Transformada de Fourier aplicada al alineamiento de imágenes multidimensionales en GPU

Álvaro Ordóñez[1], Francisco Argüello[2] y Dora B. Heras[1]

Resumen— **La alineación de imágenes es una tarea previa fundamental en numerosas aplicaciones de imágenes. La gran cantidad de información que contienen las imágenes multidimensionales puede ser utilizada para conseguir alineaciones más precisas. Además, la mayoría de los algoritmos de alineación ignoran el tiempo de cómputo. En este artículo presentamos un algoritmo de alineación de imágenes multidimensionales basado en la transformada rápida de Fourier, el análisis de componentes principales y la combinación de mapas log-polar. El algoritmo propuesto hace uso de la correlación de fase para recuperar los parámetros que permiten alinear la imagen. Asimismo, se ha realizado una implementación eficiente sobre Unidades de Procesamiento Gráfico (GPUs). Diferentes técnicas de optimización, como el uso de bibliotecas CUDA optimizadas y el uso eficiente de las diferentes memorias, han sido aplicadas para obtener el mejor rendimiento. Como resultado, el algoritmo es robusto y ha sido probado con diferentes imágenes consiguiendo alineaciones de hasta una escala 6.0× para todos los ángulos, y con una aceleración de la implementación paralela en GPU de hasta 165.4× respecto de la implementación secuencial en CPU.**

Palabras clave— **Imagen multidimensional, alineación de imágenes, teledetección, transformada de Fourier, GPU, CUDA.**

I. Introducción

ACTUALMENTE, gracias a los últimos avances tecnológicos, la obtención de imágenes multidimensionales mediante sensores espectrales es más sencilla que hace dos décadas. Estas imágenes se caracterizan por estar formadas por cientos de bandas que cubren un amplio rango del espectro electromagnético [1]. Su uso se ha extendido a multitud de aplicaciones como la agricultura [2], control de calidad [3], medicina [4], vigilancia [5], control medio ambiental [6] o geología entre otras.

La alineación de imágenes es una tarea previa fundamental en muchas de estas aplicaciones. El problema estudiado en este artículo consiste en estimar la transformación de similitud existente entre una imagen de referencia y una segunda imagen de la misma escena pero capturadas en diferentes instantes de tiempo. Existen diferentes algoritmos para resolver el problema de alineación automática de imágenes [7], [8], [9].

Los métodos basados en la transformada de Fourier (FT) son muy eficientes ya que se usa la transformada rápida de Fourier (FFT) para calcular la correlación cruzada entre las dos imágenes. Además, estos destacan por su resistencia al ruido, oclusiones, y otros defectos típicos que se pueden encontrar en las imágenes de teledetección. Chen, Defrise, y Deconinck [10] propusieron en 1994 un método para alinear imágenes usando un mapa log-polar. Este método es también conocido como Fourier-Mellin Invariant Symmetric Phase-Only Matched Filtering (FMI-SPOMF) [11], [12].

Otro enfoque en alineación de imágenes es el que se basa en métodos de detección de características. Estos métodos extraen y emparejan características relevantes que son comunes entre las dos imágenes. Las características emparejadas nos permiten determinar la transformación geométrica que es necesaria para alinear las imágenes. El método Scale-Invariant Feature Transform (SIFT) [13] ha demostrado ser uno de los mejores métodos de extracción de características. Destaca por su robustez ante cambios de iluminación y de perspectiva, y es invariante a cambios de escala o rotación. Estas propiedades lo convierten en un excelente método para la alineación de imágenes [14], [15], [16]. A partir de la presentación de SIFT se han desarrollado un gran número de variantes y extensiones del mismo. El método Speeded-Up Robust Features (SURF) [17] fue desarrollado para disminuir el elevado tiempo de ejecución de SIFT así como para mejorar la distinción de los puntos característicos.

Los cientos de bandas capturadas por los sensores multidimensionales pueden ser consideradas como una secuencia de imágenes a diferentes longitudes de onda. Debido al tiempo de captura requerido entre una longitud de onda y otra puede ser necesario alinear las diferentes bandas de la imagen. Para este propósito en [18] proponen el uso de la información mutua multivariable como medida de similitud. En [19] han desarrollado métodos basados en la descomposición mediante *wavelets*, detección de bordes, uso de información mutua y emparejamiento de características para realizar reducción de dimensionalidad, alineado, y fusión de imágenes multisensor. En [20] se presenta un método basado en la transformada de Fourier para alinear las bandas de una imagen recuperando la escala, rotación y traslación, y poder así construir cubos multidimensionales coherentes. Un esquema similar es el que se propone en [21]. En cambio en [22] se presenta un método de alineamiento de bandas basado en diferentes descriptores de características como SIFT, SURF, y Affine-SIFT (ASIFT).

Una tarea diferente es la alineación de dos imáge-

[1]Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS), Universidade de Santiago de Compostela, e-mail: {alvaro.ordonez, dora.blanco}@usc.es.

[2]Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, e-mail: francisco.arguello@usc.es.

nes multicanal. En [23] se presenta un método para la alineación de dos imágenes multicanal usando una correlación multivariable como medida de similitud. La mayoría de los métodos de alineación de este tipo de imágenes son específicos para el alineamiento de datos médicos, tal como las angiografías de resonancia magnética (en inglés, ARM) y las imágenes de resonancia médica (en inglés, MRI) [24], [25].

En el campo de la teledetección se han propuesto varios métodos para la alineación de imágenes multidimensionales. En [26] usan un algoritmo genético como técnica de búsqueda para estimar los parámetros óptimos que permiten alinear dos imágenes multiespectrales. Este enfoque puede ser considerado como una fusión de diferentes decisiones resultantes de la alineación de varias bandas. En [27] proponen usar el algoritmo de esperanza-maximización (EM) para resolver el problema de alineación y fusión de imágenes. Los parámetros de alineado son estimados maximizando una función condicional de esperanza. [28] presenta un método de alineación basado en la generación de una red de puntos de control a partir de la construcción de una pirámide de imágenes de diferentes resoluciones. Sin embargo ninguno de estos métodos está basado en la transformada de Fourier.

Por otro lado, la mayoría de estos algoritmos ignoran el tiempo de computación. En aplicaciones donde es necesario obtener un resultado en tiempo real, el tiempo de ejecución es crucial, por ejemplo, en una misión de búsqueda y rescate o en control de daños y desastres. Las Unidades de Procesamiento Gráfico (GPUs) han demostrado ser muy apropiadas para resolver eficientemente diferentes problemas en procesamiento de imágenes multidimensionales [29], [30], [31]. En la literatura podemos encontrar implementaciones de algoritmos en GPU para la alineación de imágenes de 2 y 3 dimensiones [32], [33], [34], [35], pero ninguna de ellas aplicada a imágenes multidimensionales las cuales suelen centuplicar la cantidad de datos de una imagen normal y en las cuales existe una alta correlación en la información espectral.

En este artículo se presenta la implementación en GPU del algoritmo para alineación de imágenes multidimensionales llamado PCA-based Hyperspectral Fourier-Mellin (PCA-HFM). El método explota la información espacial y espectral contenida en las diferentes bandas de las imágenes y está basado en el análisis de componentes principales (PCA), el algoritmo FMI-SPOMF, y la combinación de mapas logpolar. Ha sido desarrollado en CUDA para obtener una explotación eficiente de la arquitectura GPU. El resto del artículo se organiza de la siguiente forma. En primer lugar, la Sección II describe el algoritmo y su implementación en GPU. En la Sección III se presentan y discuten los resultados en términos de precisión de alineado y tiempos de ejecución. Por último, en la Sección IV se presentan las conclusiones.

II. Alineación de imágenes multidimensionales

En esta sección se explican las diferentes etapas del algoritmo PCA-HFM así como su implementación en GPU. El algoritmo se compone de 4 etapas (ver Fig. 1). En la primera etapa se realiza una reducción de dimensionalidad quedándonos solo con 8 componentes de cada imagen. A continuación, se calculan los mapas log-polar de cada componente y se aplica la correlación de fase a cada par de mapas. En la tercera etapa los diferentes mapas log-polar son combinados para integrar la información de todos los pares de componentes. Finalmente, en la última etapa, se procesan los picos de mayor valor para volver aplicar la correlación de fase sobre el dominio cartesiano y obtener la escala, rotación y traslación que permite alinear las imágenes.

A continuación se introducen algunos conceptos fundamentales de programación en CUDA, y se explica en detalle la implementación del algoritmo.

A. Fundamentos de programación en CUDA

CUDA es una arquitectura de cálculo paralelo de NVIDIA creada para explotar la gran potencia de las GPUs en aplicaciones de cualquier índole. Los programas en CUDA se construyen mediante funciones paralelas llamadas *kernels* [36]. Cada *kernel* es ejecutado paralelamente por un conjunto de hilos. Esto es, cada hilo ejecuta una instancia del *kernel* siguiendo el modelo de programación Una Instrucción, Múltiples Hilos (en inglés, SIMT). Un *warp* es el número de hilos que puede ejecutar de forma concurrente un multiprocesador. El programador decide el número de hilos a emplear en cada *kernel* y los organiza en una rejilla de bloques. Una rejilla es un conjunto de bloques que son ejecutados paralelamente y que son independientes. Un bloque es un conjunto de hilos que trabajan conjuntamente y comparten recursos como un mismo espacio en la memoria compartida.

En la nueva arquitectura Pascal, la jerarquía de memoria ha cambiado. En comparación con arquitecturas anteriores, en Pascal ya no es necesario seleccionar cómo queremos dividir la cantidad de memoria disponible entre la memoria compartida y la caché L1. Cada multiprocesador de la GPU GP104 contiene 96 KB de memoria compartida dedicada y 48 KB de memoria caché L1. Por otro lado, en Pascal, la memoria caché L1 puede actuar como una caché de texturas dependiendo de la carga de trabajo. Esta memoria caché unificada actuá como un *buffer* para realizar accesos coalescentes a memoria. Además, la GP104 cuenta con 2048 KB de memoria caché L2.

Con el fin de obtener el mejor rendimiento y reducir el tiempo computacional hemos empleado diferentes estrategias para explotar eficientemente la arquitectura de la GPU. Primeramente, el algoritmo ha sido desarrollado íntegramente en CUDA, reduciendo así las transferencias de datos entre la CPU y GPU al inicio y al final de la ejecución. Además, ha sido desarrollado minimizando el uso de memo-

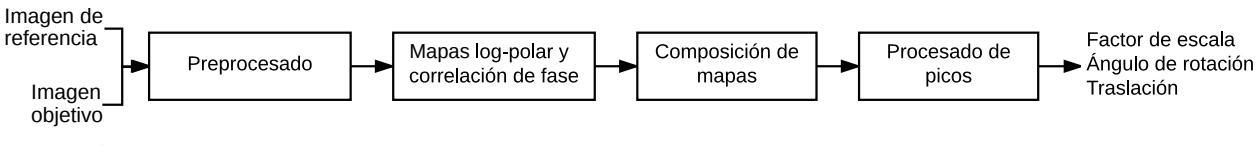


Fig. 1: Esquema del algoritmo PCA-HFM para la alineación de dos imágenes multidimensionales en GPU.

ria necesaria, y para que haga un uso eficiente de los diferentes tipos de memoria según el caso. Cuando no sea necesario mantener los datos anteriores a un cálculo, la salida se escribirá en las mismas posiciones de memoria si es posible (computaciones *in-place*).

En nuestra implementación, generalmente, las imágenes y otros datos de gran tamaño se encuentran almacenados en la memoria global, y es en esta donde se realizan la mayoría de los cómputos al no existir reutilización ni accesos no coalescentes. Además, en las arquitecturas actuales estos accesos a memoria global son menos costosos gracias a las últimas mejoras de las memorias cachés.

La latencia de la memoria compartida es aproximadamente 100 veces menor que la latencia de acceso a memoria global no cacheada previamente. Es por eso que esta es usada en aquellos *kernels* donde se accede numerosas veces a los mismos datos en memoria global.

Por otro lado, la memoria de texturas funciona como una caché de solo lectura que incluye filtrado por hardware y que como parte del proceso de lectura puede realizar interpolación lineal en punto flotante. Si se compara con el típico esquema de caché en CPU, la memoria de texturas está optimizada para acelerar accesos que sigan patrones de localidad espacial 2D. Es por eso que es ideal en cálculos de interpolación como el que se utiliza en la computación del mapa log-polar.

Por otro lado, NVIDIA proporciona bibliotecas optimizadas en GPU con funciones muy utilizadas. Por ejemplo, cálculo matricial, procesado primitivo de imágenes, cálculo de la FFT, entre otros. Se ha hecho uso de las mismas para obtener el mejor rendimiento.

Se ha utilizado el NVIDIA Visual Profiler para analizar cada *kernel* e identificar posibles limitadores del rendimiento. Usando esta herramienta se ha buscado la configuración para cada *kernel* que permita computarlo en el menor tiempo posible y obtener ocupaciones óptimas de hardware. En general, los bloques son de 256 hilos, exceptuando los *kernels* del cálculo de las componentes principales (PCA) donde un tamaño de bloque de 512 hilos consigue un mejor rendimiento. Del mismo modo, los tamaños de bloque escogidos son múltiplos del tamaño de *warp*, actualmente 32 en todas las arquitecturas, lo que garantiza que los accesos a memoria de los *warps* estén alineados con las líneas de la caché.

Por último, se han utilizado funciones de más bajo nivel para obtener una mejor precisión y rendimiento [37], especialmente cuando estas funciones son llamadas desde bucles.

B. Alineación de imágenes multidimensionales en GPU

En esta sección se describe la implementación en GPU del algoritmo de alineación PCA-HFM compuesto de 4 etapas.

B.1 Preprocesado e inicializaciones

El algoritmo comienza precomputando datos auxiliares que serán necesarios repetidamente en la computación del filtro pasa-alta y de los mapas log-polar. En esta primera etapa, se aplica una ventana de Blackman a cada banda espectral de ambas imágenes para eliminar aquellas altas frecuencias que afectan gravemente a la precisión de la alineación [38]. Esta operación consiste en una multiplicación punto a punto, es por eso que se computa en memoria global.

A continuación, se aplica el método PCA para obtener las componentes principales de las imágenes multidimensionales. Esto nos permite reducir la dimensión de las imágenes y obtener un nuevo conjunto de datos con N_{PCA} componentes no correlacionadas para cada imagen, y donde las primeras componentes retienen la mayor parte de la variación presente en los datos originales. Este cómputo se ha realizado mediante el método de descomposición en autovalores (EVD) [39]. Para ello debemos primero centrar la imagen restando a cada píxel el valor medio de su banda correspondiente usando para ello la función *cublasSgemv* de cuBLAS [40]. La matriz de covarianza es calculada usando la función *cublasSgemm*. Usando la función *cusolverDnSgesvd* de cuSOLVER [41] obtendremos los valores y vectores singulares de dicha matriz de covarianza. El último paso para la obtención de las PCs se realiza mediante la función *cublasSgemm* de cuBLAS, la cual realiza el producto entre la imagen y la conjugada transpuesta de la matriz unitaria obtenida en la descomposición.

Por último, las imágenes son extendidas a un tamaño potencia de 2 para obtener un cómputo de la FFT más eficiente mediante cuFFT [42]. Al mismo tiempo esto nos permite evitar divergencias pues no tendremos que configurar más hilos de los necesarios al ser el tamaño de las imágenes múltiplo del tamaño de *warp*. Las imágenes son centradas en el nuevo tamaño y se unifica el color de cada banda usando la memoria compartida.

B.2 Mapas log-polar y correlación de fase

La siguiente etapa, mapas log-polar y correlación de fase, comienza con la transformación de cada componente principal de las dos imágenes al dominio de la frecuencia, mediante el uso de la FFT. Se ha usado

la librería cuFFT [43] para obtener el mejor rendimiento en el cómputo de las transformadas.

A continuación se aplica un filtro pasa-alta a cada componente. Este ya ha sido precomputado en la función de inicializaciones para ahorrar tiempo de cómputo. Este simple filtro reduce los efectos de *aliasing*, muy frecuentes en las las frecuencias más débiles de la transformada de Fourier, y que afectan a la precisión de la alineación de imágenes usando métodos basados en la correlación de fase [12].

Aplicado el filtro pasa-alta, el siguiente paso es calcular los mapas log-polar para poder recuperar los parámetros de escala ρ_0 y de rotación θ_0. La transformación a coordenadas log-polar se realiza mediante interpolación sobre las componentes filtradas. Para ello es necesario realizar un gran número de accesos a memoria y realizar interpolaciones, por lo que la memoria de texturas es la mejor opción en términos de rendimiento como ya se ha dicho en la Sección II-A.

Por último, la correlación de fase es computada en cada par de mapas log-polar, uno de cada imagen. Este método de alineación se basa en el teorema de desplazamiento de la transformada de Fourier, por la cual un desplazamiento circular en el dominio espacial es equivalente a un desplazamiento de fase en el dominio de la frecuencia.

Supongamos que $I_1(x, y)$ y $I_2(x, y)$ son dos imágenes, donde (x, y) son sus coordenadas espaciales, y que $I_2(x, y)$ es una réplica generada tras trasladar $I_1(x, y)$ (x_0, y_0) coordenadas,

$$I_2(x, y) = I_1(x + x_0, y + y_0), \tag{1}$$

por el teorema de desplazamiento sus transformadas de Fourier están relacionadas

$$F_2(u, v) = \mathrm{e}^{\mathrm{i}2\pi(\mathrm{u}\mathrm{x}_0+\mathrm{v}\mathrm{y}_0)}\mathrm{F}_1(\mathrm{u}, \mathrm{v}), \tag{2}$$

donde (u, v) son las coordenadas en el dominio de la frecuencia e i es la unidad imaginaria.

Aplicando la correlación de fase en el dominio de la frecuencia,

$$\frac{F_1(u, v)F_2^*(u, v)}{|F_1(u, v)F_2^*(u, v)|} = \mathrm{e}^{-\mathrm{i}2\pi(\mathrm{u}\mathrm{x}_0+\mathrm{v}\mathrm{y}_0)}, \tag{3}$$

y, computando la transformada inversa de Fourier, obtenemos una matriz formada casi en su totalidad por ceros excepto en el pico con coordenadas (x_0, y_0), el cual nos permite alinear las imágenes.

Para recuperar el factor de escalado y el ángulo de rotación, en [10] se propone el uso de la transformada log-polar de Fourier, también conocida como transformada Fourier-Mellin. Supongamos que I_2 es una imagen réplica de I_1 pero escalada con un factor ρ_0 y rotada con un ángulo θ_0, esto es,

$$\begin{aligned} I_2(x, y) = I_1(&\rho_0(x\cos\theta_0 + y\sin\theta_0 + x_0),\\ &\rho_0(-x\sin\theta_0 + y\cos\theta_0 + y_0)), \end{aligned} \tag{4}$$

aplicando el teorema de desplazamiento de Fourier sobre las coordenadas polares (ρ, θ) obtenemos,

$$F_2(\rho, \theta) = \mathrm{e}^{\mathrm{i}(\omega_\mathrm{x}\mathrm{x}_0+\omega_\mathrm{y}\mathrm{y}_0)}\rho_0^{-2}\mathrm{F}_1(\rho_0^{-1}\rho, \theta + \theta_0), \tag{5}$$

donde $\omega_x = \rho\cos\theta$ y $\omega_y = \rho\sin\theta$.

Por otra parte, aplicando el módulo de la transformada de Fourier y el logaritmo de radio ρ, tenemos

$$|F_2(\ln\rho, \theta)| = \rho_0^{-2}|F_1(\ln\rho - \ln\rho_0, \theta + \theta_0)|, \tag{6}$$

donde ρ_0 y θ_0 pueden ser obtenidos aplicando nuevamente el teorema de desplazamiento de Fourier y la correlación de fase. En este caso, volveremos obtener una matriz con prácticamente ceros en todas sus posiciones con excepción el pico con coordenadas $(\ln\rho_0, \theta_0)$. El cómputo de las transformadas de Fourier necesarias se realiza nuevamente usando la biblioteca cuFFT.

B.3 Composición de mapas

Los mapas log-polar tras la correlación son combinados en la etapa de composición. Los diferentes experimentos que hemos realizado han demostrado que promediando estos mapas nos permite obtener mejores resultados, pues cada par de componentes PCA contribuye independientemente en la correlación de fase atenuando aquellos falsos picos que nos llevarían a obtener unos parámetros de alineación incorrectos.

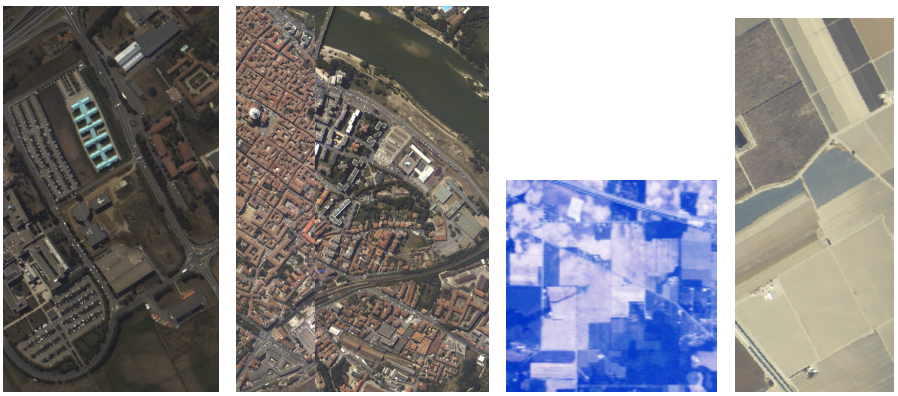
B.4 Procesado de picos

Una vez que los mapas de correlación son combinados, obtenemos una matriz con ceros en casi todas sus posiciones exceptuando algunas de estas. Si las dos imágenes a alinear fueran iguales tan solo obtendríamos un pico, es decir, un valor distinto de 0, cuyas coordenadas nos permiten alinear las imágenes. Pero debido a distorsiones y otros efectos se obtienen numerosos picos, y es por eso que debemos analizar más de uno en busca del que nos permita alinear las imágenes.

Esta etapa está formada por dos pasos: el procesado de los picos de los mapas de correlación después de la log-polar para obtener el factor de escala y el ángulo de rotación, y el procesado de los picos en el dominio cartesiano tras corregir la rotación y el escalado para obtener la traslación.

En el primer paso de esta etapa, empezamos analizando los N_{Peaks} mayores picos de los mapas de correlación de la log-polar. Para eso primero se ordenan los picos por magnitud usando la función *sort_by_key* de la biblioteca Thrust. Esta biblioteca selecciona automáticamente la implementación más eficiente para dicha ordenación [44].

A partir de las coordenadas de cada uno de estos N_{Peaks} picos obtenemos un factor de escala ρ_0 y un ángulo de rotación θ_0. Con cada uno de estos pares de parámetros se escala y se rota (θ_0 y $\theta_0 + \pi$ debido a la ambigüedad del mapa log-polar) la primera componente de la imagen objetivo utilizando las funciones *nppiRotate_32f_C1R* y *nppiResizeSqrPixel_32f_C1R* de la biblioteca NPP [45]. De esta manera si el pico fuera el acertado se estarían corrigiendo el escalado y rotación de la imagen, tan solo faltaría la traslación.



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

Fig. 2: Imágenes hiperespectrales de prueba comunes en el área de teledetección: (a) *Pavia University*, (b) *Pavia Centre*, (c) *Indian Pines*, y (d) *Salinas*.

El segundo paso de etapa etapa consiste en calcular la correlación de fase entre la primera componente de la imagen de referencia y las dos componentes rotadas y escaladas de la imagen de referencia. A continuación, seleccionamos el pico máximo de las dos correlaciones que obtenemos por cada N_{Peak} usando la función *cublasIcamax* de cuBLAS.

El pico máximo de los $2N_{\text{Peaks}}$ mapas cartesianos es el que nos permite seleccionar cual de los N_{Peaks} picos es el pico log-polar que posiblemente alinee de forma correcta la imagen. Finalmente, las coordenadas del pico seleccionado en el mapa log-polar son los que determinan el factor de escala ρ_0 y el ángulo de rotación θ_0, y las coordenadas del pico en el dominio cartesiano es el que determina los parámetros de traslación (x_0, y_0).

III. Resultados

En esta sección se muestran las imágenes de prueba y condiciones experimentales así como se presentan los resultados en términos de precisión de alineado y de tiempos de cómputo.

A. Condiciones experimentales

La versión CPU del algoritmo ha sido ejecutado sobre un ordenador con un procesador quad-core Intel Core i5-6600 a 3.3 GHz y 32 GB de RAM. Con respecto a la implementación en GPU, este se ha ejecutado en una NVIDIA GeForce GTX 1070 con 15 SMs con 128 núcleos CUDA cada uno. Los códigos han sido compilados en un sistema operativo Linux, en concreto, usando gcc versión 4.8.4 para la implementación en CPU, y nvcc y bibliotecas de NVIDIA versión 8.0.26 para la implementación en GPU. Todas los cómputos se han realizado en punto flotante con precisión simple.

Se han usado seis imágenes para realizar los experimentos de este trabajo. Estas han sido tomadas de los sensores ROSIS (Reflective Optics System Imaging Spectrometer) y AVIRIS (Airborne Visible/Infrared Imaging Spectrometer). El sensor ROSIS captura 115 bandas en el rango espectral 0.43 hasta 0.86 µm, correspondiendo principalmente al espectro visible. El sensor AVIRIS captura 224 bandas en el rango espectral 0.4 hasta 2.5 µm, correspondiente al espectro visible e infrarrojo.

Las cuatro primeras imágenes son comúnmente conocidas en el área de la teledetección hiperespectral (ver Fig. 2). La primera imagen utilizada es la imagen de la Universidad de Pavia (Italia). Es una imagen tomada con el sensor ROSIS con 103 bandas ya que se han eliminado 12 bandas con ruido. El tamaño de la imagen es 610×340 píxeles con una resolución espacial de 1.3 m por píxel.

La segunda imagen también ha sido capturada por ROSIS y se trata de la imagen del centro de Pavia (Italia). Originalmente tenía un tamaño de 1096×1096 píxeles, pero una zona central con 381 píxeles de ancho fue eliminada al no contener datos, resultando la imagen final con un tamaño de 1096×715 píxeles. También fuero eliminadas 13 bandas por contener ruido, quedando finalmente 102 bandas con una resolución espacial de 1.3 m por píxel.

La tercera imagen, Indian Pines Test Site 3, es una imagen de una zona agrícola de Tippecanoe County (Indiana) y fue tomada por el sensor AVIRIS. Consta de 220 bandas (cuatro fueron eliminadas) con resolución espacial de 20 m por píxel y tiene un tamaño de 145×145 píxeles.

La cuarta imagen fue capturada por el sensor AVIRIS sobre el Valle de Salinas (California). Es una imagen de una zona rural con un tamaño de 512×217 píxeles y 204 bandas, tras eliminar las correspondientes a la absorción del agua y algunas bandas de ruido. Su resolución espacial es de 3.7 m por pixel.

Las últimas dos imágenes han sido capturadas por el sensor AVIRIS en la Bahía de San Francisco (California). Están formadas en su mayoría por zonas de cultivo pero también por zonas edificadas. La primera imagen fue tomada el 7/6/2013 y tiene un tamaño original de 11342×744 píxeles con una resolución espacial de 16.9 m por píxel. La segunda imagen fue tomada el 11/06/2015 y tiene un tamaño original de 11247×828 con una resolución espacial de 17 m por píxel. En la figura 3 se muestra el fragmento de ta-

TABLA I: Casos correctamente alineados para cada escena. El número entre paréntesis indica el número de escalas que fueron recuperadas correctamente para todos los ángulos.

Imagen	PCA-HFM (8 PCs)	PCA-HFM (1 PC)	SIFT	SURF
Pavia University	1/4× to 4.5× (11)	1/4× to 3.5× (9)	1/2× to 2.0× (4)	1/4× to 3.0× (8)
Pavia Centre	1/4× to 6.0× (14)	1/4× to 5.5× (13)	1/5× to 2.5× (8)	1/6× to 5.5× (15)
Indian Pines	1/2× to 3.5× (7)	1/2× to 3.0× (6)	1/2× to 1.0× (2)	1.0× (1)
Salinas	1/2× to 4.0× (8)	1/2× to 4.0× (8)	(0)	1/2× to 1.5× (3)
Bay Area Box Line	1/2× to 2.0× (3)	1.0× (1)	1/2× to 1.0× (2)	1.0× (1)
Promedio de escalas correctas	(8.60)	(7.40)	(3.20)	(5.60)

maño 1432 × 688 que ha sido seleccionado. Ambas tienen 224 bandas espectrales y presentan, además del problema de alineación, cambios en la vegetación, edificios, e infraestructuras.

Todas estas imágenes pueden ser descargadas de `http://gitlab.citius.usc.es/hiperespectral/RegistrationRepository`.

(a) (b)

Fig. 3: Imágenes de prueba *Bay Area Box Line 12* capturadas por el sensor AVIRIS. (a) Fragmento de tamaño 1432×688 de la imagen tomada el 7/6/2013, y (b) Fragmento de tamaño 1432 × 688 de la imagen tomada el 11/06/2015.

B. Resultados de precisión de alineado

Para las cuatro primeras imágenes, las pruebas de alineado se han realizado sobre una versión de la misma rotada y escalada. De esta manera, podemos analizar todos los detalles en un entorno controlado. En el caso de las imágenes de la Bahía de San Francisco se ha seleccionado la imagen de 2015 como imagen objetivo y la imagen de 2013 como imagen de referencia. Con el fin de extender nuestro rango de experimentación se han aplicado diferentes factores de escala y ángulos de rotación a la imagen que pretendemos alinear en ambos casos.

El procedimiento de la experimentación ha sido el siguiente. Se ha realizado una búsqueda exhaustiva con factores de escala desde 1/6× hasta 7× en incrementos de 0.5× y con ángulos de rotación desde 0º hasta 360º en incrementos de 5º (72 ángulos). Para los factores de escala mayores a 1× se ha mantenido el tamaño original de la imagen seleccionando la parte central de la imagen escalada.

En la Tabla I se muestran las escalas correctamente recuperadas para todos sus 72 ángulos para cada una de las imágenes usando dos configuraciones diferentes del algoritmo propuesto PCA-HFM y dos métodos relevantes de la literatura. La primera y segunda columnas muestran los resultados para nuestra propuesta usando 8 y 1 componentes. La tercera y cuarta columna muestran los resultados obtenidos para dos métodos basados en la extracción de características, pues como se ha comentado en la Sección 1 son robustos a cambios de iluminación, rotación y escalado. Estos son los métodos SIFT [46] y SURF [17]. Para ambos algoritmos se ha realizado el emparejamiento de los puntos por fuerza bruta usando la distancia euclídea y se ha calculado la transformación afín usando el método Random Sample Consensus (RANSAC). Debido a la arbitrariedad de RANSAC, se han realizado 50 ejecuciones para cada prueba de alineamiento. Ambos algoritmos son utilizados empleando la primera componente.

El método propuesto PCA-HFM con 8 PCs es el que obtiene de media los mejores resultados, alineando correctamente 8.60 escalas frente a 7.40 usando solo 1 PC. Realizando la composición de los mapas de correlación de las 8 primeras componentes se descartan picos que nos llevarían a alineaciones incorrectas, que es lo que sucede utilizando solo la primera componente. Los algoritmos SURF y SIFT son los que obtienen peores resultados, 5.60 y 3.20 escalas correctas, respectivamente. La capacidad limitada de estos métodos resida probablemente en la identificación de los puntos característicos cuando las imágenes presentan cambios. Para las imágenes con mejor resolución espacial, Pavia Centre y Pavia University, es para las que se consigue alinear un mayor número de escalas, 14 y 11, respectivamente.

B.1 Resultados de tiempos de computación

En esta sección comparan los tiempos de computación de la implementación secuencial en CPU y la versión paralelizada en GPU del método propuesto, PCA-HFM con 8 PCs. Estos resultados se han obtenido realizando un promedio de 10 ejecuciones por experimento. Todas las tablas muestran los tiempos en segundos y no incluyen los tiempos de lectura de datos y de transferencias CPU-GPU, salvo que se indique lo contrario.

En la Tabla II se muestran los tiempos de utiliza-

TABLA II: Tiempos de ejecución (en segundos) en CPU y en GPU detallados en funciones para la alineación de las imágenes de la Bahía de San Francisco.

Función	CPU	GPU	Aceleración
Blackman	0.122s	**0.064s**	1.9×
PCA	2.058s	**0.249s**	8.2×
Extensión	0.126s	**0.005s**	25.4×
FFT	9.881s	**0.033s**	297.2×
High-pass	2.194s	**0.015s**	146.1×
Log-polar	2.206s	**0.015s**	150.6×
Correlación de fase	131.076s	**0.443s**	296.0×
Composición	0.008s	**0.001s**	6.3×
Ordenar picos	0.093s	**0.004s**	21.3×
Escalar y rotar	35.985s	**0.160s**	225.3×
Evaluación de picos	0.239s	**0.018s**	13.2×

ción en CPU y GPU para el alineado de las imágenes de la Bahía de San Francisco para cada función de la implementación. La mayoría del tiempo se ha empleado en la computación de la correlación de fase, la FFT, y en el escalado y rotado de la primera componente. La correlación de fase está compuesta por varias transformadas FFT, y es utilizada repetidamente a lo largo del algoritmo. Esta, junto con la propia FFT, han sido las dos funciones que mejor aceleración han obtenido en su implementación en GPU gracias al buen rendimiento de la biblioteca cuFFT. La operación de escalado y rotación también es muy costosa computacionalmente ya que conlleva cálculos aritméticos e interpolaciones. El uso de la memoria de texturas ha permitido obtener una muy buena aceleración en el cálculo de los mapas log-polar.

TABLA III: Tiempos de ejecución (en segundos) para las implementaciones en CPU y en GPU para cada imagen.

Imagen	CPU	GPU CUDA	Aceleración
Pavia University	40.80s	**0.28s**	145.7×
Pavia Centre	178.06s	**0.85s**	209.5×
Indian Pines	2.32s	**0.15s**	15.5×
Salinas	9.61s	**0.20s**	48.1×
Bay Area	183.99s	**1.01s**	182.2×

La Tabla III muestra los tiempos de ejecución del algoritmo en CPU y en GPU para todas las imágenes. Las aceleraciones más grandes se consiguen en las imágenes de mayor tamaño, 209.5× y 182.2× para las imágenes Pavia Centre y Bay Area, respectivamente.

TABLA IV: Tiempos de ejecución (en segundos) para las implementaciones en CPU y en GPU para cada imagen incluyendo los tiempos de lectura de imágenes y las transferencias CPU-GPU.

Imagen	CPU	GPU CUDA	Aceleración
Pavia University	40.94s	**0.35s**	118.7×
Pavia Centre	178.65s	**1.08s**	165.4×
Indian Pines	2.34s	**0.17s**	14.0×
Salinas	9.78s	**0.27s**	36.2×
Bay Area	185.81s	**1.65s**	113.0×

Los tiempos de ejecución incluyendo los tiempos de lectura de las imágenes y de transferencia de datos CPU-GPU son presentados en la Tabla IV. De igual forma, las mejores aceleraciones se consiguen en las imágenes de mayor tamaño.

IV. Conclusiones

En este artículo se presenta un algoritmo de alineación de imágenes multidimensionales en GPU, PCA-HFM. Esta basado en la transformada rápida de Fourier, el cálculo de componentes principales mediante PCA, la correlación de fase, y la combinación de mapas log-polar. Se han implementado una versión secuencial en CPU y una versión paralela en CUDA para su completa ejecución en una GPU NVIDIA.

Se han realizado experimentos con 6 imágenes multidimensionales con diferentes tamaños y de distintos sensores, consiguiendo resultados satisfactorios en términos de precisión de alineado y en tiempo de cómputo. Se ha llegado a alinear correctamente escalas de hasta 6.0× para todos los ángulos. Las funciones más costosas y mejor optimizadas son aquellas que contienen cálculos de FFTs, la correlación de fase y la propia transformación al dominio de la frecuencia. Se han conseguido para la imagen Pavia Centre, con tamaño 1096×715 píxeles y 102 bandas, un tiempo de alineación de 1.08s en GPU y una aceleración de 165.4× respecto de la versión secuencial en CPU.

Agradecimientos

El presente trabajo ha sido financiando por la Consellería de Cultura, Educación e Ordenación Universitaria [GRC2014/008 y ED431G/08] y el Ministerio de Educación, Cultura y Deporte [TIN2013-41129-P y TIN2016-76373-P], ambos cofinanciados por el Fondo Europeo de Desarrollo Regional (FEDER).

Referencias

[1] David Landgrebe, "Hyperspectral image data analysis," *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 17–28, 2002.

[2] Driss Haboudane, John R Miller, Elizabeth Pattey, Pablo J Zarco-Tejada, and Ian B Strachan, "Hyperspectral vegetation indices and novel algorithms for predicting green lai of crop canopies: Modeling and validation in the context of precision agriculture," *Remote sensing of environment*, vol. 90, no. 3, pp. 337–352, 2004.

[3] Laura M Dale, André Thewis, Christelle Boudry, Ioan Rotar, Pierre Dardenne, Vincent Baeten, and Juan A Fernández Pierna, "Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: a review," *Applied Spectroscopy Reviews*, vol. 48, no. 2, pp. 142–159, 2013.

[4] Donald B Malkoff and William R Oliver, "Hyperspectral imaging applied to forensic medicine," in *BiOS 2000 The International Symposium on Biomedical Optics*. International Society for Optics and Photonics, 2000, pp. 108–116.

[5] P WT Yuen and Mark Richardson, "An introduction to hyperspectral imaging and its application for security, surveillance and target acquisition," *The Imaging Science Journal*, vol. 58, no. 5, pp. 241–253, 2010.

[6] Azadeh Ghiyamat and Helmi ZM Shafri, "A review on hyperspectral remote sensing for homogeneous and heterogeneous forest biodiversity assessment," *International Journal of Remote Sensing*, vol. 31, no. 7, pp. 1837–1856, 2010.

[7] Suma Dawn, Vikas Saxena, and Bhudev Sharma, "Remote sensing image registration techniques: A survey,"

in *International Conference on Image and Signal Processing*. Springer, 2010, pp. 103–112.

[8] Manjusha Deshmukh and Udhav Bhosle, "A survey of image registration," *International Journal of Image Processing (IJIP)*, vol. 5, no. 3, pp. 245, 2011.

[9] Lisa Gottesfeld Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[10] Qin-sheng Chen, Michel Defrise, and Frank Deconinck, "Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 12, pp. 1156–1168, 1994.

[11] B Dasgupta and BN Chatterji, "Fourier-Mellin transform based image matching algorithm," *IETE Journal of Research*, vol. 42, no. 1, pp. 3–9, 1996.

[12] B Srinivasa Reddy and Biswanath N Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[13] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] Peter Schwind, Sahil Suri, Peter Reinartz, and Andreas Siebert, "Applicability of the sift operator to geometric sar image registration," *International Journal of Remote Sensing*, vol. 31, no. 8, pp. 1959–1980, 2010.

[15] Yan Ke and Rahul Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, 2004, vol. 2, pp. II–II.

[16] Qiaoliang Li, Guoyou Wang, Jianguo Liu, and Shaobo Chen, "Robust scale-invariant feature matching for remote sensing image registration," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 2, pp. 287–291, 2009.

[17] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[18] Jeffrey P Kern, Marios Pattichis, and Samuel D Stearns, "Registration of image cubes using multivariate mutual information," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. IEEE, 2003, vol. 2, pp. 1645–1649.

[19] Jacqueline Le Moigne, Arlene Cole-Rhodes, Roger Eastman, Tarek El-Ghazawi, Kisha Johnson, S Knewpijit, Nadine Laporte, Jeffrey Morisette, Nathan S Netanyahu, Harold S Stone, et al., "Multiple sensor image registration, image fusion and dimension reduction of earth science imagery," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*. IEEE, 2002, vol. 2, pp. 999–1006.

[20] Hector Erives and Glenn J Fitzgerald, "Automated registration of hyperspectral images for precision agriculture," *Computers and Electronics in Agriculture*, vol. 47, no. 2, pp. 103–119, 2005.

[21] Jun Wang, Zhiyong Xu, and Jianlin Zhang, "Image registration with hyperspectral data based on Fourier-Mellin transform," *International Journal of Signal Processing Systems*, vol. 1, no. 1, 2013.

[22] Maria Vakalopoulou and Konstantinos Karantzalos, "Automatic descriptor-based co-registration of frame hyperspectral data," *Remote Sensing*, vol. 6, no. 4, pp. 3409–3426, 2014.

[23] Gustavo K Rohde, Sinisa Pajevic, Carlo Pierpaoli, and Peter J Basser, "A comprehensive approach for multi-channel image registration," in *International Workshop on Biomedical Image Registration*. Springer, 2003, pp. 214–223.

[24] Brian B Avants, Charles L Epstein, Murray Grossman, and James C Gee, "Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain," *Medical image analysis*, vol. 12, no. 1, pp. 26–41, 2008.

[25] Juan Ruiz-Alzola, C-F Westin, Simon K Warfield, C Alberola, S Maier, and Ron Kikinis, "Nonrigid registration of 3d tensor medical data," *Medical image analysis*, vol. 6, no. 2, pp. 143–161, 2002.

[26] Hani Mahdi and Aly A Farag, "Image registration in multispectral data sets," in *Image Processing. 2002. Proceedings. 2002 International Conference on*. IEEE, 2002, vol. 2, pp. II–II.

[27] Siyue Chen, Qing Guo, Henry Leung, and Eloi Bosse, "A maximum likelihood approach to joint image registration and fusion," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1363–1372, 2011.

[28] Peter Bunting, Frédéric Labrosse, and Richard Lucas, "A multi-resolution area-based technique for automatic multi-modal image registration," *Image and Vision Computing*, vol. 28, no. 8, pp. 1203–1219, 2010.

[29] Zebin Wu, Jiafu Liu, Antonio Plaza, Jun Li, and Zhihui Wei, "GPU implementation of composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1973–1977, 2015.

[30] Luis Ignacio Jiménez, Gabriel Martín, Sergio Sánchez, Carlos García, Sergio Bernabé, Javier Plaza, and Antonio Plaza, "GPU implementation of spatial–spectral preprocessing for hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 11, pp. 1671–1675, 2016.

[31] Javier López-Fandiño, Blanca Priego, Dora B Heras, and Francisco Argüello, "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*, vol. 10, no. 1, pp. 20–28, 2017.

[32] Alexander Köhn, Johann Drexl, Felix Ritter, Matthias König, and Heinz-Otto Peitgen, "GPU accelerated image registration in two and three dimensions," in *Bildverarbeitung für die Medizin 2006*, pp. 261–265. Springer, 2006.

[33] Pinar Muyan-Ozcelik, John D Owens, Junyi Xia, and Sanjiv S Samant, "Fast deformable registration on the GPU: A CUDA implementation of demons," in *Computational Sciences and Its Applications, 2008. ICCSA'08. International Conference on*. IEEE, 2008, pp. 223–233.

[34] Alexander Kubias, Frank Deinzer, Tobias Feldmann, DIETRICH Paulus, BERND Schreiber, and Th Brunner, "2d/3d image registration on the GPU," *Pattern Recognition and Image Analysis*, vol. 18, no. 3, pp. 381–389, 2008.

[35] Ramtin Shams, Parastoo Sadeghi, Rodney Kennedy, and Richard Hartley, "Parallel computation of mutual information on the GPU with application to real-time registration of 3d medical images," *Computer methods and programs in biomedicine*, vol. 99, no. 2, pp. 133–146, 2010.

[36] Jason Sanders and Edward Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*, Addison-Wesley Professional, 2010.

[37] NVIDIA, "CUDA C Best Practices Guide," Available: `http://docs.nvidia.com/cuda/pdf/CUDA_C_Best_Practices_Guide.pdf`, 2017, Accessed: 2017-04-05.

[38] Fredric J Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[39] 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.

[40] NVIDIA, "cuBLAS Library User's Guide," Available: `http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf`, 2017, Accessed: 2017-04-05.

[41] NVIDIA, "cuSOLVER Library User's Guide," Available: `http://docs.nvidia.com/cuda/pdf/CUSOLVER_Library.pdf`, 2017, Accessed: 2017-04-05.

[42] NVIDIA, "cuFFT Library User's Guide," Available: `http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf`, 2017, Accessed: 2017-05-02.

[43] NVIDIA, "cuFFT Library User's Guide," Available: `http://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf`, 2017, Accessed: 2017-04-05.

[44] NVIDIA, "Thrust Quick Start Guide," Available: `http://docs.nvidia.com/cuda/pdf/Thrust_Quick_Start_Guide.pdf`, 2017, Accessed: 2017-04-05.

[45] NVIDIA, "NVIDIA Performance Primitives (NPP) v8.0 User's Guide," Available: `http://docs.nvidia.com/cuda/pdf/NPP_Library_Image_Geometry.pdf`, 2017, Accessed: 2017-04-05.

[46] Ives Rey Otero, *Anatomy of the SIFT Method*, Ph.D. thesis, École normale supérieure de Cachan-ENS Cachan, 2015.