Pyrophyllite PYS1A fine g	6,674	6,869	7,675	8,862	**4,773**	6,616	7,338	9,178	4,892	4,984	5,031	5,041	7,905
Promedio	8,138	8,184	8,930	10,928	7,144	7,756	8,734	11,279	**7,009**	7,098	7,170	7,171	8,255
S(%)	89,014	89,267	89,465	79,836	91,917	**91,943**	89,969	75,989	——	——	——	——	88,509

número de filas y columnas de la imagen original respectivamente, y el máximo número de hilos disponibles se fija como tamaño de bloque.

4. *listado*. El último paso es listar los píxeles candidatos en función de la distancia que los separa. En este trabajo, se usa un kernel llamado *BitonicSort* para calcular reordenar mediante un algoritmo de ordenamiento bitónico para GPUs. número de hilos se ha establecido empíricamente a 256 y el número de bloques se calcula según la siguiente expresión:

$$n\_bloques = \exp(\log_2\lfloor\frac{n\_pixels}{n\_hilos}\rfloor),$$

donde *n_bloques*, *n_pixels* and *n_hilos* denotan el número de bloques, número de píxeles y número de hilos respectivamente. Se quiere enfatizar que este último paso es puramente opcional y puede ser reemplazado por un método de extracción de endmembers. También se quiere remarcar que esta última etapa significa un tiempo despreciable con respecto al total, incluso cuando el conjunto de candidatos resulta muy grande.

IV. RESULTADOS EXPERIMENTALES

Los experimentos se han realizado utilizando un subconjunto de la imagen obtenida por *Airborne Visible Infra-Red Imaging Spectrometer* (AVIRIS), operado por *NASA's Jet Propulsion Laboratory*, sobre la región minera de Cuprite en Nevada en el verano de 1997 (disponible online[4]). La porción utilizada en los experimentos (ver Figura 7) corresponde a un subconjunto de 350×350 píxeles, que comprende 188 bandas espectrales en el rango entre los 400 y los 2500 nm y un total de 50MB de tamaño. Los minerales presentes en esta images se conocen muy bien y las firmas espectrales de referencia estan disponibles en una librería (*United States Geological Survey (USGS) library* [5]) que es usada en este trabajo para una evaluación de la precisión de los algoritmos.

Dos tipos de experimentos se llevan a cabo en este trabajo para evaluar la calidad de los píxeles candidatos extarídos y el rendimiento de la implementación en paralelo para GPUs del algoritmo SSEE. Primero, se analizarán la calidad de los candidatos seleccionados por distintos métodos (no solo

[4]http://aviris.jpl.nasa.gov
[5]http://speclab.cr.usgs.gov

TABLA II
ÁNGULO ESPECTRAL (EN GRADOS) ENTRE EL CONJUNTO DE PÍXELES OBTENIDOS POR LOS DIFERENTES MÉTODOS DE PREPROCESADO ESPACIAL (SSEE, SPP Y SSPP) Y EL CONJUNTO DE FIRMAS DE REFERECIA SELECIONADO DE LA LIBRERÍA USGS PARA LA IMAGEN AVIRIS CUPRITE. LOS RESULTADOS MOSTRADOS COREESPONDEN A LOS EMPAREJAMIENTOS MÍNIMOS DE ÁNGULO ESPECTRAL. LOS MEJORES RESULTADOS (ÁNGULOS BAJOS) SE HAN RESALTADO EN NEGRITA.

	SSEE-SVD				**SSEE-PCA**				**SPP**				**SSPP**
Ventana	35x35	50x50	70x70	175x175	35x35	50x50	70x70	175x175	3x3	5x5	7x7	9x9	
Candidatos	2416	1795	1310	403	204	140	77	28	122500	122500	122500	122500	3653
Alunite GDS84 Na03	7,467	7,088	7,472	8,761	5,735	5,965	**5,500**	6,444	5,819	5,942	6,041	6,041	6,804
Alunite GDS83 Na63	7,662	7,678	8,207	9,405	6,886	**6,663**	8,227	9,203	6,939	7,015	7,070	7,070	9,518
Alunite GDS82 Na82	5,087	5,075	5,454	6,861	4,669	**4,098**	5,014	6,545	4,224	4,285	4,365	4,365	6,905
Alunite AL706 Na	5,720	5,847	6,666	7,842	**4,268**	5,228	6,031	7,392	4,836	4,877	4,911	4,911	7,270
Alunite HS295.3B	15,044	14,865	15,465	17,049	11,398	11,202	11,084	21,917	12,196	12,660	12,834	12,834	**11,002**
Alunite SUSTDA-20	6,153	6,203	6,736	7,912	5,427	**5,359**	6,202	7,643	5,424	5,526	5,561	5,561	7,753
Buddingtonite GDS85 D-206	3,793	3,932	3,904	4,750	3,924	3,918	4,045	4,767	3,781	3,778	**3,777**	**3,777**	3,863
Calcite WS272	7,258	7,446	8,413	9,549	**5,053**	7,060	8,314	8,912	5,323	5,381	5,432	5,432	8,772
Calcite HS48.3B	7,785	7,983	8,970	10,090	**5,503**	7,665	9,074	9,569	5,678	5,753	5,827	5,827	8,900
Chalcedony CU91-6A	4,140	4,415	4,808	6,271	**2,918**	4,644	4,869	5,639	2,959	2,988	3,021	3,021	5,669
Chlorite HS179.3B	15,896	16,562	17,964	22,946	15,547	15,192	20,597	24,182	14,805	14,951	15,134	15,134	**14,126**
Chlorite SMR-13.a 104-150	20,841	21,470	22,762	27,658	20,653	20,018	27,473	28,595	19,755	20,029	20,126	20,126	**19,041**
Dickite NMNH106242	7,221	7,277	7,804	8,821	**6,563**	6,670	7,420	8,616	6,659	6,689	6,787	6,787	8,761
Halloysite NMNH106236	12,106	11,733	12,097	13,595	**9,376**	9,548	9,408	19,196	10,026	10,197	10,360	10,360	10,816
Jarosite GDS99 K-y 200C	8,589	8,607	8,796	10,992	8,959	8,797	9,806	10,405	8,357	8,380	8,392	8,392	**8,245**
Kaolinite KGa-1 (wxyl)	5,299	5,446	6,230	7,455	**3,755**	5,066	5,736	6,949	4,192	4,277	4,308	4,308	6,639
Kaolinite KGa-2 (pxyl)	4,390	4,562	5,307	6,483	**3,052**	4,440	4,950	6,079	3,335	3,346	3,363	3,363	5,461
Kaolin/Smect KLF506 95%K	3,821	3,619	4,200	6,701	4,137	5,363	5,509	5,192	**2,637**	2,721	2,763	2,763	3,080
Montmorillonite SWy-1	5,941	6,115	6,995	8,041	**4,234**	6,029	6,485	7,463	4,482	4,601	4,655	4,655	7,253
Montmorillonite SAz-1	8,733	8,688	9,045	10,164	8,348	**7,867**	9,251	14,627	8,114	8,237	8,178	8,178	9,911
Muscovite GDS107	5,214	5,084	5,782	8,258	5,571	6,314	6,623	7,026	**3,935**	4,024	4,076	4,076	4,047
Muscovite HS146.3B	8,206	7,984	8,919	12,373	8,073	10,066	10,412	11,964	7,867	7,901	7,946	7,946	**7,423**
Muscovite HS24.3	6,218	5,868	6,749	10,807	6,058	7,006	8,406	10,273	5,788	5,818	5,830	5,830	**5,354**
Nontronite GDS41	14,434	14,568	16,429	21,561	13,621	13,372	18,601	22,426	13,324	13,502	13,622	13,622	**12,093**
Pyrophyllite PYS1A fine g	6,685	6,871	7,781	8,865	**4,773**	6,640	7,315	8,294	4,872	4,942	5,021	5,021	7,987
Promedio	8,148	8,199	8,918	10,928	7,140	7,768	9,054	11,173	**7,013**	7,113	7,176	7,176	8,268
S(%)	89,014	89,267	89,465	79,836	**91,917**	89,943	89,969	81,989	——	——	——	——	88,509

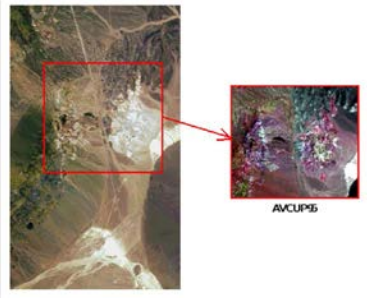

Fig. 7. Composición en falso color de la imagen hiperespectral recogida por el sensor AVIRIS sobre la región de Cuprite en Nevada.

las implementaciones del SSEE, sino también por parte del SPP y el SSPP) en comparación a un conjunto de firmas espectrales de la librería USGS, utilizando el angulo espectral (SAD) como métrica cuantitativa. Se recuerda que el algoritmo SSEE puede ser visto como un preprocesado espacial, muy parecido al SPP y al SSPP. En consecuencia, una comparación entre estos métodos es realizada para medir la calidad de los candidatos seleccionados por cada uno de ellos. Con este propósito se han seleccionado 25 firmas de minerales de la librería USGS, que se muestran en la FIgura 8. A continuación, se analizará el rendimiento de las versiones paralelas del algoritmo utilizando dos arquitecturas differentes.

A. Calidad de los píxeles candidatos

Par evaluar la calidad de los píxeles candidatos seleccionados por las implementaciones paralelas en relación al conjunto de firmas seleccionado de la librerá USGS, se consideran dos algoritmos de emparejamiento distintos [9]. El primero de ellos, obtiene el promedio de los mejores emparejamientos entre los dos conjuntos reordenados aleatoriamente en cada iteración de un total de 100, que se pueden ver en la Tabla I. Esta tabla incluye los resultados tanto de las dos versiones del algoritmo SSEE como de los algoritmos SPP y SSPP. El segundo algoritmo empareja las firmas espectrales cogiendo el valor mínimo de angulo espectral entre todas las posible combinaciones (ver la Tabla II). En ambos casos las 25 firmas de minerales seleccionadas son usadas como conjunto de referencia para ambos algoritmos. Con ánimo de es-

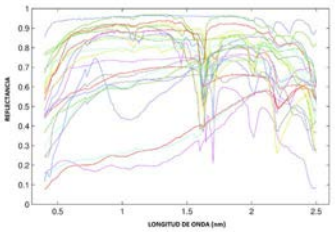

Fig. 8. Conjunto de 25 firmas espectrales seleccionadas para los experimentos de la librería USGS: Alunite GDS84 Na03, Alunite GDS83 Na63, Alunite GDS82 Na82, Alunite AL706 Na, Alunite HS295.3B, Alunite SUSTDA-20, Buddingtonite GDS85 D-206, Calcite WS272, Calcite HS48.3B, Chalcedony CU91-6A, Chlorite HS179.3B, Chlorite SMR-13.a 104-150, Dickite NMNH106242, Halloysite NMNH106236, Jarosite GDS99 K-y 200C, Kaolinite KGa-1 (wxyl), Kaolinite KGa-2 (pxyl), Kaolin/Smect KLF506 95%K, Montmorillonite SWy-1, Montmorillonite SAz-1, Muscovite GDS107, Muscovite HS146.3B, Muscovite HS24.3, Nontronite GDS41, Pyrophyllite PYS1A fine g.

tablecer una comparación justa se requiere comparar un número de de candidatos que coincidan con las firmas de referencia. Aunque surge el problema de como decidir si un un píxel candidato está bien emparejado con su respectiva firma de referencia. En nuestros experimentos se considera que una firma de referencia es encontrada cuando su valor de ángulo espectral esta por debajo de 10° (donde el peor caso entre dos firmas en un valor de 90°). En base a lo comentado anteriormente, se utiliza la siguiente métrica para mostrar el porcentaje de acierto S:

$$S(\%) = \frac{\frac{m}{P} + [1 - \frac{n}{N}]}{2} * 100,$$

donde n es el número de candidatos extraídos por el método, N es el total de píxeles de la imagen, m es el número de firmas satisfactoriamente emparejadas y P es el número de firmas de referencia. Las Tablas I y II incluyen los resultados de esta métrica. En el caso del algoritmo SPP, el número de píxeles candidatos es igual al tamaño de la imagen ya que este algoritmo no realiza ninguan reducción en el número de candidatos, pero tanto el SSPP como las versiones del SSEE si reducen esta cantidad.

Como se muestra en las Tablas I y II, varias de las 25 firmas de referencia no están bien emparejadas por algunos de los métodos, como por ejemplo *Alunite HS295.3B* o *Chlorite SMR-13.a 104-150*. Esto se debe a la dificultad de garantizar que esas mismas firmas se encuentren presentes en la imagen original, teniendo en cuenta que algunos de los minerales que están representados en la imagen pueden no estar en el conjunto de 25 firmas de referencia. Muchos de los algoritmos son capaces de proporcionar una serie de píxeles candidatos que pueden ser muy similares, espectralmente hablando, a las firmas de referencia de la librería USGS, incluso considerando que el umbral superior es de 10°. En general podemos concluir en base a las Tablas I y II que el algoritmo SSEE proporciona buenos resultados en la mayoría de los casos, siendo levemente mejores en la versión que usa la PCA. Considerando todos los algoritmos, los mejores resultados son obtenidos por el algoritmo SPP pero se ha de tener en cuenta que el SSEE cuando usa la PCA obtiene un conjunto mucho más reducido de píxeles candidatos, lo cual tendrá un fuerte impacto en el rendimiento computacional del que se habla en el siguiente apartado.

B. Computational Performance

Antes de presentar los resultados de rendimiento obtenidos, se quiere señalar que todas las implementaciones GPU de los algoritmos SSEE, SPP y SSPP mencionadas en esta sección obtienen exactamente los mismos resultados tanto en serie como en paralelo. Con intención de evaluar el rendimiento computacional se han realizado los experimentos en dos arquitecturas diferentes:

- Arquitectura 1. Un ordenador de sobremesa (Intel core i7 920 CPU a 2.67 GHz y 6 GB de RAM) con una tarjeta NVidia GTX 580, con 512 núcleos operando a 1.54 GHz (de ahora en adelante Arquitectura 1).
- Arquitectura 2. Un clúster[6], con nodos Nvidia TESLA S2070 (2 M2075 por nodo) y un núcleo Intel Xeon CPU E5645 a 240GHz y 24GB DDR3, divididos en 12 módulos de 2GB cada uno (de ahora en adelante Arquitectura 2).

En ambos casos las versiones serie de los algoritmos fueron ejecutadas en unco de los núcleos disponibles, y los tiempos paralelos han sido medidos en función de los recursos de cada arquitectura. La aceleración media ha sido calculada entre cada par CPU/GPU tras 10 ejecuciones.

La Figura 9 muestra el rendimientos de la implementaciones paralelas del algoritmo SSEE considerando las dos arquitecturas propuestas (para los resultados de los algoritmos SPP y SSPP, nos remitiremos a los trabajos [13], [14] respectivamente donde se describen dichos métodos más en detalle). De la Figura 9, podemos concluir que la versión del algoritmo SSEE que utiliza la PCA en vez de la SVD exibe un coste computacional más bajo, mayormente debido a la reducción del volumen de datos sumado a un rango de paralelización mayor. La versión con la SVD está limitada en este caso debido a que su paralelización en GPUs es rentable únicamente con volúmenes de datos muy grandes [18]. Si se tienen en cuenta los resultados de precisión de la subsección IV-A, podemos decir que la versión PCAmejora claramente los resultados obtenidos por la versión SVD.

V. Conclusiones y Líneas Futuras

En este trabajo se ha presentado una nueva implementación del algoritmos *spatial-spectral endmem-*

[6]http://www.ceta-ciemat.es/

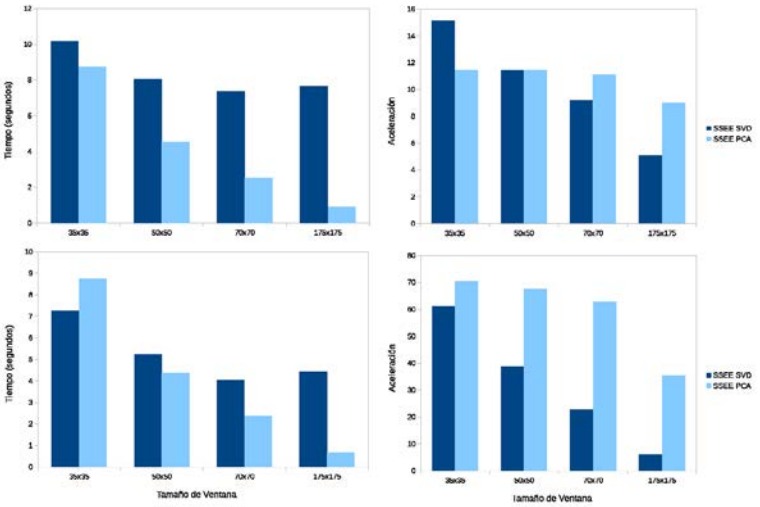

Fig. 9. Tiempos de ejecución (en segundos) y rendimiento (aceleración) para las implementaciones paralelas del algoritmo SSEE, considerandose distintos tamaños de ventana, tras procesar la imagen AVIRIS Cuprite en la Arquitectura 1 (fila superior) y la Arquitectura 2 (fila inferior). Los valores obtenidos son los promedios tras 10 ejecuciones.

ber extraction (SSEE), el cual se puede considerar como un preprocesado espacial capaz de seleccionar píxeles candidatos para una posterior extracción de endmembers. Diferentes implementaciones han sido consideradas, basándonos en el método de cálculo de los autovectores. La implementación GPU propuesta ha sido comparada con otros métodos de preprocesado espacial conocidos, tales como, el algoritmo SPP y el SSPP. Los resultados obtenidos sugieren que la implementación GPU del algoritmo SSEE proporciona unos resultados competitivos ante los algoritmos mencionados anteriormente, y permiten una reducción de los tiempos computacionales a la vez que se mantiene una alta calidad del conjunto de píxeles seleccionados como candidatos a ser endmembers. Para futuros trabajos nos centraremos en la inclusón de nuevos algoritmos de preprocesado espacial en la comparación y el desarrollo de nuevas estrategias de paralelización en el preprocesado usando GPUs y otra arquitecturas de computación de alto rendimiento tales como las *field programmable gate-arrays* (FPGAs).

VI. Agradecimientos

Este trabajo ha sido subvencionado por la Junta de Extremadura (a través del decreto 297/2014, ayudas para la realización de actividades de investigación y desarrollo tecnológico, de divulgación y de transferencia de conocimiento por los Grupos de Investigación de Extremadura, Ref. GR15005). Además, el presente trabajo ha sido llevado a cabo haciendo uso de la infraestructura de computación facilitada por el Centro Extremeño de Tecnologías Avanzadas (CETA-CIEMAT), financiado por el Fondo Europeo de Desarrollo Regional (FEDER). El CETA-CIEMAT pertenece al CIEMAT y al Gobierno de España.

Referencias

[1] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 44–57, 2002.

[2] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 5, no. 2, pp. 354–379, 2012.

[3] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 650–663, 2004.

[4] A. Plaza, P. Martinez, R. Pérez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 40, no. 9, pp. 2025–2041, 2002.

[5] D. M. Rogge, B. Rivard, J. Zhang, A. Sanchez, J. Harris, and J. Feng, "Integration of spatial–spectral information for the improved extraction of endmembers," *Remote Sens. Environ.*, vol. 110, no. 3, pp. 287–303, 2007.

[6] S. Lopez, J. F. Moure, A. Plaza, G. M. Callico, J. F. Lopez, and R. Sarmiento, "A new preprocessing technique for fast hyperspectral endmember extraction,"

IEEE Geoscience and Remote Sensing Letters, vol. 10, no. 5, pp. 1070–1074, September 2013.

[7] M. Zortea and A. Plaza, "Spatial preprocessing for endmember extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, pp. 2679–2693, 2009.

[8] G. Martin and A. Plaza, "Region-based spatial preprocessing for endmember extraction and spectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 745–749, 2011.

[9] G. Martin and A. Plaza., "Spatial-spectral preprocessing prior to endmember identification and unmixing of remotely sensed hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 5, no. 2, pp. 380–395, 2012.

[10] J. Plaza, E. M. T. Hendrix, I. Garcia, G. Martin, and A. Plaza, "On endmember identification in hyperspectral images without pure pixels: A comparison of algorithms," *Journal of Mathematical Imaging and Vision*, vol. 42, no. 2-3, pp. 163–175, 2012.

[11] J. A. R. y X. Jia, "Remote Sensing Digital Image Analysis," in *Springer- Verlag*, 1999.

[12] J. W. Boardman, F. A. Kruse, and R. O. Green, "Mapping Target Signatures Via Partial Unmixing of Aviris Data," *Proc. JPL Airborne Earth Sci. Workshop*, pp. 23–26, 1995.

[13] J. Delgado, G. Martin, J. Plaza, L. I. Jimenez, and A. Plaza, "Fast spatial preprocessing for spectral unmixing of hyperspectral data on graphics processing units," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 2, pp. 952–961, 2016.

[14] L. I. Jimenez, G. Martin, S. Sanchez, J. Plaza, and A. Plaza, "GPU implementation of spatial-spectral preprocessing for hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, submitted, 2016.

[15] J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado, "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geoscience and Remote Sensing Letters*, vol. 43, no. 3, pp. 441–445, 2007.

[16] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, pp. 779–785, 1994.

[17] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[18] S. Lahabar and P. J. Narayanan, "Singular value decomposition on GPU using CUDA," *Proceedings of the IEEE Parallel and Distributed Processing Symposium*, pp. 1–10, 2009.

