This a **post-print** of the article “Multi-GPU registration of high-resolution multispectral images using HSI-KAZE in a cluster system” published in the Proceedings of IGARSS 2022 – 2022 IEEE International Geoscience and Remote Sensing Symposium.

The published article is available on https://doi.org/10.1109/IGARSS46834.2022.9884717

Á. Ordóñez, D. B. Heras and F. 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*, Kuala Lumpur, Malaysia, 2022, pp. 5527-5530, doi: 10.1109/IGARSS46834.2022.9884717.

MULTI-GPU REGISTRATION OF HIGH-RESOLUTION MULTISPECTRAL IMAGES USING HSI-KAZE IN A CLUSTER SYSTEM

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 registration methods have been demonstrated to be very effective to register multispectral images with large distortions or transformations despite the higher execution time that they require. In this paper, a first approach to a multi-node, multi-GPU implementation of the Hyperspectral KAZE (HSI-KAZE) method for co-registering bands and multispectral images is presented. Different multispectral datasets are distributed among the available nodes of a cluster using MPI and exploiting the parallel stream-based capabilities of the GPUs inside each node using CUDA.

***Index Terms*—** Multispectral, registration, MPI, GPU, CUDA.

1. INTRODUCTION

Image registration is a common task for preprocessing remote sensing images with the objective of aligning them prior to further processing associated, for example, change detection, environmental monitoring, anomaly detection, or mosaic composition, among other possible applications.

The registration algorithms can be classified into area-based and feature-based methods [1]. Feature-based methods extract distinctive and common features that can be used to compute the geometric transformation. Hyperspectral KAZE (HSI-KAZE) [2] is a feature-based method specially tuned for the registration of hyperspectral images. On the other hand, area-based methods use pixel values to compute the differences between images without constructing an explicit correspondence between them. These characteristics make feature-based methods more suitable when there are large distortions or transformations except for the higher computational time they require [1].

This work was supported in part by Ministerio de Ciencia e Innovación, Government of Spain [grant number PID2019-104834GB-I00], and Consellería de Cultura, Educación e Universidade [grant numbers ED431C 2018/19, and accreditation 2019-2022 ED431G-2019/04]. This work was also supported by Junta de Castilla y León (PROPHET-2 Project) [grant number VA226P20]. All are co-funded by the European Regional Development Fund (ERDF).

To alleviate the high computational cost of the registration process, different feature-based algorithms have been proposed for their execution in specialised hardware such as GPUs [2, 3, 4]. Nevertheless, GPU does not provide enough computational resources if the size or number of datasets to be registered is high. In [5], we presented a first approach to a multi-GPU scheme for a two-level registration of 5-band multispectral images. In the first level, the bands of each multispectral image were registered (co-registration), and in the second one, the different multispectral images belonging to the same dataset were registered to build a scene. As this requires a high number of registrations per dataset, this proposal registered two separate datasets in parallel using a single shared-memory machine with two GPUs, i.e., each GPU performed both levels of registration (co-registration and building a scene) on a different multispectral dataset. However, the parallelism at the dataset level was limited, since each GPU performed both levels sequentially [5].

In this paper, a first approach to a multi-node and multi-GPU implementation of the HSI-KAZE method to register a large number of multispectral datasets is proposed. The datasets are cyclically distributed among the available nodes of a cluster using MPI. Within each node, the registration of bands and then the registration of the different multispectral images are also performed in parallel thanks to the exploitation of the available GPUs of the node using OpenMP and CUDA.

2. MULTI-NODE AND MULTI-GPU REGISTRATION OF REMOTE SENSING MULTISPECTRAL IMAGES

In this section, the details of the implementation based on the distribution of data among the nodes of a cluster using MPI and the execution of the registration processes on the GPUs of each node using CUDA is presented.

At the beginning, different datasets made up of 5-band multispectral images that we call frames are stored in the disk of the master node. Two levels of registration have to be performed for each dataset. First, the bands that form each multispectral image must be co-registered. Finally, the resulting

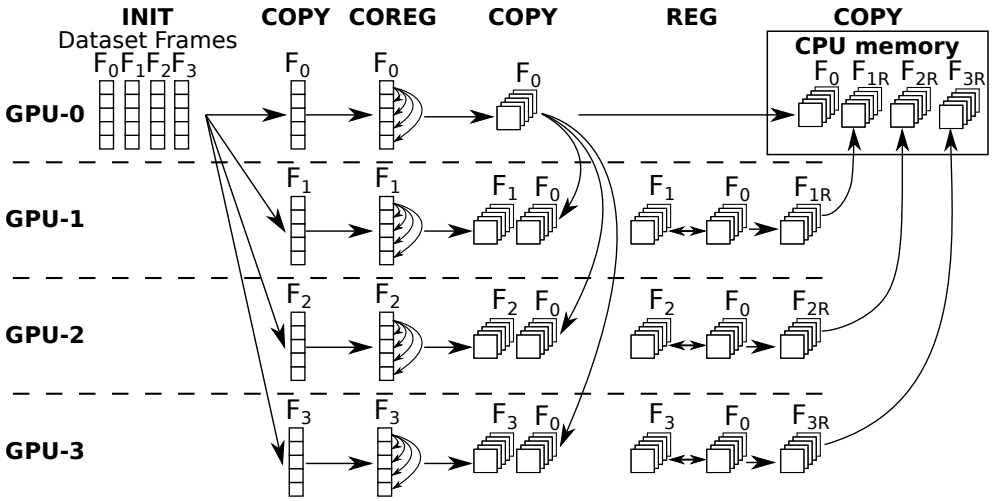

Fig. 1. Schematic of the proposed multi-node and multi-GPU parallel approach for the registration of remote sensing multispectral images. Example for one of the nodes.

images belonging to the same dataset must be registered to build a scene. The proposal is to perform both levels of registration in the same node for each dataset. So, as a first stage, the master node distributes the datasets among the available nodes in the cluster using MPI. It is important to note that the master node also processes datasets.

Figure 1 shows the proposed registration implementation that is executed on each node. Each node has several GPUs available, four GPUs in our experimental setup. Once the node has received a dataset, each frame (a 5-band multispectral image) is distributed across the available GPUs. In the figure, one frame is assigned to each GPU. In case the number of frames is higher than GPUs, they will be cyclically distributed. Thus, each frame is copied from the host memory to the corresponding GPU memory (first COPY in Figure 1). A thread is created for each GPU to start co-registering the bands of each frame in parallel using HSI-KAZE on GPU [2]. The 1st band of each frame is used as the reference band, i.e. the other four bands are the target bands, which are registered with respect to the 1st band (COREG in Figure 1).

Once all the bands of all the frames have been registered, F_0 (the reference frame) is copied from the GPU-0 memory to the memory of the other GPUs (second COPY in Figure 1). Next, the registration of the different frames with respect F_0 can begin. Again, each GPU computes the registration of a frame in parallel (REG in Figure 1). For this purpose, HSI-KAZE selects the band with the highest entropy to carry out the registration. Then, each registered frame is copied back from the different GPU memories to the host memory (last COPY in Figure 1). Finally, the slave node sends all the registered frames of the dataset to the master node using MPI.

3. RESULTS

In this section, the proposed implementation of HSI-KAZE on multiple nodes and GPUs is analysed in terms of computational efficiency. The proposal is compared to a multi-node and multi-core implementation using OpenMP to distribute the workload between all the available cores but not exploiting the GPUs.

The experiments were carried out on three nodes of the FinisTerrae-II, a supercomputer installed at the Supercomputing Center of Galicia (CESGA) [6]. Each node consists of two 12-core Intel Haswell 2680 CPUs, two NVIDIA Tesla K80 GPU accelerators, and 128 GB of memory. Each NVIDIA K80 GPU accelerator consists of two identical Tesla K80 GPUs featuring four Tesla K80 GPUs per node. The proposed implementation has been designed to perform the minimum number of data transfers between GPUs and CPU-GPU in order to reduce the execution time.

The multi-GPU program was compiled using the nvcc compiler (version 11.2.152), the CUB 1.8.0 version, the open source OpenMPI library (version 11.2.152), and the -O3 option. The multi-core program was compiled using the g++ compiler (version 10.1.0), the same version of the OpenMPI library, and the -O3 option. All programs use single precision. The results of computation times and speedup provided correspond to the average of five independent executions for each experiment.

Three different datasets named *Reservoir*, *Parasol*, and *House* were considered for the tests[1]. Each dataset consists of three, four, and five different frames of 1280×960 pixels and 5 bands respectively, as different passes over the same

[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.

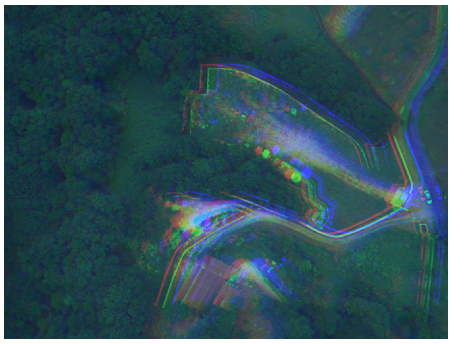

Fig. 2. False-colour composite of one frame of the *Parasol* dataset before co-registration.

Table 1. Execution times (in seconds) per node and stage of the multi-node and multi-GPU implementation proposed. "Co-reg." refers to the registration of bands while "Reg." refers to the registration of the resulting frames.

Stage	Master Node	Slave Node 1	Slave Node 2
Initialisation	1.03	0.69	0.97
Image transfers (MPI)	0.18	0.02	0.14
Co-reg. GPU 0	355.77	421.16	733.25
Reg. GPU 0	-	-	76.04
Co-reg. GPU 1	298.29	356.05	370.98
Reg. GPU 1	50.66	85.42	60.19
Co-reg. GPU 2	341.43	419.20	433.67
Reg. GPU 2	53.08	92.35	87.38
Co-reg. GPU 3	-	358.96	378.12
Reg. GPU 3	-	54.65	59.10
Result transfers (MPI)	410.93	0.02	0.03
Write to disk	2.69	-	-
Overheads	16.40	2.65	12.65
Total	**825.75**	**514.94**	**823.06**

region were performed for each geographical location. The radiance bands, ranging from 475 to 842 nm., were captured by a MicaSense RedEdge MX sensor on 18/07/2018 in different UAV flights and, as a result, have different spatial resolutions. Although the sensor captures all the bands at the same time, their co-registration is required. Figure 2 shows one frame of the *Parasol* scene before co-registration. Fuzzy edges and separation of the green, blue and red channels are clearly observed, for example, on roads and roofs of buildings. This problem does not appear in the frames once the bands are co-registered, as can be observed in Figure 3.

Table 1 shows the execution time for each node and stage of the proposed multi-node, multi-GPU implementation. The master node distributes the datasets among the three nodes of the cluster. The *Reservoir* dataset is registered in the master node, *Parasol* in slave node 1, and *House* in slave node 2. The different number of frames per dataset results in a different workload on each node as can be seen in the table. This results in different execution times for the different nodes. For example, in the case of *House*, in the first iteration the bands of four frames are registered in parallel in node 2 as there are four GPUs per node. When they finish, the bands of the fifth frame are co-registered while at the same time the registration of the four frames of *Parasol* has already started in slave node 1. The time required for registering the three datasets is 825.75s, which corresponds to the node with the highest execution time.

Table 2. Execution times (in seconds) and speedups of the GPU implementation (multi-GPU) over the CPU one (multi-core).

	CPU	GPU	Speedup
Master Node	3,080.24	825.75	3.73×
Slave Node 1	2,101.98	514.94	4.08×
Slave Node 2	3,077.49	823.06	3.74×
Total	**3,080.24**	**825.75**	**3.73**×

As can be seen in the table, in slave node 1, GPU-0 does not perform registration of frames because *Parasol* has only four frames, so only three GPUs are needed for the Reg. stage (frame registration). Similar situations occur at the master node with the *Reservoir* dataset (3 frames). It can also be observed that, although the master node registers the smallest dataset, it is the node that takes the longest execution time. This is because it is the node responsible for reading the data from the disk, distributing it, waiting to receive the results from the slave nodes, and writing to disk. But most of the execution time of this master node comes from waiting for the slave nodes.

As can be seen, the time required to distribute the two largest datasets to the slave nodes is less than 0.25s taking into account send and receive transfers. The time associated with GPU memory transfers is not indicated separately but included in the "Co-reg." and "Reg." stages, as it represents only 0.01s in the worst case. The "overheads" stage refers to the time required to profile the implementation and create the OpenMP threads for the different GPUs. This time is longer in the master node because this node has to distribute and receive the data, and on slave node 2 because this node requires the creation of a thread for the registration of each frame and, in addition, registers the dataset with the largest number of frames.

For comparison purposes, Table 2 compares the multi-GPU and a multi-core CPU implementations according to their execution time. The same data distribution is carried out in both implementations, but in the case of the multi-ccudaMemcpyore one, instead of GPUs, only the 24 CPU cores available per node are used to run the OpenMP version of HSI-KAZE [2]. On each node, four threads are launched to register in parallel four frames as in the multi-GPU version. The OpenMP implementation of HSI-KAZE launches up to 12 threads per frame to compute the most costly stages

(a) (b) (c)

Fig. 3. False-colour composite of the co-registered reference frames for the datasets: (a) *House*, (b) *Reservoir*, and (c) *Parasol*.

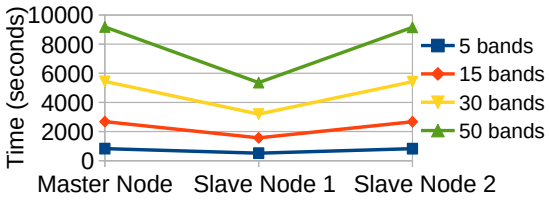

Fig. 4. Execution times when the number of bands increases.

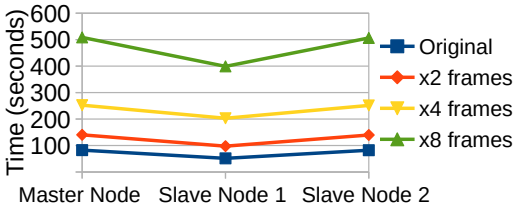

Fig. 5. Execution times when the number of frames increases.

of the algorithm, e.g. the keypoint matching stage. In total, 48 threads can be launched at the same time in a node (24 cores per node $\times$ 2 threads due to the Hyper-Threading Technology). 3,080.24s is the time required to register the three datasets in the cluster using the multi-core version compared to the 825.75s required by the multi-GPU version. The speedup achieved through the proposed multi-GPU parallelization is $3.74\times$.

Finally, the scaling properties of the implementation when the workload increases is analysed. Figure 4 shows the execution time for registering the same datasets in each node, as in the previous experiments, but increasing the number of bands per dataset. A linear growth is observed, as expected. In the case of increasing the number of frames per dataset (see Figure 5), a similar behaviour is observed. It is worth noting that, as explained above, slave node 1 presents the second smallest dataset and does not need to wait for the other nodes to finish as the master node.

4. CONCLUSIONS

In this article, a multi-node and multi-GPU implementation of HSI-KAZE is proposed to perform a two-level registration of multispectral images. First, the registration is performed at the band level and then at the image level. The proposal distributes the different multispectral datasets among the available nodes of a cluster using MPI. In this way, different datasets are registered in parallel. As each node has four GPUs, the different bands and images that belong to the same dataset are also processed in parallel. This implementation is about $3.73\times$ faster than a multi-core, multi-node implementation. As future work, we plan to analyse the performance of the proposed method with hyperspectral images of higher spatial resolution (VHR) where the limited memory of GPUs could be a bottleneck.

5. REFERENCES

[1] Jacqueline Le Moigne, Nathan S Netanyahu, and Roger D Eastman, *Image registration for remote sensing*, Cambridge University Press, 2011.

[2] Álvaro Ordóñez, Francisco Argüello, Dora B Heras, and Begüm Demir, "GPU-accelerated registration of hyperspectral images using KAZE features," *The Journal of Supercomputing*, pp. 1–15, 2020.

[3] Dongdong Yu, Feng Yang, Caiyun Yang, Chengcai Leng, Jian Cao, Yining Wang, and Jie Tian, "Fast Rotation-Free Feature-Based Image Registration Using Improved N-SIFT and GMM-Based Parallel Optimization," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 8, pp. 1653–1664, 2016.

[4] Yunsheng Zhang, Peilong Zhou, Yue Ren, and Zhengrong Zou, "GPU-accelerated large-size VHR images registration via coarse-to-fine matching," *Computers & Geosciences*, vol. 66, pp. 54–65, 2014.

[5] Álvaro 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.

[6] CESGA, "FinisTerrae-II supercomputer," https://www.cesga.es/en/infrastructures/computing/, [Online, accessed: 03-Jan-2022].