Implementación paralela del algoritmo HSI-MSER para el registro de imágenes hiperespectrales

Daniel del Castillo[1,2], Álvaro Ordóñez[3], Dora B. Heras[1,2] y Francisco Argüello[2]

Resumen— **El registro de imágenes es una tarea destinada a alinear imágenes como un paso anterior a otros tipos de procesado como son la detección de cambios o el seguimiento de objetos. En el caso de imágenes de teledetección hiperespectrales tenemos una gran cantidad de información tanto espacial como espectral para cada imagen. Esto ha propiciado la aparición de métodos de registro específicos para este tipo de imágenes que no solo usan las bandas RGB, como sucede en los algoritmos más clásicos, sino que además aprovechan la información espectral contenida en todas las bandas, que pueden llegar a ser del orden de un centenar. No obstante, para algunos de estos algoritmos, especialmente los basados en detección de características como líneas, puntos o regiones, el tiempo de ejecución sigue siendo una barrera para su uso en aplicaciones que requieren un procesado especialmente rápido. Uno de estos algoritmos es *Hyperspectral Image-Maximally Stable Extremal Regions* (HSI-MSER), que trata de encontrar regiones comunes en las imágenes explotando la información disponible en las bandas espectrales para conseguir un mejor registro. En este artículo se presenta una versión paralela del algoritmo HSI-MSER sobre una arquitectura heterogénea de bajo coste. Ha sido diseñada para explotar eficientemente las arquitecturas de la CPU y GPU usando OpenMP y CUDA, respectivamente. Como resultado, para imágenes hiperespectrales de teledetección disponibles en la bibliografía estado del arte se consiguen aceleraciones de hasta 7× respecto a la versión secuencial de HSI-MSER.**

Palabras clave— **Registro de imágenes, hiperespectral, teledetección, CUDA, GPU, OpenMP**

I. Introducción

El registro de imágenes es una operación crucial a la hora de trabajar con imágenes de teledetección de superficie terrestre capturadas por drones o satélites. Se puede definir como un proceso que toma dos imágenes que capturan información de la misma área geográfica y modifica la escala, la rotación y el desplazamiento de una de ellas para que se alinee con la otra. Esta técnica puede ser necesaria para diferentes propósitos. El más habitual es realizar un análisis de cambios sobre imágenes de la misma área capturadas en distintos momentos, a diferentes alturas o con equipos distintos.

Es importante destacar que existe una gran variedad de algoritmos de registro. La mayoría de ellos se pueden clasificar en dos tipos: basados en área o basados en características. Los métodos basados en área, como los basados en la *Fourier-Mellin transform* (FMT) [1], trabajan directamente con las intensidades buscando correlaciones para realizar el alineamiento. Por el contrario, los métodos basados en características, como *Speeded Up Robust Features* (SURF) [2], buscan ciertos elementos dentro de cada imagen, como líneas o puntos. Esto hace que tengan un coste computacional mayor que los basados en intensidad, pero, sin embargo, consiguen una mayor tolerancia a cambios en la imagen [3]. Esto permite registrar imágenes con mayores diferencias de escala entre ellas, lo que resulta de gran utilidad para ciertas aplicaciones cuando se ha de trabajar con imágenes obtenidas, por ejemplo, a alturas muy diferentes.

En la actualidad, cada vez son más los sistemas que incorporan sensores hiperespectrales que permiten obtener mapas detallados del terreno con una alta resolución espectral, lo que hace necesario disponer de técnicas de registro que hagan un uso eficiente y eficaz de toda la información que contienen. HSI-MSER [4] es un algoritmo para el registro de dos imágenes hiperespectrales basado en la detección de características, en este caso, regiones de interés. El método explota eficientemente la información espectral disponible en distintas bandas espectrales e incorpora este tipo de información en el descriptor que se construye para cada región detectada. Esto permite registrar imágenes con diferencias de escala de hasta 15×.

A pesar de ser más tolerantes a cambios, los métodos basados en características presentan como desventaja un mayor tiempo de ejecución, ya que se basan en una extracción profunda de características como, por ejemplo, regiones con determinadas propiedades sobre ambas imágenes. Las características son a continuación descritas mediante vectores de gran longitud para realizar después un proceso de emparejamiento entre las dos imágenes que requiere una búsqueda exhaustiva. Esta desventaja se hace notar especialmente a la hora de trabajar con imágenes hiperespectrales, ya que una primera aproximación exigiría buscar características sobre todas las bandas espectrales de ambas imágenes. Sin embargo, las empresas y usuarios que hacen uso de este tipo de imágenes no siempre tienen acceso a supercomputadores o infraestructuras computacionales costosas. Además, resulta mucho más conveniente en muchos

[1]Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, e-mail: {d.delcastillo,dora.blanco}@usc.gal.

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

[3]Universidade da Coruña, Grupo Integrado de Ingeniería, CITIC, Elviña, 15071 A Coruña, e-mail: alvaro.oiglesias@udc.gal

casos tratar este tipo de imágenes directamente en el punto de recogida de los datos para poder ofrecer a los expertos información que facilite una toma de decisiones rápida. Es por ello que se necesitan algoritmos especialmente diseñados para explotar el paralelismo del hardware comúnmente disponible sobre el terreno. En la actualidad este podría, por ejemplo, consistir en un sistema multinúcleo provisto de una o varias GPUs. Existen en la literatura diferentes algoritmos diseñados para su ejecución en este tipo de arquitecturas paralelas, pero ninguno de ellos está basado en registro mediante detección de regiones de interés [5], [6].

En este artículo se presenta una primera implementación paralela del algoritmo HSI-MSER [4], que realiza registro de imágenes hiperespectrales basado en la detección de regiones delimitadas mediante elipses, para su ejecución en un sistema heterogéneo basado en un procesador multicore y GPU. El algoritmo ha sido desarrollado en OpenMP y CUDA para obtener una explotación eficiente de la arquitectura.

El resto del artículo tiene la siguiente estructura. La sección II explica las bases de este trabajo y el algoritmo original HSI-MSER. La sección III describe la implementación híbrida en CPU y GPU desarrollada. En la sección IV se pueden encontrar una evaluación de los resultados conseguidos, analizando no solamente la eficiencia en cuanto a ahorro de tiempo de computación, sino también la calidad en cuanto a capacidad de registro de los algoritmos paralelos desarrollados. Por último, en la sección V se indican los retos alcanzados, así como vías de trabajo futuro.

II. Algoritmo original

En esta sección se describe el algoritmo HSI-MSER para el registro de dos imágenes hiperespectrales de teledetección, para luego explicar el diseño del algoritmo paralelo mediante OpenMP y CUDA.

Maximally stable extremal regions (MSER) [7] es un algoritmo de detección de características que extrae regiones que persisten al analizar la imagen con diferentes umbrales de gris. Estas regiones son invariantes a las transformaciones de la imagen, lo que las hace ideales para el registro. *Scale Invariant Feature Transform* (SIFT) [8] es un conocido detector y descriptor de puntos clave basado en la construcción de un espacio de escala gaussiano. HSI-MSER adapta MSER para su uso en imágenes hiperespectrales, utilizando una adaptación de SIFT como descriptor espacial de regiones e introduciendo un método para seleccionar qué bandas espectrales utilizar en función de su entropía, denominado *Entropy-Based Band Selection* (EBS). HSI-MSER además introduce un descriptor espectral a cada región detectada y propone un método exhaustivo para el cálculo de la transformación geométrica que al ser aplicada sobre una imagen la alinea con la otra. Todo esto ha permitido a este algoritmo aprovechar la información presente en las distintas bandas de una imagen hiperespectral para mejorar la precisión de registro.

El algoritmo HSI-MSER está compuesto por 6 fases, como se puede ver en la Figura 1, que son explicadas a continuación.

Selección de bandas. Las imágenes hiperespectrales están formadas por cientos de bandas contiguas que, dada su proximidad en cuanto a la longitud de onda asociada a ellas, contienen información redundante en algunos casos. Habitualmente se aplica un método de extracción de las bandas estadísticamente más relevantes que permita realizar el registro sobre un número de ellas que no supere la decena. Tal como se puede observar en la Figura 1, la primera fase del algoritmo consiste en aplicar el método de selección de bandas llamado *Entropy-Based Band Selection* (EBS) [4]. Este método tiene la peculiaridad de tener en cuenta las dos imágenes, en lugar de escoger las bandas de cada imagen por separado. EBS selecciona las N bandas con mayor entropía que están separadas por al menos D bandas, cuando estas están ordenadas por longitud de onda. De esta forma consigue seleccionar bandas con una alta entropía, pero que también difieren sustancialmente en longitud de onda.

Detección de regiones. En esta segunda fase, se analizan cada una de las bandas escogidas en ambas imágenes en busca de regiones de interés. Para ello se usa una adaptación del algoritmo MSER [7]. Este algoritmo procesa todos los píxeles de la imagen en orden de intensidad y busca regiones que mantengan su forma al ir aumentando el umbral de nivel de gris. De esta forma se obtienen regiones que existen en distintas bandas espectrales. Después, estas regiones son definidas como elipses.

Descripción de las regiones. En esta fase se describen las regiones que se detectaron durante la fase anterior. A cada región se le asigna un descriptor, es decir, una secuencia de números que resumirá distintas características de la misma. Las mismas regiones, pero extraídas en diferentes imágenes y bajo cambios de iluminación o transformaciones geométricas distintas, deben dar lugar a un descriptor similar. Esto nos permitirá emparejar las regiones de ambas imágenes en la siguiente fase. HSI-MSER propone un descriptor compuesto por información tanto espacial como espectral.

El algoritmo SIFT [8] es usado para construir la parte espacial del descriptor. Como SIFT fue diseñado para describir puntos y no regiones, en HSI-MSER se emplea una adaptación que tiene en cuenta el área de cada región. Para computar el descriptor primero se calculan los ángulos dominantes de cada región usando el gradiente de su superficie y una ventana circular con ponderación gaussiana. Una vez se ha determinado el ángulo dominante se puede construir el descriptor, compuesto por 128 parámetros. Estos valores se normalizan para reducir la influencia de cambios en la iluminación o el brillo.

Además de este descriptor espacial, también se tiene en cuenta la información espectral de cada región. Para ello se usa la firma espectral del centro de la elipse de todas las bandas, no solo de las escogidas durante la selección de bandas.

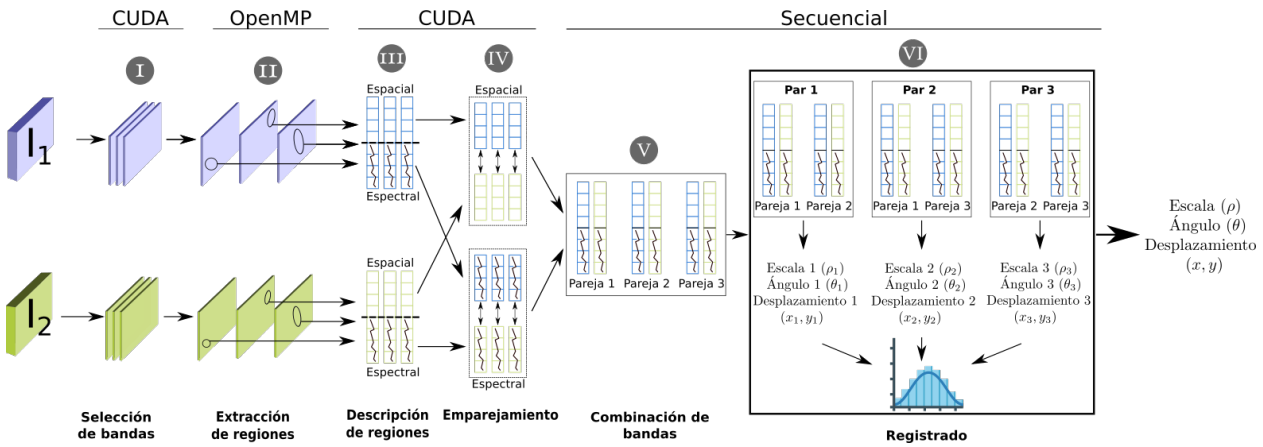


Fig. 1: Plataforma de computación paralela utilizada en cada una de las fases del algoritmo HSI-MSER. Adaptado de [4].

Emparejamiento de regiones. Una vez que las regiones han sido descritas, tratamos de encontrar parejas entre las regiones de ambas imágenes. Estos emparejamientos se hacen de forma independiente para cada banda de la imagen, pero se tiene en cuenta la información espectral del resto de bandas, como se explicó en la subsección anterior. Para calcular la similitud entre dos regiones que determina si dos regiones están emparejadas se usa la distancia euclídea para la parte espacial y la similitud coseno para la parte espectral. Únicamente se considera un emparejamiento si estas distancias están dentro de unos parámetros establecidos [4].

Combinación de las bandas. En este paso se combinan todas las parejas de regiones provenientes de todas las bandas. Esto permite usar en el registro regiones que solo aparecen en algunas frecuencias del espectro.

Registro. Para registrar las imágenes, HSI-MSER realiza una búsqueda exhaustiva basada en histogramas [4]. Para cada pareja se computan el ángulo, la escala y el desplazamiento correspondientes, tal y como se puede ver en la Figura 1. Los distintos ángulos se guardan en un histograma que representa los 360°, divididos en 72 contenedores con un tamaño base de 5° y un solapamiento de 2.5° a cada lado, llevando el total a 10°de tamaño. Una vez que este histograma se ha construido, las escalas del contenedor con más elementos se ordenan para obtener la pareja o parejas que representan la mediana. Estas parejas de regiones son las que se emplean para calcular la transformación geométrica que nos permite alinear las imágenes hiperespectrales.

III. Algoritmo paralelo propuesto para registro mediante HSI-MSER

En esta sección, se presenta la implementación paralela de HSI-MSER en CUDA y OpenMP para explotar arquitecturas heterogéneas. Se ha escogido para cada fase del algoritmo el paradigma de computación más adecuado con el fin de maximizar rendimiento.

En el Algoritmo 1 se presenta el pseudocódigo de la implementación paralela de HSI-MSER. Para cada fase, se indica con < y > que partes han sido parale-

lizadas en CUDA y con # las partes que lo han sido con OpenMP. A continuación se explican los detalles asociados a cada fase.

Selección de bandas. Esta fase se paralelizó en la GPU usando el modelo de programación CUDA. La naturaleza completamente paralela del cálculo de entropía hace que este proceso sea realizable en la arquitectura de una GPU de manera eficiente [9]. Para el cómputo de la entropía de cada banda (línea 1, Alg. 1) se usan varias funciones de la biblioteca CUB [10]. En concreto, `DeviceReduce::Min` y `DeviceReduce::Max` se utilizan para calcular de forma eficiente los valores máximo y mínimo de cada banda. Una vez se tienen estos valores, se usa `DeviceHistogram::HistogramEven` para calcular el histograma. Por último, se realiza una reducción de los valores del histograma, haciendo uso de las operaciones a nivel de *warp* para mejorar su rendimiento. Cuando ya se tienen los valores para la entropía de cada banda, se procede a seleccionar las bandas en CPU (línea 2, Alg. 1). Para ello se ordenan las bandas y se procesan siguiendo lo explicado en la Subsección II.

Extracción de regiones. MSER se basa en el algoritmo *Union find*. Este algoritmo tiene una complejidad de $O(m \cdot \alpha(m, n))$, donde n es el número de elementos, m es el número de operaciones de búsqueda que se necesitarán y α es la función inversa de Ackerman [11]. m es mayor o igual que n, pero tiene una relación de carácter lineal ($m = kn$). Esta complejidad se considera como cuasi lineal debido a la lentitud con la que crece la función inversa de Ackerman.

En la literatura ya se han publicado técnicas que se pueden aplicar para paralelizar este algoritmo [12], incluso en la GPU [13], [14]. Sin embargo, MSER, a pesar de estar basado en *Union find*, presenta varias diferencias respecto a este. Una de ellas es que los píxeles deben ser procesados por orden de intensidad creciente, lo que dificulta la paralelización de la construcción del árbol, ya que cada píxel es colocado en el árbol respecto a las posiciones de los píxeles anteriores. Si se trata de dividir la imagen para realizar la construcción del árbol en paralelo, ya sea por posición física de cada píxel o por intensidad, se consiguen árboles inconexos. En [15] se propone una

Algorithm 1 Pseudocódigo de la implementación híbrida en OpenMP y CUDA

Entrada: Imagen de referencia $\mathbf{I}_1$ e imagen objetivo $\mathbf{I}_2$.
Salida: Escala ρ, ángulo θ y desplazamiento (x, y).

Fase I. Selección de bandas.
< Cálculo de la entropía para cada banda >
Selección de las bandas con base en la entropía

para cada banda b seleccionada:
Fase II. Detección de regiones.
Realizar particiones para cada imagen según el número de hilos disponibles
Paralelizado con OpenMP
Detectar regiones por separado en cada partición

Fase III. Descripción de regiones.
< Calcular la orientación dominante de cada región >
< Computar una ventana gaussiana a partir de la banda >
< Computar el descriptor de cada región >

Fase IV. Emparejamiento de regiones.
< Calcular las distancias entre los descriptores espaciales de las regiones de ambas imágenes >
para cada región r de I_1^b:
Si La menor distancia desde r a una región de I_2^b cumple ciertos parámetros
Si La similitud espectral cumple ciertos parámetros
Guardar la pareja

Fase V. Combinación de bandas.
Combinar las parejas encontradas en distintas bandas en un único vector

Fase VI. Registro.
Computar una transformación geométrica para cada una de las parejas
Generar un histograma de ángulos
Escoger el rango de ángulos más repetido y usar la pareja mediana para el registro final

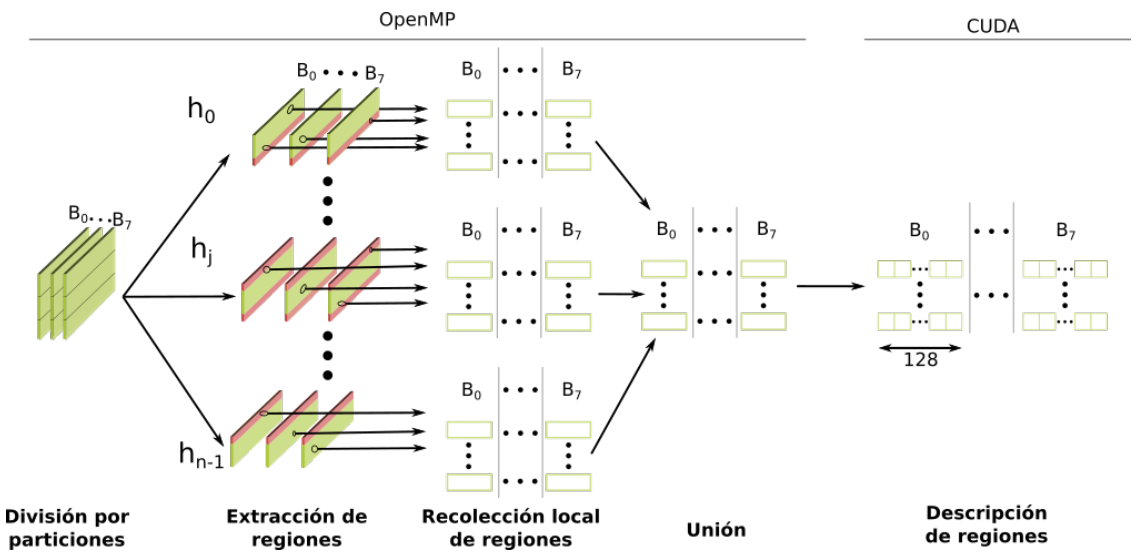


Fig. 2: Esquema de paralelización de la fase de extracción de regiones.

paralelización de MSER que fusiona estos árboles resultantes, pero que no se almacena el área de cada región, dato necesario para convertir estas regiones en elipses.

La Figura 2 muestra en detalle cómo ha sido finalmente paralelizada esta etapa. Primero, se divide la imagen horizontalmente en un número de particiones igual al número de hilos disponibles (línea 4, Alg. 1). Luego, como se puede ver en la Figura 2, se buscan las regiones en cada partición concurrentemente (línea 5, Alg. 1) haciendo uso de OpenMP [16]. Como la detección de regiones no modifica la imagen, la memoria puede ser compartida y no hay necesidad de comunicación explícita entre hilos. Cada hilo

de cómputo guarda localmente las regiones que encuentra, hasta que, finalmente, se unen todas. Para evitar que no se detecten incorrectamente regiones que estén situadas justo en los límites de una partición, se ha añadido un solapamiento entre dichas particiones. Este solapamiento se ha ilustrado en la Figura 2 con el color rojo. Un 20 % de solapamiento significa que cada hilo procesa un 120 % respecto a la versión sin solape, compartiendo un 10 % con cada uno de los hilos vecinos.

Debido a la extracción de regiones por particiones, ha sido necesario ajustar el parámetro de *min_area* de HSI-MSER para que tenga en cuenta el tamaño de la imagen completa y no solo el de la partición. Este parámetro determina cuál es el tamaño mínimo de una región para ser considerada. Su valor representa un porcentaje respecto al tamaño total de la imagen, es decir, si el valor es un 10 %, cada región tiene que tener al menos el 10 % de la imagen para ser considerada. La idea detrás de este cálculo es que, imágenes de menor tamaño es probable que tengan regiones más pequeñas. El parámetro *min_area* ha sido ajustado para que detecte regiones del mismo tamaño que si se procesara la banda completa.

Descripción de las regiones. El proceso de descripción de cada región se ha paralelizado en la GPU, tal y como aparece en la Figura 2. Para ello se han implementado tres *kernels* CUDA. En concreto, el cómputo del histograma de los ángulos (línea 6, Alg. 1), la generación de la ventana de ponderación gausiana (línea 7, Alg. 1) y el cómputo final de los descriptores (línea 8, Alg. 1). Estos *kernels* se caracterizan por una alta carga de operaciones aritméticas y trigonométricas. Además, excepto en el caso de la ventana gaussiana, es necesario el uso de instrucciones atómicas. Estos *kernels* han sido analizados y optimizados siguiendo la metodología *Assess, Parallelize, Optimize, Deploy (APOD)* [17] y haciendo uso de la herramienta de *profiling* NVIDIA Nsight Compute. Se han aplicado las optimizaciones recomendadas por NVIDIA [17] para obtener el mejor rendimiento en la GPU, por ejemplo, minimizar la transferencia de datos entre la memoria principal y GPU, escoger el número de hilos por bloque y el número de bloques óptimo por cada kernel, y maximizar la ocupancia de los multiprocesadores evitando divergencias dentro de los *warps* de cada *kernel*, entre otras.

Emparejamiento de regiones. Para el emparejamiento se ha usado una aproximación de la distancia euclídea [9]. Este método traduce el cálculo de las distancias entre las regiones de una misma banda (línea 9, Alg. 1) en operaciones matriciales. Las matrices son ideales para su cómputo en GPU, ya que se procesan repitiendo la misma operación y conforman un conjunto de datos sin dependencias. Esto nos permite obtener una gran aceleración en esta etapa. Además, se ha empleado la biblioteca cuBLAS [18] siguiendo la implementación descrita en [9]. Una vez se tienen las distancias entre los descriptores espaciales, se evalúa la distancia espectral para aquellos que cumplan ciertos parámetros establecidos por el algoritmo HSI-MSER original y se guarda la pareja si la similitud espectral también es satisfactoria (líneas 10-13, Alg. 1).

Combinación de las bandas y registro. Ambas fases se ejecutan secuencialmente en la CPU debido a su bajo coste computacional, tal y como se mostrará en la siguiente sección (líneas 14-17, Alg. 1).

IV. Resultados

En esta sección se presentan las condiciones de experimentación y las imágenes utilizadas, así como los resultados obtenidos tanto en términos de tiempos de ejecución y rendimiento como en términos de capacidad de registro.

El algoritmo ha sido ejecutado sobre un ordenador con un procesador Intel Core i7 9700k de 3.6 GHz con 8 cores, 32 GB de RAM y una GPU NVIDIA GeForce RTX 3050. Por otro lado, se han usado g++ 11.3 y nvcc 12.1 como compiladores y Ubuntu 22.04.2 LTS como sistema operativo. Las versiones de las bibliotecas de NVIDIA como CUB y cuBLAS vienen determinadas por la versión 12.1 de CUDA. Los tiempos de ejecución y aceleraciones son el resultado de promediar 10 ejecuciones independientes. Por último, se han usado las opciones O3z arch=native"para generar código optimizado para la arquitectura correspondiente. El algoritmo HSI-MSER se ha ejecutado con los valores por defecto [4] a excepción del parámetro *min_area* tal y como se explicó en la sección III.

Se han empleado 14 imágenes para realizar los experimentos de este trabajo. Las características de cada una de ellas pueden encontrarse en el artículo del método original [4] y están disponibles para su descarga en [19]. Las imágenes *Pavia University* y *Pavia Centre* han sido tomadas por el sensor *Reflective Optics System Imaging Spectrometer* (ROSIS). Las restantes han sido capturadas por el *Airborne Visible / Infrared Imaging Spectrometer* (AVIRIS) [20]. Para *Jasper Ridge*, *Santa Barbara Front*, *Santa Barbara Line*, *Baraboo Hills* y *Crown Point* se cuenta con dos imágenes que han sido capturadas en distintos años y, por lo tanto, presentan cambios de escala, rotación, desplazamiento e incluso distorsión. Nos referiremos a estas imágenes como casos reales. La escala, ángulo y desplazamientos de referencia entre estas imágenes están disponibles en [4]. Para trabajar con las imágenes sin pareja, se genera una imagen objetivo aplicándole una escala y un ángulo a la original, lo que nos permite estudiar el registro bajo condiciones controladas.

A. Tiempos de ejecución y aceleraciones

La Tabla I presenta los tiempos de ejecución de la implementación original secuencial y de la implementación paralela propuesta en este trabajo utilizando distinto número de hilos para los casos reales. La versión CUDA ejecuta la selección de bandas, la descripción de regiones y el emparejamiento en la GPU, lo que le da una ventaja considerable frente a la versión secuencial. Las versiones CUDA + OpenMP, además

Tabla I: Tiempos de ejecución (segundos) de la implementación secuencial y de la implementación paralela propuesta con distinto número de hilos en los casos reales.

Imagen	Secuencial	CUDA	CUDA + OpenMP 2	CUDA + OpenMP 4	CUDA + OpenMP 8
Santa Barbara Front	38.24	7.22	5.54	4.64	4.27
Crown Point	39.96	11.38	7.05	5.78	5.21
Jasper Ridge	50.41	12.35	7.79	5.98	5.33
Santa Barbara Line	36.69	12.00	7.65	5.91	5.14
Baraboo Hills	36.07	12.00	7.32	5.91	4.99
Media	40.27	10.45	6.26	4.86	4.22

de ejecutar las partes mencionadas en la GPU, ejecutan la detección de regiones usando 2, 4 y 8 hilos con la ayuda de OpenMP. La Figura 3 muestra las aceleraciones de las implementaciones paralelas frente la implementación secuencial.

Los resultados muestran que gracias a la paralelización de la selección de bandas, la descripción de regiones y el emparejamiento en GPU, el tiempo medio se reduce de un valor promedio de 40.27s a 10.45s. Esto hace que se consiga una aceleración promedio de 4×. Si además añadimos la paralelización de la detección de regiones en OpenMP, en el caso con 8 hilos, se consigue una aceleración cercana a 10×. Para esta versión, el tiempo promedio para registrar dos imágenes hiperespectrales es de 4.22s, lo que constituye una mejora notable respecto a los 40.27s del método secuencial.

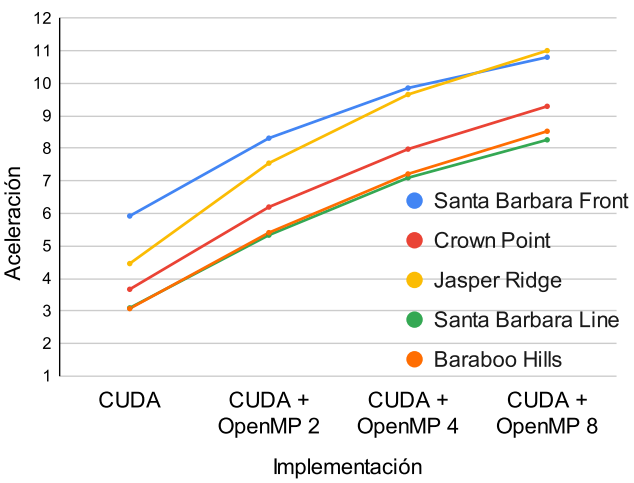


Fig. 3: Aceleraciones de la implementación paralela respecto a la secuencial en distintas imágenes. Se utilizan los tiempos de la Tabla I.

B. *Capacidad de registro*

Para comprobar cómo ha afectado la paralelización a la capacidad de registro, se han llevado a cabo varios experimentos. Se ha seguido el mismo procedimiento propuesto en el artículo original de HSI-MSER [4]. Este método se basa en ir aumentando y disminuyendo la escala original de cada imagen a la vez que se aplican rotaciones con el fin de aumentar el conjunto de imágenes de prueba. La escala se va aumentando en incrementos de 0.5× y se reduce en fracciones (1/2, 1/3, 1/4...). Para cada escala se aplica una rotación de 0 a 355°, en incrementos de 0.5°. Es decir, para cada incremento de escala estamos probando 72 rotaciones. Se considera que una escala ha sido correctamente registrada cuando se registran todos los ángulos asociados a la misma. Esto hace un total de 2592 casos probados por imagen.

La Tabla II resume el rango de escalas registradas con éxito para cada una de las imágenes utilizando las distintas versiones del algoritmo. Se indica entre paréntesis el número de escalas correctamente registradas para todos los 72 ángulos. Como se puede observar, el número de escalas correctamente registradas disminuye al aumentar el número particiones. El algoritmo secuencial es capaz de registrar una media de 19 escalas en comparación con las 10.89 de la versión más eficiente en términos de cómputo (CUDA + OpenMP 8). Al aumentar el número de hilos, aumenta el número de particiones horizontales en las que se dividen las bandas en la fase de extracción de regiones. Esta división de datos entre hilos deteriora la capacidad de detectar regiones en los bordes, reduciendo el número de emparejamientos correctos y, por tanto, impidiendo realizar un registro adecuado en casos con una gran diferencia de escala.

Habiendo estudiado los tiempos de ejecución, aceleraciones y el número de escalas correctamente registradas en comparación la implementación secuencial original, se selecciona la implementación CUDA + OpenMP 4 como una solución de compromiso entre aceleración y capacidad de registro.

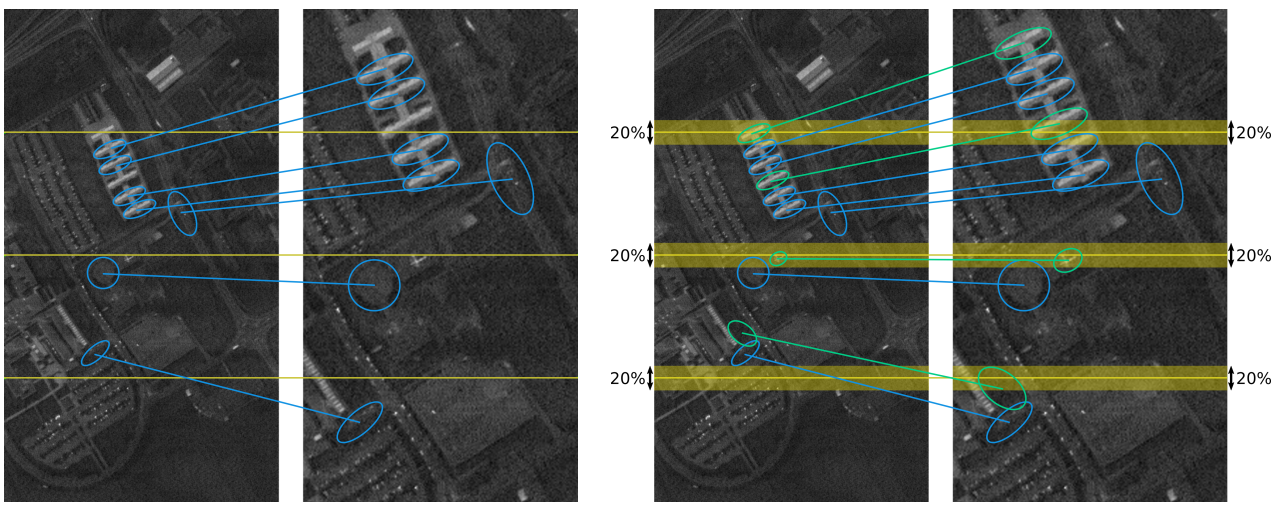
C. *Influencia de la adición de solapamiento de datos entre particiones sobre la calidad de registro*

Como se explicó en la Sección III, cada banda de la imagen se divide horizontalmente en tantas particiones como número de hilos disponibles. Cada hilo extrae regiones de forma paralela en una partición distinta. Para evitar la pérdida de regiones entre las particiones, efecto analizado en la sección anterior, se añade un solapamiento, es decir, una región frontera añadida a los datos que se asignan a cada hilo.

La Tabla III muestra el número de escalas correctamente registradas para distintos tamaños de solapamiento (0 %, 10 %, 20 % y 30 % respecto al tamaño de la partición asignada a cada hilo) con la implementación CUDA + OpenMP 4, que fue seleccionada como decisión de compromiso. Se puede observar que el número de escalas correctamente registradas incrementa al aumentar el solapamiento. Sin embargo, podemos ver que un solapamiento del 30 % obtiene los mismos resultados que el de 20 %, una media de 15.44 escalas en comparación con las 13.44 obte-

Tabla II: Capacidad de registro. Rango y número de escalas registradas con éxito para cada imagen para las distintas implementaciones del algoritmo.

Imagen	Secuencial [4]	CUDA + OpenMP 2	CUDA + OpenMP 4	CUDA + OpenMP 8
Pavia University	1/7× - 12.0× (29)	1/5× - 10.5× (24)	1/5× - 9.0× (21)	1/5× - 7.5× (18)
Pavia Centre	1/8× - 15.0× (36)	1/8× - 13.0× (32)	1/7× - 10.0× (25)	1/6× - 7.5× (19)
Indian Pines	1/3× - 8.0× (17)	1/3× - 6.5× (14)	1/2× - 5.0× (10)	1/1× - 4.0× (7)
Salinas	1/5× - 7.0× (17)	1/5× - 6.5× (16)	1/4× - 7.0× (16)	1/3× - 5.5× (12)
Jasper Ridge	1/3× - 7.0× (15)	1/2× - 3.5× (7)	1/2× - 3.0× (6)	1/2× - 3.0× (6)
Santa Barbara Front	1/3× - 7.0× (15)	1/2× - 7.0× (14)	1/2× - 5.5× (11)	1/2× - 5.5× (11)
Santa Barbara Line	1/5× - 7.0× (17)	1/4× - 6.0× (14)	1/4× - 5.0× (12)	1/3× - 4.0× (9)
Baraboo Hills	1/2× - 4.0× (8)	1/1× - 3.5× (6)	1/1× - 4.0× (7)	1/1× - 3.0× (5)
Crown Point	1/7× - 6.0× (17)	1/5× - 6.0× (15)	1/4× - 5.5× (13)	1/4× - 4.5× (11)
Media	19.00	15.78	**13.44**	10.89



(a) Sin solapamiento. (b) 20 % de solapamiento.

Fig. 4: Influencia de la adición de solapamiento en la detección de regiones en *Pavia University* con 4 particiones.

nidas sin solapamiento. Es por ello que se selecciona la implementación CUDA + OpenMP 4 con 20 % de solapamiento.

Los efectos del solapamiento se muestran en la Figura 4. En esta figura se han representado varias parejas de regiones sobre la imagen de *Pavia University* con una diferencia de escala de 1.75×. El área abarcada por el solapamiento está marcada en amarillo, en este caso, un 20 %. En la figura se representa cómo ciertas regiones, pintadas en color verde, no serían detectadas sin la adición de solapamiento al quedar divididas por las separaciones entre particiones.

D. Número de parejas y regiones

En la sección anterior hemos visto como el solapamiento nos permite mejorar el número de escalas correctamente registradas, haciendo que la implementación paralela obtenga resultados más próximos a la implementación secuencial en cuanto a capacidad de registro de escalas. En esta sección se analizan los motivos de esta mejora.

En la Tabla IV se comparan la implementación secuencial y las implementaciones seleccionadas, la implementación CUDA + OpenMP 4 con y sin solapamiento del 20 %, en base a diferentes métricas. Para esta comparación se ha registrado cada caso real sin aplicar ningún escalado o rotación adicional. En la primera fila de cada imagen se muestra el número de parejas de regiones encontradas durante la fase de emparejamiento. Se puede observar, que el número de parejas disminuye entre la versión secuencial y la versión con cuatro particiones. Esto es esperable, ya que como se explicó anteriormente, las particiones pueden dividir regiones en dos, dando lugar a su pérdida o dos regiones de menor tamaño, que luego pueden ser filtradas por el propio algoritmo o pueden no ser fiables. Sin embargo, se puede ver que con un 20 % de solapamiento se consigue recuperar gran parte del número de parejas de la versión secuencial. Este mismo efecto lo vemos en la segunda y tercera fila, donde se muestran el número de regiones extraídas para cada imagen. El mayor número de regiones obtenido en algunas imágenes se debe a que la misma o similar región es obtenida en las diferentes particiones debido al solapamiento.

En la cuarta fila, se muestra el porcentaje de parejas correctas. Para calcular esta métrica, se mide la distancia entre las regiones que conforman cada pareja tras aplicarle a la región de la imagen a registrar la transformación geométrica de referencia [4]. De esta manera, si la distancia tras aplicar la escala, ángulo y desplazamientos de referencia es mayor que 2 píxeles, la pareja es considerada como incorrecta. Se puede ver que el porcentaje de parejas correctas

Tabla III: Capacidad de registro. Rango y número de escalas registradas con éxito para cada imagen para distintos valores de solapamiento en la versión CUDA + OpenMP 4.

Imagen	0 %	10 %	20 %	30 %
Pavia University	1/5× - 9.0× (21)	1/4× - 10.0× (22)	1/4× - 11.0× (24)	1/4× - 11.0× (24)
Pavia Centre	1/7× - 10.0× (25)	1/7× - 10.5× (26)	1/6× - 11.5× (27)	1/6× - 11.5× (27)
Indian Pines	1/2× - 5.0× (10)	1/2× - 5.0× (10)	1/2× - 6.5× (13)	1/2× - 6.5× (13)
Salinas	1/4× - 7.0× (16)	1/4× - 7.0× (16)	1/4× - 7.0× (16)	1/4× - 7.0× (16)
Jasper Ridge	1/2× - 3.0× (6)	1/2× - 5.0× (10)	1/2× - 5.0× (10)	1/2× - 5.0× (10)
Santa Barbara Front	1/2× - 5.5× (11)	1/2× - 7.0× (14)	1/3× - 7.0× (15)	1/3× - 7.0× (15)
Santa Barbara Line	1/4× - 5.0× (12)	1/4× - 6.0× (14)	1/4× - 6.5× (15)	1/4× - 6.5× (15)
Baraboo Hills	1/1× - 4.0× (7)	1/1× - 4.0× (7)	1/1× - 4.0× (7)	1/1× - 4.0× (7)
Crown Point	1/4× - 5.5× (13)	1/3× - 6.0× (13)	1/3× - 5.5× (12)	1/3× - 5.5× (12)
Media	13.44	14.67	**15.44**	15.44

está, por lo general, en torno al 75 %. Este porcentaje mejora ligeramente al añadir solapamiento, indicando que se recuperan más regiones correctas que incorrectas con su adición. En el caso de *Crown Point*, el porcentaje de parejas correctas es considerablemente menor debido a la distorsión no lineal que presenta. Para poder evitar esto, se necesitan métodos capaces de calcular transformaciones geométricas con mayores grados de libertad que escala, ángulo y desplazamiento.

En la quinta fila se indica el tiempo de ejecución total (media de 10 ejecuciones del registro). Por último, se ha añadido el número de escalas que se registran para cada uno de estos casos reales si se varían escalas y ángulos de manera exhaustiva. Al comparar las distintas implementaciones, se puede observar que existe una relación clara entre el número de parejas detectadas y el número de escalas registradas. Cuando la diferencia de escala es demasiado grande, el registro falla por no contar con parejas para calcular la transformación o porque las pocas parejas obtenidas son incorrectas. Gracias al solapamiento del 20 %, se recuperan regiones y parejas que se encontraban en los bordes de las particiones, consiguiendo aumentar el número de escalas registradas.

No obstante, la versión con solapamiento requiere de un mayor tiempo de ejecución, ya que al añadirle un solapamiento a cada partición, estamos aumentando el espacio donde debemos buscar regiones. Por consiguiente, tal y como se ha explicado, se obtiene un mayor número de regiones, que deriva en un aumento del tiempo de cómputo de las siguientes etapas (descripción, emparejamiento, combinación y registro). En el caso más desfavorable, para las imágenes *Jasper Ridge*, se necesitan 5.72s frente a los 5.22s de la versión sin solapamiento, pasando de registrar 10 a 6 escalas, respectivamente. Si lo comparamos con la versión secuencial, esta necesita 43.95s.

E. Aceleración por etapas

Las Tablas V y VI presentan los tiempos de ejecución y aceleraciones para las implementaciones secuencial y la seleccionada (CUDA + OpenMP 4 con 20 % de solapamiento) para el caso real de mayor tamaño (*Baraboo Hills*) y para el de menor tamaño (*Santa Barbara Front*). Como se puede observar, las fases más costosas son la detección, descripción y emparejamiento de regiones. En ellas es donde se ha conseguido mejores aceleraciones, destacando especialmente la fase de emparejamiento que consigue una aceleración próxima a 21×. La aproximación de la distancia euclídea en cálculos matriciales para el emparejamiento de las regiones, nos permite explotar adecuadamente el modelo de paralelismo y arquitectura de la GPU. Las otras dos fases que se ejecutan en GPU, selección de bandas y descripción de regiones, han obtenido aceleraciones de hasta 2.68× y 16.19×, respectivamente. La fase de detección de regiones está paralelizado usando OpenMP y consigue una aceleración de hasta 4.41× usando 4 hilos y solapamiento del 20 %. En el cómputo total, se obtiene una aceleración de hasta 7.87×. Recordemos que se podría conseguir una aceleración mayor aumentando el número de hilos que se usan en la detección en el caso de que no fuera necesario registrar este tipo de escalas.

Si comparamos los resultados entre ambas imágenes, se puede ver que en la detección de regiones se produce una aceleración mayor en *Baraboo Hills*, debido a su mayor tamaño, que mejora la eficiencia de la paralelización en esa fase. Sin embargo, en la descripción pasa justo lo contrario. Esto se debe a que *Santa Barbara Front*, a pesar de ser la imagen más pequeña de las utilizadas, se obtienen una gran cantidad de regiones (ver Tabla IV). Este desacoplamiento entre el tamaño y el número de regiones que se detectan es lo que produce que *Santa Barbara Front* sea de las imágenes que más aceleración experimentan, como se vio en la Figura 3, a pesar de ser la más pequeña.

V. Conclusiones

En este artículo se ha presentado una versión paralela del algoritmo HSI-MSER de registro de imágenes hiperespectrales basado en características. Este algoritmo adapta MSER y SIFT para su uso en imágenes hiperespectrales, aprovechando la información espectral de forma eficiente.

La implementación propuesta paraleliza el algoritmo haciendo uso los distintos núcleos disponibles en la CPU así como de la GPU, obteniendo una considerable reducción de los tiempos de ejecución cuando la comparamos con la versión secuencial original. La implementación propuesta ha sido evaluada con

Tabla IV: Número de parejas de regiones, porcentaje de parejas correctas, número de regiones, número de escalas registradas correctamente y tiempos de ejecución de la versión secuencial y CUDA + OpenMP 4 sin solapamiento y con 20 % de solapamiento.

Imagen		Secuencial	CUDA + OpenMP 4	CUDA + OpenMP 4 + 20 %
Jasper Ridge	Parejas	253	187	236
	Regiones imagen 1	32538	33379	37879
	Regiones imagen 2	32625	33387	38128
	Porcentaje de parejas correctas	0.742	0.778	0.786
	Tiempo (s)	43.95	5.22	5.72
	Número de escalas registradas	15	6	10
Santa Barbara Front	Parejas	618	531	630
	Regiones 1	27601	28578	32843
	Regiones 2	31688	32618	37379
	Porcentaje de parejas correctas	0.727	0.732	0.756
	Tiempo (s)	33.70	3.89	4.28
	Número de escalas registradas	15	11	15
Santa Barbara Line	Parejas	977	825	988
	Regiones 1	24420	25335	28861
	Regiones 2	25703	26812	30261
	Porcentaje de parejas correctas	0.773	0.764	0.774
	Tiempo (s)	33.78	5.17	5.64
	Número de escalas registradas	17	12	15
Baraboo Hills	Parejas	133	110	137
	Regiones 1	24413	25317	28876
	Regiones 2	22223	23077	26261
	Porcentaje de parejas correctas	0.756	0.771	0.769
	Tiempo (s)	32.83	5.00	5.38
	Número de escalas registradas	8	7	7
Crown Point	Parejas	693	572	704
	Regiones 1	27937	28764	33281
	Regiones 2	27412	28283	32607
	Porcentaje de parejas correctas	0.219	0.24	0.24
	Tiempo (s)	36.11	5.01	5.49
	Número de escalas registradas	17	13	12

Tabla V: Tiempos de ejecución en segundos y aceleración de la versión CUDA + OpenMP 4 con un 20 % de solapamiento respecto a la versión secuencial para *Baraboo Hills*.

Fase	Secuencial (s)	CUDA + OpenMP 4 + 20 % (s)	Aceleración
Leer las imágenes	0.92	0.92	—
Copiar imágenes a GPU	—	0.28	—
Selección de bandas	0.32	0.12	2.68×
Detección de regiones	8.29	1.88	4.41×
Descripción de regiones	14.99	1.02	14.66×
Emparejamiento	8.30	0.40	20.94×
Registro	0.01	0.01	—
Total	32.83	5.38	6.10×

imágenes accesibles públicamente para las que hay datos disponibles en la literatura. Para la evaluación se han tenido en cuenta tanto la aceleración respecto a la implementación secuencial original como la capacidad de registro. Se ha comprobado que, si bien al ejecutar la versión paralela del registro se producen

Tabla VI: Tiempos de ejecución en segundos y aceleración de la versión CUDA + OpenMP 4 con un 20 % de solapamiento respecto a la versión secuencial para *Santa Barbara Front*.

Fase	Secuencial (s)	CUDA + OpenMP 4 + 20 % (s)	Aceleración
Leer las imágenes	0.48	0.48	—
Copiar imágenes a la GPU	—	0.16	—
Selección de bandas	0.17	0.10	1.59×
Detección de regiones	3.43	1.05	3.28×
Descripción de regiones	18.59	1.15	16.19×
Emparejamiento	10.97	0.52	21.26×
Registro	0.06	0.06	—
Total	33.70	4.28	7.87×

pérdidas de capacidad de registro, especialmente para imágenes con diferencias de escala considerables, es posible mitigar este efecto mediante el solapamiento de datos entre hilos. De este modo, se consigue registrar hasta 7.87× más rápido con la versión con 4 hilos y solapamiento, en comparación con el algoritmo secuencial, sin una pérdida sustancial de capacidad de registro para diferencias de escala extremas entre imágenes.

Como trabajo futuro, se propone investigar en mayor grado la paralelización de la fase de detección de regiones. Se estudiará la posibilidad de construir el árbol basado en el algoritmo *Union Find* de forma paralela. Además, será necesario probar los algoritmos en entornos de producción reales que exigen el registro de imágenes de mayor tamaño obtenidas en mayor variedad de condiciones de captura.

Agradecimientos

El presente trabajo ha sido financiado por MCIN/AEI/10.13039/50110001103 a través de los proyectos TED2021–130367B–I00, FJC2021-046760-I y PID2019–104834GB–I00, por la Xunta de Galicia - Consellería de Cultura, Educación, Formación Profesional e Universidades mediante las ayudas 2019-2022 ED431G-2019/04 y ED431C-2022/16, y por el Fondo Europeo de Desarrollo Regional (FEDER). También se ha recibido financiación de la Junta de Castilla y León (proyecto PROPHET–II, VA226P20).

Referencias

[1] Xiaoxin Guo, Zhiwen Xu, Yinan Lu, and Yunjie Pang, "An application of Fourier-Mellin transform in image registration," in *Proceedings of the The Fifth International Conference on Computer and Information Technology*, USA, 2005, CIT '05, p. 619–623, IEEE Computer Society.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, 2006.

[3] Àlvaro Ordóñez, Dora B. Heras, and Francisco Argüello, "Comparing area-based and feature-based methods for co-registration of multispectral bands on GPU," in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021, pp. 1575–1578.

[4] Álvaro Ordóñez, Álvaro Acción, Francisco Argüello, and Dora B. Heras, "HSI-MSER: Hyperspectral image registration algorithm based on MSER and SIFT," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 12061–12072, 2021.

[5] Antonio Plaza, David Valencia, Javier Plaza, and Pablo Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing*, vol. 66, no. 3, pp. 345–358, 2006.

[6] Álvaro Ordóñez, Dora Blanco Heras, and Francisco Argüello, "Multi-GPU registration of high-resolution multispectral images using HSI-KAZE in a cluster system," *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, pp. 5527–5530, 2022.

[7] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[8] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, nov 2004.

[9] Álvaro Ordóñez, Francisco Argüello, Dora B. Heras, and Begüm Demir, "GPU-accelerated registration of hyperspectral images using KAZE features," *The Journal of Supercomputing*, vol. 76, no. 12, pp. 9478–9492, Dec 2020.

[10] "NVIDIA: CUB library," 2023, Accessed 22 May 2023.

[11] Robert Endre Tarjan, "Efficiency of a good but not linear set union algorithm," *J. ACM*, vol. 22, no. 2, pp. 215–225, apr 1975.

[12] Richard J Anderson and Heather Woll, "Wait-free parallel algorithms for the union-find problem," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 1991, pp. 370–380.

[13] Stefano Allegretti, Federico Bolelli, Michele Cancilla, and Costantino Grana, "A block-based union-find algorithm to label connected components on GPUs," in *Image Analysis and Processing–ICIAP 2019: 20th International Conference, Trento, Italy, September 9–13, 2019, Proceedings, Part II 20*. Springer, 2019, pp. 271–281.

[14] Jun Chen, Qiang Yao, Houari Sabirin, Keisuke Nonaka, Hiroshi Sankoh, and Sei Naito, "An optimized union-find algorithm for connected components labeling using GPUs," *arXiv preprint arXiv:1708.08180*, 2017.

[15] Hailiang Xu, Siqi Xie, and Fan Chen, "Fast MSER," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3380–3389.

[16] OpenMP, "The OpenMP API specification for parallel programming," `https://www.openmp.org/`, [En línea, accedido: 07-Jun-2023].

[17] "CUDA Best Practices Guide," `https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html`, [En línea, accedido: 18-Jun-2023].

[18] NVIDIA, "cuBLAS library," `https://developer.nvidia.com/cublas`, [En línea, accedido: 07-Jun-2023].

[19] Álvaro Ordóñez, Francisco Argüello, and Dora B. Heras, "Hyperspectral image repository," `https://gitlab.citius.gal/hiperespectral/RegistrationRepository`, [En línea, accedido: 07-Jun-2023].

[20] "AVIRIS database," `https://aviris.jpl.nasa.gov/dataportal`, [En línea, accedido: 07-Jun-2023].

[21] Sabyasachi Maiti and Amit K. Bhattacharya, "Shoreline change analysis and its application to prediction: A remote sensing and statistics based approach," *Marine Geology*, vol. 257, pp. 11–23, 2009.

[22] Amy Lowe, Nicola Harrison, and Andrew P. French, "Hyperspectral image analysis techniques for the detection and classification of the early onset of plant disease and stress," *Plant Methods*, vol. 13, no. 1, pp. 80, Oct 2017.