This a **post-print** of the article “Region-based multispectral image registration on heterogeneous computing platforms” published in the Proceedings of IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium.

The published article is available on https://doi.org/10.1109/IGARSS53475.2024.10641884

D. Del Castillo, Á. Ordóñez, D. B. Heras and F. Argüello, "Region-Based Multispectral Image Registration on Heterogeneous Computing Platforms," *IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium*, Athens, Greece, 2024, pp. 1008-1012, doi: 10.1109/IGARSS53475.2024.10641884.

REGION-BASED MULTISPECTRAL IMAGE REGISTRATION ON HETEROGENEOUS COMPUTING PLATFORMS

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

[1] Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Spain.
[2] Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Spain.

ABSTRACT

Feature-based methods are widely used for the registration of remote sensing images because of their robustness to viewpoint, scale, and light changes. However, they are computationally demanding, especially when dealing with multi or hyperspectral images. Hyperspectral Maximally Stable Extremal Regions (HSI-MSER) is a hyperspectral remote sensing image registration method based on MSER for feature detection and Scale Invariant Feature Transform (SIFT) for feature description. This article presents a first approach to a parallel implementation of the HSI-MSER algorithm for the registration of multispectral images on a heterogeneous computing platform. The results of the registration capabilities under extreme scaling and rotating conditions show that the proposed parallel implementation obtains a speedup of $3.33\times$ compared to the sequential implementation making it suitable for applications with execution time constraints.

***Index Terms*—** Multispectral, registration, heterogeneous computing, GPU, CUDA.

1. INTRODUCTION

Efficient alignment of remote sensing images captured by drones or satellites plays a key role in a variety of applications. This process, known as image registration, involves adjusting scale, rotation, and translation to match images from the same geographic area, crucial for tasks like change analysis over time or mosaic composition [1]. The registration of multispectral images captured by drones for mosaic composition is a computationally demanding task that involves aligning tens of high-resolution images of the same scene, sometimes even requiring co-registration of the bands within the same image. Managing the huge amount of captured data in reasonable time makes necessary the development of efficient parallel registration techniques.

HSI-MSER is a hyperspectral image registration algorithm based on a process of feature detection, particularly regions of interest [2]. By efficiently utilizing spatial and spectral information across different bands, the algorithm generates descriptors for each region detected in each of the images to be registered. The descriptors are then matched between images to determine the best alignment. Despite their good registration results for real images that present noise and other alterations, feature-based methods, including HSI-MSER, require high execution times due to the costly stages involved in the registrations: feature detection, feature description, and feature matching. This is particularly challenging in the case of hyper and multispectral images, as in these cases searching for features across all spectral bands is required. This makes it necessary to develop algorithms that are tailored to exploit the computational resources. One possible option is to exploit the parallelism available in widespread hardware, like multi-core processors and GPUs. Different implementations for image registration using GPUs or multi-core processors have already been proposed in the literature [3, 4] achieving large speedups. However, none of them are based on region detection.

This article introduces a first approach to a parallel implementation of HSI-MSER for registering two high-resolution multispectral images. The algorithm is implemented for a heterogeneous system, integrating a multi-core processor and a GPU, and utilizes OpenMP and CUDA for efficient hardware exploitation.

2. PARALLEL REGISTRATION OF MULTISPECTRAL IMAGES USING HSI-MSER

The original HSI-MSER was specially designed to register hyperspectral images. For this reason, it includes a band selection stage to select the bands that are more suitable for the registration process. However, this stage is not necessary for multispectral images as the number of bands is much lower and they are not as correlated as in hyperspectral images. For

This work was supported in part by grants TED2021–130367B–I00 and PID2022–141623NB–I00 funded by MCIN/AEI/10.13039/501100011033 and by "European Union NextGenerationEU/PRTR". It was also supported by Xunta de Galicia - Consellería de Cultura, Educación, Formación Profesional e Universidades [Centro de investigación de Galicia accreditation 2019-2022 ED431G-2019/04 and Reference Competitive Group accreditation, ED431C-2022/16], and by "ERDF/EU".

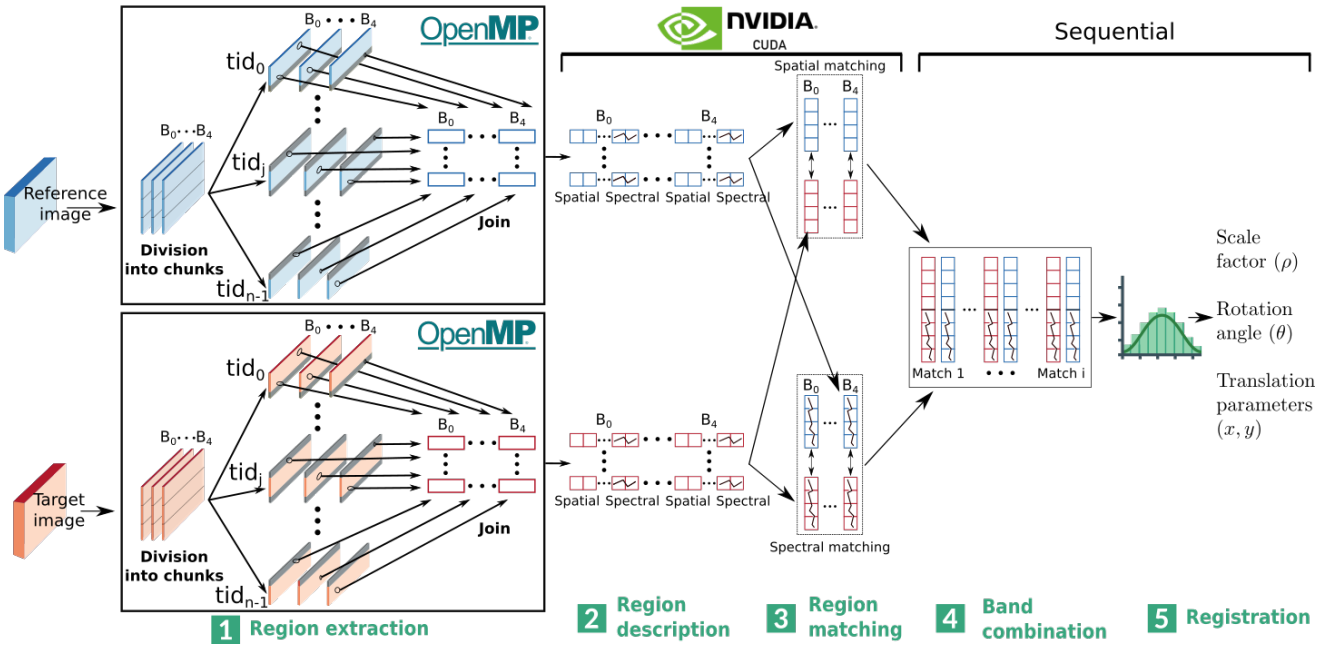


Fig. 1. Schematic of the proposed parallel implementation of HSI-MSER to register two multispectral images on a heterogeneous computing platform.

this reason, the band selection stage was removed in the implementation presented in this article.

Figure 1 outlines the proposed parallel implementation of HSI-MSER for the registration of two multispectral images [2]. The algorithm comprises five stages: (1) extracting regions, (2) describing regions, (3) matching regions, (4) band combination, and (5) registering the images.

In the first stage, the regions of interest are extracted from the images using the MSER algorithm [5]. MSER works by identifying regions that remain stable over a wide range of image intensity thresholds. To do that, a union-find data structure (implemented as a tree structure) is used to keep track of the regions as the threshold is increased. All the pixels are processed in order of intensity (from the darkest to the brightest and vice versa) and therefore each pixel is added to the tree according to the intensity and position of the pixels that have already been processed. This dependence on the order of the pixels makes the parallelization of the stage challenging. If we try to split the image by intensity values or by its spatial position to build the global tree in parallel, we will get disjoint trees. For this reason, this stage is implemented by carrying out parallel MSER detections on different chunks of the image and joining the results.

Figure 1 illustrates how this is performed. First, the image is divided into chunks of equal size. In each chunk, independent MSER detections are performed in parallel by n threads running on n cores using OpenMP (tid_{n-1} in Figure 1). To avoid losing regions that are located in the boundaries of the chunks, overlapping areas with the neighbouring chunks (in grey color in Figure 1) are added. Since region detection does not modify the image, memory can be shared and there is no need for explicit communication between threads. Each thread stores locally the regions it extracts. Finally, the regions of each band are joined together to form the final set of regions.

The extracted regions are then described using a descriptor consisting of a spatial and a spectral part. The spatial descriptor is computed using the SIFT [6] algorithm, while the spectral one is the spectral signature of the region. This stage has been parallelized using CUDA, particularly three main kernels have been implemented: (1) orientation histograms, (2) Gaussian blur, and (3) final descriptor. All of them involve a high number of arithmetic and trigonometric operations that are very efficiently computed in the GPU.

In the third stage, regions from both images are matched on GPU using an approximation of the Euclidean distance [4]. This approximation allows to transform the calculation of the distances between regions of the same band into matrix operations, which have been efficiently parallelized on a GPU using the cuBLAS library following the approach proposed in [4]. After obtaining the distances between spatial descriptors, the cosine similarity between the spectral descriptors of the regions is computed. Finally, the regions are matched if both the spatial distance and the spectral similarity are below

a threshold [2].

The fourth and fifth stages are executed sequentially on the CPU as they do not involve high computational costs. In the fourth stage, the matches of the regions from all bands are combined into a single set of matches in order to use all the information available in the last stage. Finally, in the fifth stage, an exhaustive search is performed to find the best registration parameters (scale, rotation, and translation) considering all the possible pairs of matches [2].

3. RESULTS

The experiments were carried out on a PC with a quad-core Intel i7-4790 CPU at 3.60 GHz and 24 GB of RAM. The sequential code was written in C and C++, and compiled using the gcc and the g++ 10.5.0 versions with the O3 optimization level enabled under Ubuntu 22.04. The parallel implementations were developed using the CUDA 12.3 toolkit. The GPU used was an NVIDIA GeForce RTX 2070 with 8 GB of memory. The results of computation times and speedup provided correspond to the average of ten independent executions.

Three different datasets named *Reservoir*, *Parasol*, and *House* were considered for the tests[1]. Each dataset consists of two images of 1280×960 pixels and 5 bands captured by a MicaSense RedEdge MX sensor in 2018 in different UAV flights over Galician river basins (Spain). Consequently, the images exhibit variations in scale, rotation, translation, perspective, and distortions.

The experiments consist of registering the images of each dataset using the different implementations of the algorithm. To evaluate the registration capabilities under extreme conditions, a scaling factor between $1/2\times$ and $5.5\times$ is applied to the images. For each factor, the images are rotated by 72 different angles between 0° and 355° with a step of 5°. This allows us to increase the dataset size.

Table 1 presents the range of correctly registered scale factors for each image pair after applying the 72 different angles and the execution times using the sequential and parallel implementations of the HSI-MSER algorithm. The number of correctly registered scale factors is shown in parentheses. The CUDA version executes in parallel only the stages implemented on the GPU (region description and matching), while the second and third versions also parallelize the region extraction stage using OpenMP with 4 and 8 threads, respectively. The results show that the CUDA version obtains the same results as the sequential version, while the CUDA + OpenMP 4 and CUDA + OpenMP 8 versions obtain a slightly lower number of correctly registered scale factors. The reason for this decrease is that as the number of threads increases, so does the number of chunks into which the image is divided. As a consequence, the number of regions extracted decreases as we are losing regions located in the boundaries of the chunks. This can be mitigated by adding an overlapping area between chunks as illustrated in Figure 2. As a result, regions located in the boundaries of the chunks are recovered (in pink color).

The last two columns of Table 1 present the results obtained by adding an overlapping area of 20% and 30% between chunks. These show that the overlapping area allows us to recover regions located in the boundaries of the chunks, and therefore, the number of correctly registered scale factors increases, particularly 5.3 to 6.3 scales on average.

The second row of Table 1 shows the execution times needed to register each pair of images using the corresponding implementation. The figures indicate that the best results are obtained with the CUDA + OpenMP 8 implementation, which takes 9.85 seconds on average compared to 42.92 seconds for the sequential implementation. As can be seen in the penultimate column, adding an overlapping area of 20% between chunks increases the execution time by almost 2 seconds on average. This is because the overlapping area increases the number of regions extracted. This results in a higher computational cost but also a higher number of correctly registered scale factors.

Finally, Table 2 shows the execution times for each stage of the registration process using the sequential and the CUDA + OpenMP 8 + 20% implementations for one of the datasets, in particular, the *House* dataset. The results show that the region extraction and description stages are the most time-consuming stages. A speedup of $2.2\times$ and $20.47\times$ is obtained for these stages, respectively. On average, the speedup obtained for the whole process is $3.33\times$.

4. CONCLUSIONS

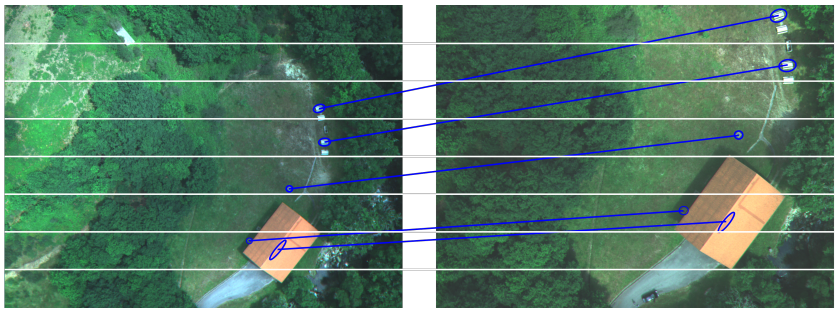
In this work, a first approach to a parallel implementation of the HSI-MSER algorithm for the registration of high-resolution multispectral images has been presented. The algorithm has been implemented on a heterogeneous system using OpenMP and CUDA. The results show that the proposed parallel implementation achieves a speedup of $3.33\times$ on average compared to the sequential implementation at the cost of lower registration capacity. The results also demonstrate that this effect can be mitigated by adding an overlapping area between chunks in the region extraction stage.

As future work, we plan to further investigate the parallelization of the region detection stage, exploring the parallel construction of the tree to mitigate the loss of regions located in the boundaries of the chunks.

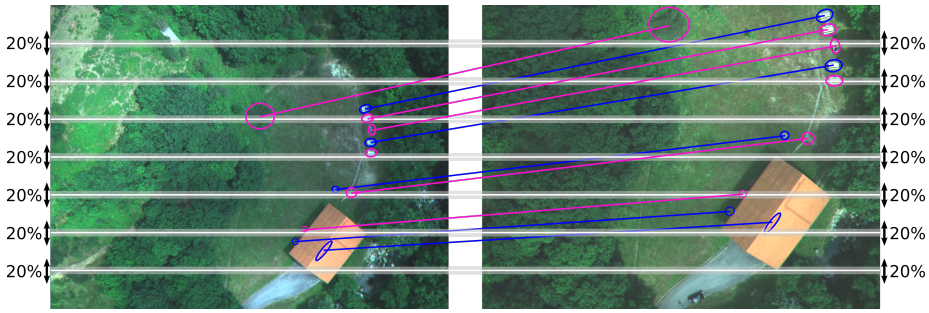
[1]These images were obtained in partnership with the Babcock company, supported in part by the Civil Program UAVs Initiative, promoted by the Xunta de Galicia.

Table 1. Range of correctly registered scale factors after applying 72 different angles and execution times (in seconds) for each pair of images using the sequential and parallel implementations of the HSI-MSER algorithm.

Dataset	Sequential	CUDA	CUDA + OpenMP 4	CUDA + OpenMP 8	CUDA + OpenMP 8 + 20%	CUDA + OpenMP 8 + 30%
Reservoir	1/2× to 3.0× (6) 44.95s	1/2× to 3.0× (6) 36.61s	1.0× to 3.0× (5) 13.52s	1/2× to 2.5× (5) 9.92s	1/2× to 3.0× (6) 12.18s	1/2× to 3.0× (6) 12.81s
Parasol	1/2× to 4.5× (9) 39.95s	1/2× to 4.5× (9) 31.87s	1/2× to 4.5× (9) 12.95s	1/2× to 3.5× (7) 9.15s	1/2× to 3.5× (7) 10.92s	1/2× to 3.5× (7) 11.46s
House	1.0× to 5.5× (10) 42.95s	1.0× to 5.5× (10) 33.72s	1.0× to 3.0× (5) 14.27s	1.0× to 2.5× (4) 10.49s	1.0× to 3.5× (6) 12.90s	1.0× to 3.5× (6) 13.79
Average	(8.3) 42.92s	(8.3) 34.07s	(6.3) 13.61s	(5.3) 9.85s	(6.3) 12.00s	(6.3) 12.69s



(a) Without overlapping



(b) 20% of overlapping

Fig. 2. Example of the recovery of regions by adding an overlapping area between chunks in the *House* images.

Table 2. Execution times (in seconds) for each stage of the registration process using the sequential and parallel implementations of the HSI-MSER algorithm for the *House* dataset.

Stage	Sequential (s)	CUDA + OpenMP 8 +20% (s)	Speedup
Read images	0.04	0.04	-
Copy images to the GPU	-	0.13	-
Region extraction	25.35	11.52	2.20×
Region description	15.12	0.74	20.47×
Region matching	2.45	0.47	5.16×
Registration	0.01	0.01	-
Total	42.96	12.91	3.33×

5. REFERENCES

[1] Barbara Zitova and Jan Flusser, "Image registration methods: a survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[2] 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.

[3] Jorge Fernández-Fabeiro, Arturo Gonzalez-Escribano, and Diego R Llanos, "Distributed programming of a hyperspectral image registration algorithm for heterogeneous GPU clusters," *Journal of Parallel and Distributed Computing*, vol. 151, pp. 86–93, 2021.

[4] 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, 2020.

[5] 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.

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