The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-05918-z

Using heterogeneous computing and edge computing to accelerate anomaly detection in remotely sensed multispectral images

Javier López-Fandiño[1,2] · Dora B. Heras[1,2] · Francisco Argüello[2]

Accepted: 15 January 2024

Abstract
This paper proposes a parallel algorithm exploiting heterogeneous computing and edge computing for anomaly detection (AD) in remotely sensed multispectral images. These images present high spatial resolution and are captured onboard unmanned aerial vehicles. AD is applied to identify patterns within an image that do not conform to the expected behavior. In this paper, the anomalies correspond to human-made constructions that trigger alarms related to the integrity of fluvial ecosystems. An algorithm based on extracting spatial information by using extinction profiles (EPs) and detecting anomalies by using the Reed–Xiaoli (RX) technique is proposed. The parallel algorithm presented in this paper is designed to be executed on multi-node heterogeneous computing platforms that include nodes with multi-core central processing units (CPUs) and graphics processing units (GPUs) and on a mobile embedded system consisting of a multi-core CPU and a GPU. The experiments are carried out on nodes of the FinisTerrae III supercomputer and, with the objective of analyzing its efficiency under different energy consumption scenarios, on a Jetson AGX Orin.

Keywords Multispectral · Anomaly detection · Extinction profiles · Heterogeneous computing · Edge computing

Javier López-Fandiño, Dora B. Heras and Francisco Argüello contributed equally to this work.

✉ Javier López-Fandiño
javier.lopez.fandino@usc.es

Dora B. Heras
dora.blanco@usc.es

Francisco Argüello
francisco.arguello@usc.es

[1] Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Santiago de Compostela, A Coruña, Spain

[2] Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Santiago de Compostela, A Coruña, Spain

Published online: 16 February 2024

1 Introduction

Anomaly detection (AD) plays a critical role in numerous fields, including remote sensing, surveillance, and environmental monitoring [1–3]. In the context of multispectral image analysis, AD techniques hold significant potential for identifying and characterizing irregularities or deviations from the norm in the images, providing valuable insights into complex natural systems [4].

The processing of multispectral or hyperspectral images always demands a considerable amount of computational resources due to the high dimensionality and complexity of the data and the need for real-time processing for many applications. This is especially true in the case of AD tasks over very high-resolution images, as AD algorithms require processing the whole spatial and the spectral information available in the images [4–6]. Different computational paradigms, ranging from high-performance computing (HPC) platforms such as clusters, grids, or clouds, to specialized accelerators such as graphics processing units (GPUs), field-programmable gate arrays (FPGAs), or even quantum computing solutions, have been exploited in this context. The choice of the most appropriate computing platform depends on the problem at hand and on the context in which it must be addressed [7]. For instance, in certain scenarios, data may be efficiently offloaded to supercomputers, while in others, it may be more practical to tackle the problem in situ using edge computing solutions or, simply, the commodity hardware available, for instance, a personal computer with a hardware accelerator such as a GPU.

Supercomputers play a dominant role in parallel computing applied to remote sensing [8]. A supercomputer is a mixture of shared and distributed memory systems that, in many cases, include heterogeneous nodes. Each node could consist of central processing units (CPUs), GPUs, FPGAs, or any other accelerator. Many models and portable libraries have emerged as possible standards for supercomputer programming, being OpenMP [9] and MPI [10] the most widely used. In fact, MPI is the facto current standard for parallel programming.

Accelerators have also played a dominant role in parallel computing and, especially, in image processing including multidimensional images from the remote sensing field [7]. GPUs, renowned for their ability to perform thousands of computations in parallel, excel in tasks involving repetitive mathematical operations. Their highly parallel nature based on multi-threaded, many-core processors and their very high memory bandwidth make them particularly well-suited for processing large-scale datasets and computationally intensive tasks. The development of NVIDIA's Compute Unified Device Architecture (CUDA) [11] has simplified the programming model for GPUs. Many libraries have been developed to provide the most common operations implemented in CUDA. OpenCL is also a relevant standard for parallel programming across CPUs, GPUs, and other processors [12].

Presently, GPU-based computers integrate the attributes of general-purpose computing, a high degree of parallelism, and high memory bandwidth, at a relatively lower cost compared to other alternatives. This makes them an attractive

option compared to a massively parallel system made up of only CPUs [13]. The heterogeneous computing approach that combines the strengths of both CPUs and GPUs can lead to even greater performance gains. By distributing the computational load between CPUs and GPUs based on their respective strengths, a more balanced and efficient workflow can be achieved, thereby maximizing overall processing speed and resource utilization. This kind of architectures has been previously used to tackle other tasks in multispectral images, such as registration [14] or domain adaptation [15], among others.

Recently, the use of edge computing architectures has arisen as a promising hardware alternative for remote sensing applications [7]. Edge computing is effective in reducing the delay between the acquisition and the processing of data [16]. Most edge computing devices are heterogeneous computing platforms such as, for example, the NVIDIA Jetson [17] used in this paper. These devices can help to make it possible to perform onboard remote sensing computation. Edge computing architectures are also very effective in reducing power consumption, which, nowadays, is a relevant requirement from both the sustainability and the economic perspectives. This can be especially important in real-time monitoring scenarios when the time needed to transfer the raw data to HPC centers is not an acceptable option [18].

Addressing the challenge of reducing the AD computation times, this paper presents an efficient heterogeneous parallel implementation to perform AD in high-resolution multispectral images. The objective is the detection of anomalies corresponding to human constructions within natural fluvial ecosystems. The resulting algorithm is executed on both a supercomputer and an edge computing device. Our methodology allows to automatically identify and characterize anomalies corresponding to human-made constructions by combining the use of extinction profiles (EPs) for extracting spatial information from the images with the well-known Reed–Xiaoli (RX) AD algorithm. The proposed algorithm can contribute to the development of effective conservation and management strategies for fluvial ecosystems, promoting sustainable development while safeguarding the ecological integrity of the ecosystem.

The remainder of this paper is structured as follows: Sect. 2 reviews the background related to efficient parallel implementations of AD for multispectral and hyperspectral images. In Sect. 3, the AD algorithm proposed in this work is introduced, with a particular emphasis on the parallel features of the algorithm. Section 4 describes the experimental setup and presents the results obtained on a real-world dataset of multispectral images. Finally, in Sect. 5, the concluding remarks are presented.

2 Related work

Processing multispectral images of fluvial ecosystems presents several challenges due to the inherent complexity and variability of natural landscapes [19]. The similarity between the spectral signatures of the various materials in the scene, such as geological features and human-made structures, coupled with the prevalence of vegetation in the landscapes under study, pose difficulties in distinguishing human

constructions, which seamlessly blend into the surrounding environment [20]. Additionally, multispectral images are vulnerable to atmospheric conditions, sensor noise, and fluctuating illumination, adding complexity to the accurate identification of anomalies in these dynamic environments [21].

Various approaches have been proposed to tackle the challenging problem of AD in multispectral images [19]. These methods can be broadly categorized into unsupervised, supervised, and semi-supervised techniques. Unsupervised methods, such as statistical modeling and clustering algorithms, do not require labeled training data and can automatically identify anomalies based on deviations from the normal data distribution [1, 22, 23]. Supervised methods, on the other hand, rely on labeled training data to learn the characteristics of normal and anomalous samples, enabling them to classify new instances accordingly [2, 24–26]. Semi-supervised methods aim to strike a balance between the two by utilizing a limited amount of labeled data and a larger pool of unlabeled data during the training process.

One notable algorithm that has garnered attention in AD is the RX algorithm [27]. The RX algorithm is an unsupervised technique based on the concept of the Mahalanobis distance. It characterizes anomalies by measuring the spectral deviations from the local mean of the surrounding pixels. The effectiveness of the RX algorithm lies in its ability to handle high-dimensional data and identify subtle anomalies, making it a promising candidate for detecting human constructions within fluvial ecosystems in multispectral images. Several variations of the RX algorithm have been applied to improve the detection accuracy of the traditional algorithm [28–31].

In the context of AD in multispectral images within fluvial ecosystems, incorporating spatial information alongside spectral data becomes a crucial aspect of achieving accurate and reliable results, as it has been shown by [32] for AD in hyperspectral images. This has also been explored in the previous classification-oriented works [23, 33–35]. Spectral information alone may not provide sufficient context to differentiate between natural variations and genuine anomalies in the complex and heterogeneous landscapes of fluvial ecosystems. By considering the spatial relationships among neighboring pixels, valuable contextual cues can be extracted, enhancing the discrimination of anomalies from the background. In this work, we incorporate the spatial information together with the spectral one by introducing the filtering technique called EP [36], which is an alternative to the widely recognized attribute profile (AP) [37, 38]. EP is based on the concept of extinction filters (EFs) which are extrema-oriented connected filters that, unlike the AP, preserve the original height of the extrema kept in the image. The parameter tuning is also easier for EPs than for APs due to the fact that they are independent of the kind of attribute being used and only based on the number of extrema to be kept at each level of the EP [36].

Several works have explored the use of parallelization techniques to efficiently tackle the real-time AD problem. [39] presents a GPU implementation of the RX algorithm for AD using CUDA. It remarks the relevance of processing sub-images of the image independently in different hardware to exploit the parallelism minimizing the communications, as they are a common bottleneck in multi-GPU implementations. The paper also analyzes the power consumption of the algorithms, which is especially relevant for real Earth observation missions using onboard computation.

The projection of AD to a widely used edge computing device, an NVIDIA Jetson GPU, is explored in [40] over a hyperspectral urban image acquired by the AVIRIS sensor, concluding that there is a promising solution for hyperspectral image processing in low-power consumption scenarios. In turn, [41] proposes the use of FPGAs to perform recursive RX-based AD in hyperspectral images from both urban and natural scenes, also acquired by the AVIRIS sensor. They show how different variants of the traditional RX algorithm can be projected efficiently to FPGA architectures, along with GPU and cloud computing alternatives, concluding that the main limiting factor in FPGAs and GPUs is memory capacity. This limiting factor can be avoided by scaling the number of nodes used. Another FPGA AD algorithm focused on minimizing power consumption is presented in [42], demonstrating that hyperspectral images with thousands of pixels and hundreds of spectral bands can be processed with a power budget of only 1.3 W. Embedded devices such as FPGAs can also be used to perform AD through the use of deep learning techniques. For instance, a deep convolutional neural network is used in [43] for natural anomaly detection in multispectral images. Another possibility to make the algorithms more adequate for edge computing devices is to align the data processing with the data acquisition process. For this purpose, a line-by-line AD technique is presented in [44], which aims to process hyperspectral data in a manner consistent with its collection by push-broom scanners.

When assessing a task such as AD is also important to take into account that different architectures may be better suited for each specific processing step. This aspect is explored in [45], where a combination of CPU and GPU is employed to achieve a time-efficient AD technique. The proposed technique is based on the use of multivariate normal mixture models applied to a simulated search and rescue scenario in a real hyperspectral image captured by a HySpex visual and near-infrared (VNIR) hyperspectral camera.

3 Heterogeneous parallel EP-based AD scheme

In this section, the proposed parallel algorithm for detecting anomalies in high-resolution multispectral images of fluvial ecosystems is presented. A hybrid algorithm using MPI, OpenMP, and CUDA is used to exploit the different levels of parallelism offered by a heterogeneous architecture that, as it will be explained later, includes nodes with a multi-core CPU and a GPU.

The outline of the algorithm is shown in Fig. 1. It consists of three main stages: First, the spatial information is extracted from the input image by computing an EP over each band of the image separately. The five bands available in the images under study are represented in the figure. As a result, an extended EP is produced by accumulating the results of the individual EPs constructed for each band of the image. Then, a second stage consists of applying the RX anomaly detector over the resulting extended EP. This stage is more efficiently calculated over only one node of the computer platform. A two-dimensional gray-scale intensity image is obtained as a result. Finally, the processing requires calculating and applying a threshold to

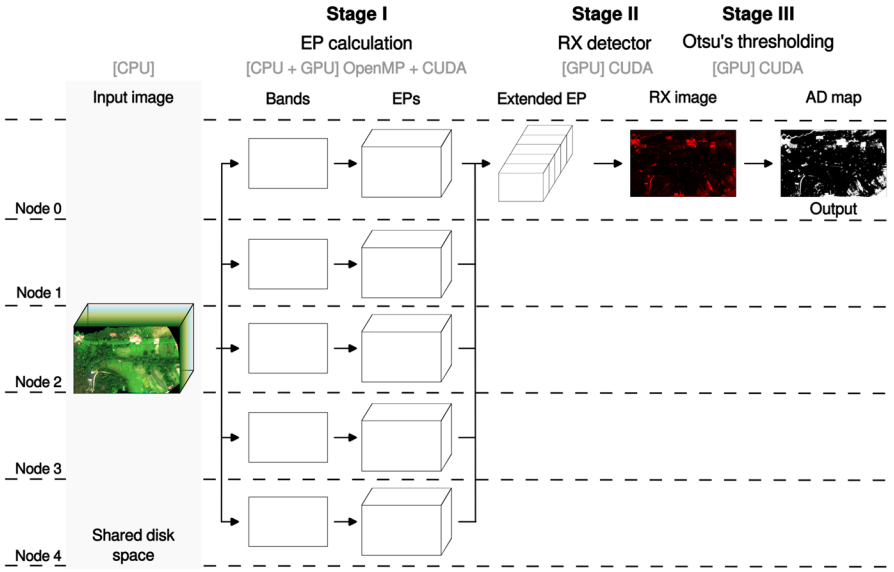


Fig. 1 Parallel RX+EP AD algorithm for heterogeneous computing

produce the output AD map. This is the objective of the third stage, which applies Otsu's algorithm, producing a binary map of anomalies as output.

More details of the implementation presented in Fig. 1 are offered in Algorithm 1, where all the computational steps for each stage of the algorithm and the platforms where are computed are annotated on the right side.

More in detail, the first stage of the algorithm presented in Fig. 1 and detailed in lines 1–18 of Algorithm 1, can be considered as a spatial processing where the structures of interest of the image are highlighted through the computation of the EPs. As the anomalies that need to be detected correspond to a set of uniform pixels more than isolated pixels, this spatial processing stage helps in identifying these structures at different levels. This is a very common process followed when changes or anomalies need to be detected in multi- or hyperspectral images. In this case, the method for extracting spatial information is the use of EPs. The processing of the EP for each image band is independent, making it a good candidate to be computed in parallel. After the tasks corresponding to the processing of each band are distributed through the use of MPI, a hybrid CPU-GPU approach is used to tackle this operation. As each EP calculation consists of some steps where the individual operations are performed over each pixel of the image, it is a perfect fit to be computed in GPU using CUDA. Other steps that are performed over the node-array representation of the image tree [36] offer fewer opportunities for parallelization to be more efficiently computed by the CPU. This is the case for the stages devoted to obtain the parents of each node by means of the union-find algorithm and the computation of the node array (steps 4–5 in pseudocode in Algorithm 1).

The calculation of the EP of a one-band gray-scale image consists of the application of several opening and closing operations at different granularity levels over the

original images [36]. So, inside each node, different levels of the EP need to be computed. To perform this operation in the most efficient way, the node-array representation of the image is computed first (lines 2–5 in the pseudocode). This representation can be seen as a max-tree [36] computation where the parent of each node is obtained in a union-find process and stored in a structure together with the attribute of interest for each node, that is, the area in this case. Once this information is gathered, it is possible to compute the extinction values of each node as proposed by [36] (line 6). The steps corresponding to the application of the extinction filter for each number of extrema selected (lines 7–9 and 16–18 in Algorithm 1) can be computed in parallel through the use of OpenMP once the extinction values of the image have been computed in the previous step, as they are independent computations for each level of the EP. Figure 2 illustrates the EP for one band of a multispectral image. The original band is in the middle, and the different components of the profile, corresponding to the result of applying a three-level EP, are represented on both sides.

It is worth noting that, for the computation of the closing profiles (lines 10–18), the same code is used but carried out over the negated input image. This is a usual approximation to this task that has been used, for instance, in [38]. The EPs for the different bands of the input multispectral image are concurrently computed in different nodes of the computing platform, through the use of MPI, thus reducing the computation time. The opening and closing profiles individually computed for each band of the image are stored in memory composing an extended EP. This EP will be the input of the second stage of the algorithm.

The second stage of the algorithm consists of the application of the anomaly detector, the RX algorithm, over the extended EP. As the second stage of the algorithm (lines 19–22) presents a low computational load, it is efficiently computed in only one node, so the extended EP is computed by gathering the individual EPs, as shown in Fig. 1. This part of the algorithm is individually performed

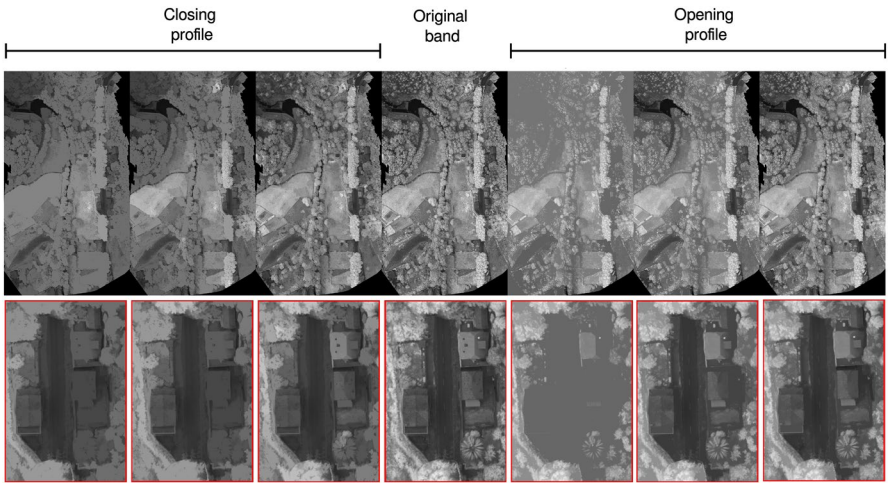


Fig. 2 EP computation for one band of a multispectral image (first row) and zoom over a small region (second row)

over each pixel and, therefore, totally computed in GPU to achieve low execution times. We can identify in the algorithm the steps for the parallel calculation of the RX anomaly detector [27] to obtain a one-band gray-scale image where the higher the intensity of a pixel, the higher the probability of it being an anomaly. It is based on the calculus of the Mahalanobis distance between each pixel of the stacked EP and the average pixel value of the same stacked EP. All the steps are calculated in GPU using CUDA.

The third and final stage of the algorithm performs the application of a threshold technique to obtain a binary AD map (lines 23–26 in the algorithm) identifying each pixel as anomaly or non-anomaly. This is tackled through Otsu's threshold algorithm as it is an automatic threshold technique that has proven to produce the best discrimination with a low computational cost [46, 47]. This algorithm calculates the threshold based on the histogram of the gray levels of the image. This stage of the algorithm is also executed on GPU as both the histogram calculation step and the final binary decision over each pixel of the image can benefit from the highly parallelizable GPU architecture thus achieving lower execution times.

Algorithm 1 Parallel RX+EP AD algorithm.

	Stage 1: Extinction Profile application.	
	Input: Multiband image.	
	Output: Extended EP.	
	Parameters: Nº of levels and number of extrema to be kept at each level of EP.	
1:	**for** band in image **do**	▷ MPI
	Opening:	
2:	Max-tree computation:	
3:	Reorder image and indexes.	▷ GPU
4:	Get parents of each node (union-find).	▷ CPU
5:	Compute node array.	▷ CPU
6:	Compute extinction values.	▷ CPU-GPU
7:	**for** level in EP **do**	▷ OpenMP
8:	Calculate extinction filter.	▷ CPU-GPU
9:	Get filtered image.	▷ GPU
	Closing:	
10:	Negate image.	▷ GPU
11:	Max-tree computation:	
12:	Reorder image and indexes.	▷ GPU
13:	Get parents of each node (union-find).	▷ CPU
14:	Compute node array.	▷ CPU
15:	Compute extinction values.	▷ CPU-GPU
16:	**for** level in EP **do**	▷ OpenMP
17:	Calculate extinction filter.	▷ CPU-GPU
18:	Get filtered image.	▷ GPU
	Stage 2: RX detector.	
	Input: Extended EP.	
	Output: RX Image.	
19:	Compute mean vector.	▷ GPU
20:	Compute covariance matrix.	▷ GPU
21:	Invert covariance matrix.	▷ GPU
22:	Compute Mahalanobis distance.	▷ GPU
	Stage 3: Otsu's thresholding.	
	Input: RX image.	
	Output: AD map.	
23:	Rescale image between [0, 255].	▷ GPU
24:	Compute image histogram.	▷ GPU
25:	Calculate threshold value by Otsu's method.	▷ GPU
26:	Generate binary image regarding the threshold.	▷ GPU

3.1 Comparison with other parallel AD implementations

The algorithm proposed in this paper aims to exploit the capabilities of the available hardware on a heterogeneous computing platform at different levels. This has been the methodology used by other parallel AD implementations in the literature, such as [45], where a dual Quad-Core Intel Xeon CPU and a NVidia GeForce 8800 Ultra GPU are used to reduce the execution times of an AD method for hyperspectral images based on the use of multivariate normal mixture models. Similarly in [41], the authors conclude that embedded, GPU, and cloud architectures should be combined to achieve efficient processing of remote sensing data.

In particular, in our paper, the first stage of the algorithm, as we described in Algorithm 1, exploits a multi-node implementation for extracting parallelism. The reason is that the EP calculation computed by this stage of the algorithm can be individually processed for each band. The implementation combines the use of MPI, OpenMP, and CUDA to efficiently distribute the computational load of this stage among the available nodes of a supercomputing device. The same approach is applied, for instance, by [41], which introduces a multi-node cloud implementation of an AD algorithm based on the use of RX.

The second stage, the RX detector, is performed entirely in a single GPU by using CUDA in our algorithm. Given that the amount of data to be processed is small in this stage, it would be difficult to compensate for the communication times needed to use a distributed architecture. This task has been tackled similarly in other parallel AD implementations such as [39, 41]. The same remarks apply to the Otsu's thresholding needed to obtain the final AD map.

Finally, when prioritizing the minimization of power consumption, the use of embedded devices has been established in the literature as the optimal alternative [40, 42]. Different parallel AD algorithms in the literature, for instance, [41] or [43], have been projected to FPGAs. In our paper, an NVIDIA Jetson platform [17] is used. This platform presents, in comparison with a FPGA, the advantage that the implementation required is more similar to that of general-purpose architectures.

4 Experimental results

This section is devoted to summarize the experiments carried out for the validation of the algorithm presented in this work for the AD of human-made constructions in fluvial ecosystems. First, the dataset and experimental setup selected for the experimentation will be presented in Sects. 4.1 and 4.2. Then, the achieved accuracy and performance results will be analyzed in Sect. 4.3.

4.1 Dataset

The experiments were conducted utilizing data captured by the MicaSense RedEdge multispectral sensor, onboard a specialized unmanned aerial vehicle (UAV). This

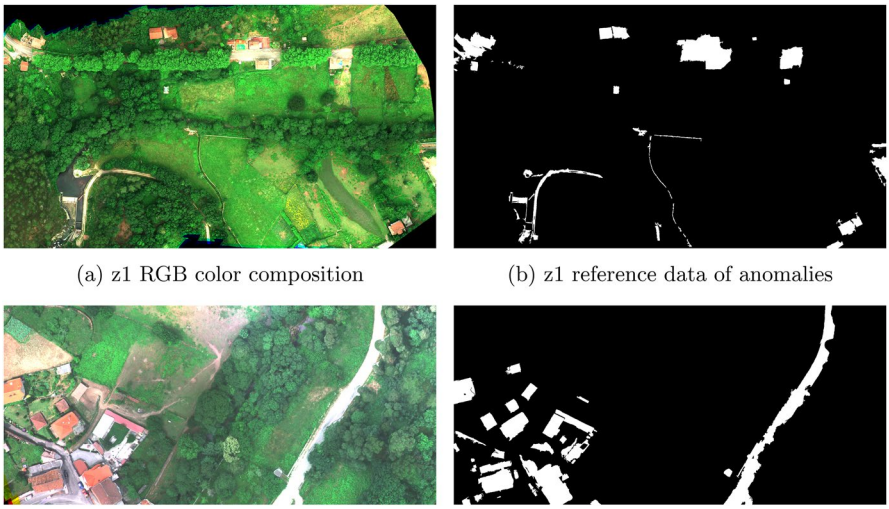
advanced sensor is capable of capturing imagery across five distinct spectral bands: blue (475 nm), green (560 nm), red (668 nm), red-edge (717 nm), and near-infrared (NIR) (840 nm). The aerial images of fluvial ecosystems were taken during the summer months of 2018 in the region of Galicia, Spain, at an altitude of 120 m, offering a very high spatial resolution of 8.2 cm per pixel [20].

Figure 3 shows an RGB color composition and the reference data of anomalies available for the dataset used for the experiments. These images depict watershed ecosystems located in densely vegetated regions. Within this context, structures such as buildings, dams, and roads are categorized as anomalies necessitating detection to trigger alarms. These alarms, in turn, will be managed by individuals responsible for overseeing the ecosystem. It is important to note that anomalies within the reference data of each image represent a small fraction of the total pixel count, which aligns with typical scenarios. Furthermore, it is of paramount importance to emphasize that detecting all anomalies in the reference data is crucial for this application. Additionally, it is crucial to highlight that detecting all the anomalies in the reference data is imperative for this application as missing alarms could provoke damage in the ecosystem.

Table 1 summarizes the main characteristics of the two considered 5-band multispectral images. As can be seen, each of the images presents around 4% of anomalies over the total number of pixels. The size of the images corresponds to the case where the pixel information is stored in a 4-byte format.

4.2 Experimental setup

In this section, we describe the hardware and software configurations utilized for conducting the experiments in this paper. Two different hardware setups have been



(a) z1 RGB color composition (b) z1 reference data of anomalies

(c) z2 RGB color composition (d) z2 reference data of anomalies

Fig. 3 Oitavén river dataset consisting of two multispectral images (z1 and z2). Anomalies in white color

Table 1 Dataset description consisting of two multispectral images

Name	Oitavén river z1	Oitavén river z2
Width	3807	2081
Height	2141	957
Number of spectral bands	5	5
Number of anomalies (pixels)	321,710	83,130
Number of non-anomalies (pixels)	7,829,077	1,908,387
Percentage of anomalies (%)	3.95	4.17
Percentage of non-anomalies (%)	96.05	95.83
Size (MB)	155.5	38

used in order to compare different approximations to solve this problem: different nodes of a distributed memory supercomputer and a Jetson NVIDIA computing platform.

First, a multi-node supercomputer called FinisTerrae III with multiple GPUs per node, the FinisTerrae III supercomputer, is used to maximize the exploitation of parallelism at different levels. FinisTerrae III is located at the Galician Supercomputing Center (CESGA) [48] and consists of 354 nodes that are interconnected as shown in Fig. 4. As shown in Table 2, each node includes two Intel Xeon Ice Lake 8352Y processors with 32 cores each and 256 GB of memory. For the experiments, five nodes were used, as each node computes the

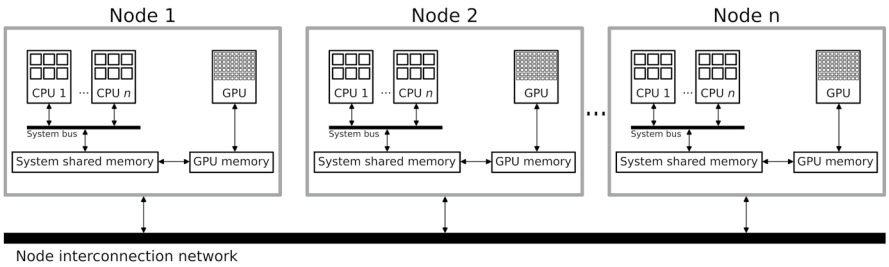


Fig. 4 FinisTerrae III distributed memory system

Table 2 Hardware setup of the different computing platforms used for the experiments

Platform	FinisTerrae III node	Jetson AGX Orin
CPU	2x Intel Xeon Platinum 8352Y	Arm Cortex-A78AE
Number of cores	32+32	12
Frequency	2.8 GHz	2.2 GHz
RAM size	256 GB	64 GB
Cache size	80 MiB L2 + 96 MiB L3	3 MB L2 + 6 MB L3
GPU	NVIDIA A100	NVIDIA Ampere-based
DRAM size	40 GB	64 GB (shared with CPU)

EP over one band of the 5-band multispectral images captured by the MicaSense RedEdge multispectral sensor. In each one of these nodes, CUDA codes run on one of the NVIDIA A100 GPUs available in the node. This GPU model is based on the NVIDIA Ampere architecture, and it is equipped with 108 multiprocessors and 64 cores per multiprocessor, resulting in 6912 cores. The CUDA capability version is 8.0, and each card has 40 GB of DRAM memory, as shown in Table 2.

The previously described computing platform is too expensive to be available in the usual remote sensing environments where decisions need to be taken for many applications with short response times and far from supercomputing centers. A more affordable computing platform that has been considered is an NVIDIA Jetson AGX Orin platform, also described in Table 2. This platform was selected as a representative of edge computing devices, aiming for in-place real-time computation of the proposed algorithm. This platform also allows us to analyze the effect that different energy power availability has on the computation times of the scheme, as remote sensing applications usually require computation in energy-limited platforms.

The NVIDIA Jetson AGX Orin Developer Kit [17] used to evaluate the performance of the proposed algorithm over a mobile embedded system provides a 12-core Arm Cortex-A78AE v8.2 64-bit CPU together with a 2048-core NVIDIA Ampere architecture GPU. It also includes 64 GB of RAM. Besides, as shown in Table 3, the kit allows us to configure the hardware to operate within different power budgets ranging from 15 to 60 Watts, allowing us to simulate real edge computing scenarios. The varying energy consumption is achieved by disabling some hardware components, such as reducing the number of online CPU cores or disabling the GPU Texture Processor Cluster, and also by limiting the frequency of both the CPU and the GPU cores [49].

The FinisTerrae III codes have been compiled using the g++ 10.1.0 version with OpenMP 4.0 support under Linux. Regarding the GPU implementation, the CUDA codes have been compiled using the nvcc version 12.2 of the toolkit. Version 4.1.4 of the OpenMPI library was used for the multi-node experiments. The Jetson codes have been compiled using the g++ 9.4.0 version with OpenMP 4.0 support under Linux and nvcc version 11.4 of the CUDA toolkit. The Thrust library was used to accelerate sorting operations.

Table 3 Jetson AGX Orin power mode budgets

Power budget	15 W	30 W	50 W	MAXN (60 W)
Online CPU number	4	8	12	12
CPU max frequency (MHz)	1113.6	1728	1497.6	2201.6
GPU max frequency (MHz)	408	612	816	1301

4.3 Results

4.3.1 Accuracy assessment

In this section, the results, in terms of AD accuracy, are presented. For this purpose, two main metrics are considered: the area under the curve (AUC) and the percentage of anomalies detected (i.e., the true-positive rate). The number of true-positive (TP), true-negative (TN), false-positive (FP), and false-negative (FN) pixels are shown for completeness purposes, together with the precision–recall AUC (PR-AUC). The AUC can be considered as a standard in the literature to analyze the quality of AD algorithms [50–52]. The percentage of anomalies detected will show how the proposed method improves the ability to detect anomalies as compared to a traditional RX-based algorithm. We emphasize this metric because, as mentioned earlier, the primary objective of this study is to detect as many anomalies as possible, even if it leads to a higher false-positive rate. It is worth noting that the PR-AUC archives higher values when low false positive and negative rates are obtained, whereas the main purpose of this work is to increment the number of anomalies detected. Tests were performed to check that the parallel versions of the algorithm present the same accuracy values as the sequential one.

Table 4 shows the accuracy values obtained for the considered dataset with different EP configurations, whereas Fig. 6 shows the corresponding AD maps obtained for both images. The EP parameters, indicated between parenthesis in the table, have been chosen empirically for each image. Each value indicates the number of extrema to be kept for each level of the EP calculated for each band of the image. As can be seen in the table, varying the EP configuration parameters allows for improving the anomaly detection up to 23% and 27% for the z1 and z2, respectively, with respect to the case without the application of EP (first row in the table for each image). This also results in an improvement of the AUC for

Table 4 Accuracy of the parallel AD algorithm for z1 and z2 multispectral image. Results for different configurations of the EP stage. The best results for each column and image are highlighted in bold

	AUC	PR-AUC	Detected anomalies (%)	TP	TN	FP	FN
z1 image							
RX	0.818	**0.636**	65.46	210,605	**7,255,470**	**138,711**	111,105
RX+EP(8,4,2,1)	0.865	0.545	83.06	267,220	7,035,707	793,370	54,490
RX+EP(64,8,4,1)	0.876	0.566	83.90	269,934	7,154,346	674,731	51,776
RX+EP(64,8,4,2)	**0.888**	0.570	**88.32**	**284,139**	6,990,535	838,542	**37,571**
z2 image							
RX	0.722	**0.597**	46.35	78,104	**1,786,809**	**35,857**	90,407
RX+EP(64,32,16,8)	0.696	0.418	51.52	86,808	1,600,064	222,942	81,703
RX+EP(8,4,2,1)	0.694	0.415	56.72	95,586	1,497,095	325,911	72,925
RX+EP(32,16,2,1)	**0.788**	0.529	**73.00**	**123,009**	1,543,023	279,983	**45,502**

both considered images even though the number of FP is bigger when the EPs are introduced. Nevertheless, this increase is much smaller than the TP one, being about 5% for both z1 and z2 images.

Figure 5 shows the ROC curves of the accuracies obtained for the proposed method for different configurations of the EP stage of the algorithm that produces the extended EP of the image. It can be seen that a proper parameterization of the EP is relevant to achieve the best results. Nevertheless, in general, almost every parameterization improves the accuracies obtained without the application of EPs. It is also worth noting that, as expected, increasing the ability to detect the highest number of anomalies possible (high values of TP), which is our main objective for the aim of the ecosystem supervision studied in this work, also involves an increase in FP.

The AD maps obtained for both images are shown in Fig. 6. In the images, TP is colored in green, FP is colored in blue, FN is colored in red, and TN remains black as background. As it can be seen, the areas colored in red, highly decrease when the use of EPs is included together with the RX, meaning that the capability of the algorithm to detect real anomalies is greater in these cases. It also can be seen that the blue areas that appear do not have regular shapes and have the appearance of irregular structures. This will make it easier for these areas to be disregarded later by the application of some automatic post-processing technique.

4.3.2 Performance results

The execution times achieved for the different hardware setups introduced in 4.2 are summarized in this section. All data shown here correspond to the average values of 10 independent runs over the EP configurations which achieve the highest accuracy values, i.e., RX+EP(64,8,4,2) for the z1 image and RX+EP(32,16,2,1) for the z2 image. Table 5 shows the time needed to perform the computation for z1 and z2 images in the FinisTerrae III setup, using one and five nodes (one per band of the image), respectively. An alternative implementation where all the processing

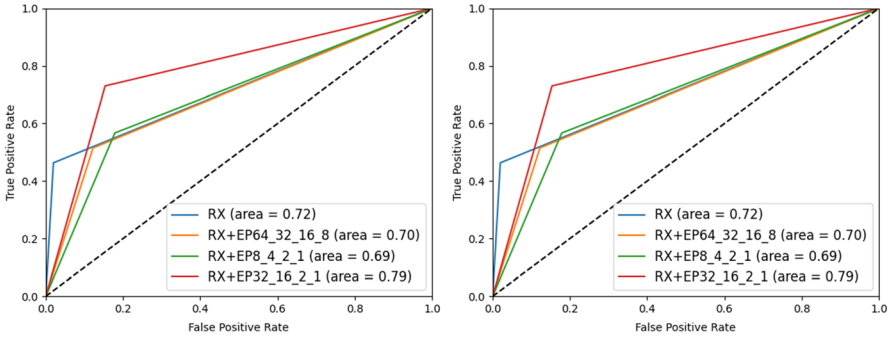
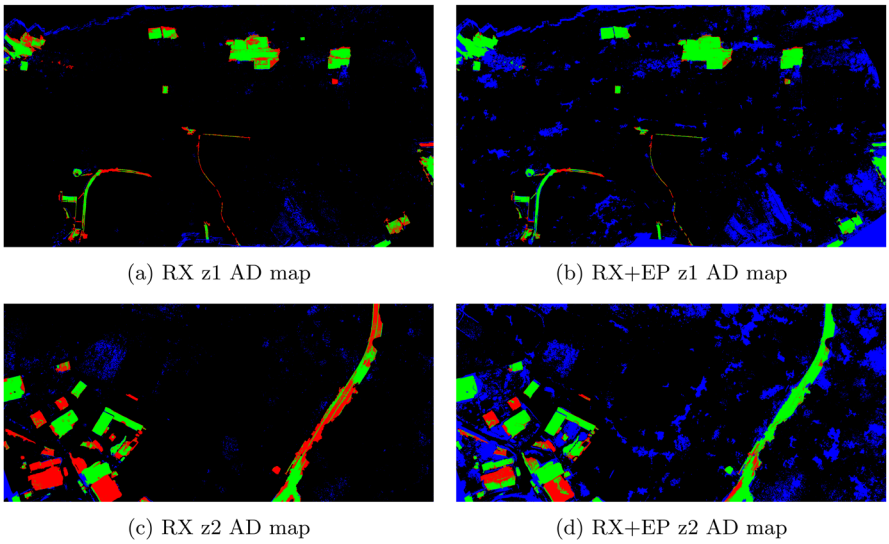


Fig. 5 ROC curves for z1 and z2 images with the parallel RX+EP algorithm for AD and for different EP configurations



(a) RX z1 AD map (b) RX+EP z1 AD map

(c) RX z2 AD map (d) RX+EP z2 AD map

Fig. 6 AD maps for z1 and z2 images with the parallel RX+EP algorithm

is performed in the CPU is included as a baseline. The speedup of the different versions with respect to the CPU baseline is also included. As it can be seen, the most expensive computation steps are *Get parents* and *Node array*, which have to sequentially walk through the image to obtain the max-tree representation that will then be used to obtain the extinction values. These steps are even more costly, in relative terms, in the CPU-GPU version, as they cannot be accelerated with the GPU. On the other hand, the *Negate image* and *RX: Mahalanobis* steps are the ones that can benefit the most from the use of GPUs, achieving speedups up to 469× and 169×, respectively.

Therefore, given the few parallelization opportunities available inside the EP computation for each band of the image, it becomes crucial to exploit the parallelism that can be achieved with the use of a multi-node hardware platform such as the FinisTerrae III. As it is shown in Table 5, the EP computation stage of the z1 image can be accelerated from 8.3 s to just 1.4 s when all the bands are computed in parallel in different nodes. Similarly, the EP computation stage of the z2 image takes 2.1 s in the single-node CPU version and only 0.36 s in the multi-node CPU-GPU one. This allows the total speedup of the RX+EP AD algorithm, as compared with the CPU version, to increase from 6.9× in the single-node configuration to 23× in the multi-node configuration for the z1 image and from 4.3× to 8.1×, in the same configurations, for the z2 image.

In order to illustrate the performance of the algorithm for different image sizes, experiments have been carried out. The multi-node CPU-GPU version of the RX+EP algorithm, executed on the FinisTerrae III supercomputer, was used for these experiments. Figure 7 shows the time needed to execute the algorithm for three different sizes: the original z1 image, an image with the same number of spectral bands but 2× bigger in the spatial dimension, and an image 4× bigger in the

Table 5 Execution times, in seconds, and speedups for z1 and z2 images for the RX+EP(64,8,4,2) and RX+EP(32,16,2,1) configurations, respectively, on FinisTerrae III. Times include CPU processing, network communication costs, and GPU computations

Step	z1 image					z2 image				
	Single-node			Multi-node		Single-node			Multi-node	
	CPU	CPU-GPU		CPU-GPU		CPU	CPU-GPU		CPU-GPU	
EP opening										
Order image	0.6539	0.0809	8.1×	0.0163	40.2×	0.0988	0.0232	4.3×	0.0047	21.1×
Get parents	2.6112	2.5340	1.0×	0.5052	5.2×	0.6729	0.6462	1.0×	0.1297	5.2×
Node array	0.5631	0.5933	0.9×	0.1181	4.8×	0.1484	0.1531	1.0×	0.0308	4.8×
Extinction values	0.1207	0.1021	1.2×	0.0199	6.1×	0.0465	0.0403	1.2×	0.0082	5.7×
EP application	0.2264	0.1619	1.4×	0.0334	6.8×	0.0586	0.0460	1.3×	0.0090	6.5×
EP closing										
Negate image	0.0623	0.0012	52.4×	0.0002	264.8×	0.0390	0.0004	96.0×	0.0001	469.3×
Order image	0.6604	0.0776	8.5×	0.0142	46.4×	0.0990	0.0238	4.2×	0.0050	19.7×
Get parents	2.5567	2.4906	1.0×	0.5006	5.1×	0.6572	0.6335	1.0×	0.1275	5.2×
Node array	0.5153	0.5128	1.0×	0.1025	5.0×	0.1387	0.1387	1.0×	0.0279	5.0×
Extinction values	0.0964	0.0704	1.4×	0.0140	6.9×	0.0423	0.0342	1.2×	0.0069	6.1×
EP application	0.2114	0.1320	1.6×	0.0273	7.7×	0.0512	0.0359	1.4×	0.0071	7.2×
EP total	8.2778	6.7568	1.2×	1.3519	6.1×	2.0526	1.7752	1.2×	0.3569	5.8×
RX: mean	0.5363	0.0064	83.3×	0.0064	83.5×	0.1293	0.0012	108.2×	0.0012	108.4×
RX: covariance	24.5064	0.8390	29.2×	0.8390	29.2×	5.8359	0.1996	29.3×	0.1996	29.2×
RX: inverse cov	0.0001	0.0007	0.1×	0.0007	0.1×	0.0001	0.0007	0.1×	0.0007	0.1×
RX: Mahalanobis	20.0732	0.1191	168.5×	0.1191	168.6×	4.8048	0.0292	164.6×	0.0292	164.6×
Otsu's threshold	0.0106	0.0038	2.8×	0.0038	2.8×	0.0019	0.0008	2.2×	0.0008	2.2×
RX+Otsu total	45.1266	0.9682	46.5×	0.9682	46.6×	10.7719	1.2316	8.8×	1.2316	8.7×
Total	**53.4044**	**7.7251**	**6.9×**	**2.3201**	**23.0×**	**12.8245**	**3.0068**	**4.3×**	**1.5885**	**8.1×**

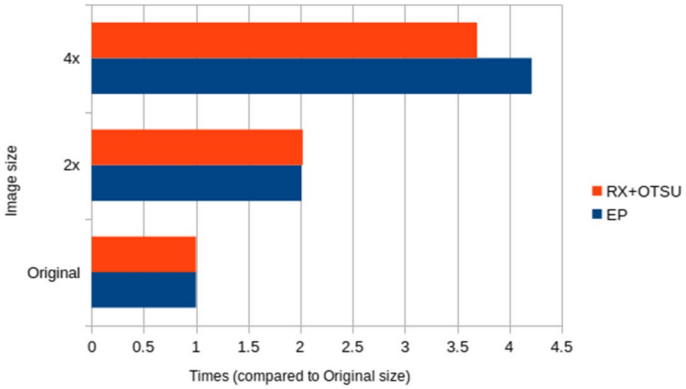


Fig. 7 Execution time scaling when the size of the z1 image is increased. The times are expressed in multiples of the execution time over the times for the original z1 image. Times include CPU processing, network communication costs, and GPU computations

spatial dimension. The time is shown separately for the EP stage and for the RX and Otsu stages. As can be seen, the time needed for the execution of the algorithm for these images scales nearly linearly with the image size. Nevertheless, in the 4× size image, it starts to be noticeable that the spatial part of the processing, the EP, loses some performance as it needs 4.2× the time needed for the original image. On the other hand, the RX detector achieves better performance (it only needs 3.69× the time needed for the original image). This can be explained because the spectral part is more suitable for pixel-level GPU parallelism and the bigger the number of pixels to process, the bigger the opportunities for exploiting the high number of processing cores available in the GPU.

Regarding the edge computing Jetson platform introduced in Sect. 4.2, experiments have been carried out with three different power budgets (15 W, 30 W, and MAXN (60 W)). The findings previously commented in the FinisTerrae III platform remain valid for this platform: The *Get parents* and *Node array* steps of the EP computation are still the most time-consuming steps, and the speedups achieved when the available GPU is exploited also remain similar. Nevertheless, as was expected, the more limiting the power budget selected, the higher the execution times achieved for the same computations.

The most parallelizable steps are those that benefit the most from the higher power budgets, as they can exploit both, the larger number of cores available and the increased frequency of the CPU and GPU. In this way, the *RX: mean* and the *RX: Mahalanobis* steps are the ones with a higher increase in speedup when the CPU version is compared with the CPU-GPU one. The speedup increasing on these steps ranges from 4× to 10× for the 15 W power budget up to 9.7× to 17.7× for the 60 W power budget.

As it is shown in Table 6, the reduction in the available power and the disabling of some hardware components, resulting in fewer parallel computing opportunities for the algorithm, greatly increase the time needed for the computation up to 7.1×. This makes it clear that the power requirements must be carefully chosen in an

Table 6 Summary of execution times, in seconds, and speedups, with respect to the 15 W version, for different versions of the algorithm on the Jetson AGX Orin. Times include CPU and GPU processing

Version		z1 image		z2 image	
		CPU	CPU-GPU	CPU	CPU-GPU
15 W	EP	140.2101	105.9384	6.4297	5.0576
	RX+Otsu	87.7020	29.9635	21.7822	7.9783
	Total	227.9121	135.9019	28.2119	13.0360
30 W	EP	23.8902	20.9899	5.7982	5.7447
	RX+Otsu	60.4277	16.6504	14.9905	3.7830
	Total	84.3178	37.6403	20.7887	9.5278
60 W	EP	14.4470	13.1580	3.5670	3.6841
	RX+Otsu	45.1647	6.0055	11.1657	1.2641
	Total	59.6117	19.1635	14.7327	4.9481
	Speedup 30 W	2.7×	3.6×	1.4×	1.4×
	Speedup 60 W	3.8×	7.1×	1.9×	2.6×

edge computing environment to achieve the right balance between autonomy and performance.

5 Conclusions

This paper introduces a computationally efficient parallel algorithm for AD. The algorithm is specifically designed to run on heterogeneous computing platforms, comprising nodes with multi-core CPUs and GPUs. AD is accomplished through a combination of an extended extinction profile for spatial information extraction and a detector known as the RX algorithm.

The resulting parallel hybrid MPI+OpenMP+CUDA algorithm outperforms a traditional RX approach, detecting up to 27% more anomalies in the images presented in this paper. Experiments were conducted using the FinisTerrae III multinode supercomputer, analyzing high-resolution multispectral images of fluvial ecosystems. Speedups of up to 23 × were achieved.

Furthermore, the same algorithm was executed on a mobile embedded system, specifically a NVIDIA Jetson. This aimed to assess the feasibility of running the algorithm under various power consumption limitations. Experiments have shown an increase of up to 7.1 × in execution time when the power consumption is limited to 15 W, compared to the situation with a limit of 60 W.

A challenge for the future would be addressing workload imbalances when processing images, such as hyperspectral ones, with a higher number of spectral bands. In such cases, it may not be practical to allocate as many nodes as there are bands in the image. Additionally, adapting the implementation to work with series of images covering more extensive spatial areas would allow the application of the algorithm to a wide variety of remote sensing applications.

Acknowledgements The authors want to acknowledge Centro de Supercomputación de Galicia (CESGA) for providing access to the supercomputer FinisTerrae III. We also would like to thank Augas de Galicia from Xunta de Galicia for the support in the construction of the reference information for the multispectral images.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was supported in part by the Civil Program UAVs Initiative, promoted by the Xunta de Galicia and developed in partnership with the Babcock company to promote the use of unmanned technologies in civil services. It was also supported by grants PID2019–104834GB–I00, PID2022–141623NB-I00, and TED2021–130367B–I00 funded by MCIN/AEI/10.13039/501100011033 and by "European Union NextGenerationEU/PRTR." We also have to acknowledge the support 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], by Junta de Castilla y León [Project VA226P20 (PROPHET–II)], and by European Regional Development Fund (ERDF).

Data Availability Statement Restrictions apply to the availability of the data used in this study. Datasets are the property of Babcock International and are used with permission.

Declarations

Conflicts of interest The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical approval Not applicable.

References

1. Guo Q, Pu R, Cheng J (2016) Anomaly detection from hyperspectral remote sensing imagery. Geosciences 6(4):56
2. Hu X, Xie C, Fan Z, Duan Q, Zhang D, Jiang L, Wei X, Hong D, Li G, Zeng X et al (2022) Hyperspectral anomaly detection using deep learning: a review. Remote Sens 14(9):1973
3. Racetin I, Krtalić A (2021) Systematic review of anomaly detection in hyperspectral remote sensing applications. Appl Sci 11(11):4878
4. Su H, Wu Z, Zhang H, Du Q (2021) Hyperspectral anomaly detection: a survey. IEEE Geosci Remote Sens Mag 10(1):64–90
5. Han W, Zhang X, Wang Y, Wang L, Huang X, Li J, Wang S, Chen W, Li X, Feng R et al (2023) A survey of machine learning and deep learning in remote sensing of geological environment: challenges, advances, and opportunities. ISPRS J Photogramm Remote Sens 202:87–113
6. Di, L., Yu, E.: Challenges and opportunities in the remote sensing big data. Remote Sens Big Data, pp 281–291 (2023)
7. Cavallaro G, Heras DB, Wu Z, Maskey M, Lopez S, Gawron P, Coca M, Datcu M (2022) High-performance and disruptive computing in remote sensing: Hdcrs-a new working group of the grss

earth science informatics technical committee [technical committees]. IEEE Geosci Remote Sens Mag 10(2):329–345
8. Plaza A, Du Q, Chang Y-L, King RL (2011) Foreword to the special issue on high performance computing in earth observation and remote sensing. IEEE J Select Top Appl Earth Observ Remote Sens 4(3):503–507
9. Dagum L, Menon R (1998) Openmp: an industry standard api for shared-memory programming. IEEE Comput Sci Eng 5(1):46–55
10. Dongarra J, Walker D, Lusk E, Knighten B, Snir M, Geist A, Otto S, Hempel R, Lusk E, Gropp W et al (1994) Mpi-a message-passing interface standard. Int J Supercomput Appl High Perform Comput 8(3–4):165
11. NVIDIA, Vingelmann P, Fitzek, FHP (2020) CUDA, release: 10.2.89 https://developer.nvidia.com/cuda-toolkit
12. Munshi A (2009) The opencl specification. In: 2009 IEEE Hot Chips 21 Symposium (HCS), pp 1–314 IEEE
13. Christophe E, Michel J, Inglada J (2011) Remote sensing processing: from multicore to gpu. IEEE J Select Top Appl Earth Observ Remote Sens 4(3):643–652
14. Ordóñez, Á., Heras, D.B., Argüello, F.: Multi-gpu registration of high-resolution multispectral images using hsi-kaze in a cluster system. In: IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium, pp. 5527–5530 (2022). IEEE
15. Garea AS, Heras DB, Argüello F, Demir B (2023) A hybrid cuda, openmp, and mpi parallel tca-based domain adaptation for classification of very high-resolution remote sensing images. J Supercomput 79(7):7513–7532
16. Haut JM, Bernabé S, Paoletti ME, Fernandez-Beltran R, Plaza A, Plaza J (2018) Low-high-power consumption architectures for deep-learning models applied to hyperspectral image classification. IEEE Geosci Remote Sens Lett 16(5):776–780
17. NVIDIA (2022) Nvidia jetson agx orin series technical brief v1.2. Technical report, NVIDIA https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf
18. Liu J, Xiang J, Jin Y, Liu R, Yan J, Wang L (2021) Boost precision agriculture with unmanned aerial vehicle remote sensing and edge intelligence: A survey. Remote Sens 13(21):4387
19. Rhee DS, Kim YD, Kang B, Kim D (2018) Applications of unmanned aerial vehicles in fluvial remote sensing: an overview of recent achievements. KSCE J Civil Eng 22:588–602
20. Argüello F, Heras DB, Garea AS, Quesada-Barriuso P (2021) Watershed monitoring in galicia from uav multispectral imagery using advanced texture methods. Remote Sens 13(14):2687
21. Gxokwe S, Dube T, Mazvimavi D (2020) Multispectral remote sensing of wetlands in semi-arid and arid areas: a review on applications, challenges and possible future research directions. Remote Sens 12(24):4190
22. Ma N, Peng Y, Wang S, Leong PH (2018) An unsupervised deep hyperspectral anomaly detector. Sensors 18(3):693
23. Truax DD (2004) Comparing spectral and object based approaches for classification and transportation feature extraction from high resolution multispectral imagery
24. Leon-Lopez KM, Mouret F, Arguello H, Tourneret J-Y (2021) Anomaly detection and classification in multispectral time series based on hidden markov models. IEEE Trans Geosci Remote Sens 60:1–11
25. Mouret F, Albughdadi M, Duthoit S, Kouamé D, Rieu G, Tourneret J-Y (2021) Outlier detection at the parcel-level in wheat and rapeseed crops using multispectral and sar time series. Remote Sens 13(5):956
26. Coca M, Datcu M (2021) Anomaly detection in post fire assessment. In: 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, pp 8620–8623 IEEE
27. Reed IS, Yu X (1990) Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution. IEEE Trans Acoust Speech Signal Process 38(10):1760–1770
28. Zhou J, Kwan C, Ayhan B, Eismann MT (2016) A novel cluster kernel rx algorithm for anomaly and change detection using hyperspectral images. IEEE Trans Geosci Remote Sens 54(11):6497–6504
29. Imani M (2017) Rx anomaly detector with rectified background. IEEE Geosci Remote Sens Lett 14(8):1313–1317
30. Yang X, Huang X, Zhu M, Xu S, Liu Y (2022) Ensemble and random rx with multiple features anomaly detector for hyperspectral image. IEEE Geosci Remote Sens Lett 19:1–5

31. Molero JM, Garzón EM, García I, Plaza A (2012) Anomaly detection based on a parallel kernel rx algorithm for multicore platforms. J Appl Remote Sens 6(1):061503–061503
32. Imani M (2018) Anomaly detection using morphology-based collaborative representation in hyperspectral imagery. Eur J Remote Sens 51(1):457–471
33. Dalla Mura M, Villa A, Benediktsson JA, Chanussot J, Bruzzone L (2010) Classification of hyperspectral images by using extended morphological attribute profiles and independent component analysis. IEEE Geosci Remote Sens Lett 8(3):542–546
34. Liu C, Tao R, Li W, Zhang M, Sun W, Du Q (2020) Joint classification of hyperspectral and multispectral images for mapping coastal wetlands. IEEE J Select Top Appl Earth Observ Remote Sens 14:982–996
35. Gao Y, Li W, Zhang M, Wang J, Sun W, Tao R, Du Q (2021) Hyperspectral and multispectral classification for coastal wetland using depthwise feature interaction network. IEEE Trans Geosci Remote Sens 60:1–15
36. Ghamisi P, Souza R, Benediktsson JA, Zhu XX, Rittner L, Lotufo RA (2016) Extinction profiles for the classification of remote sensing data. IEEE Trans Geosci Remote Sens 54(10):5631–5645
37. Dalla Mura M, Benediktsson JA, Waske B, Bruzzone L (2010) Morphological attribute profiles for the analysis of very high resolution images. IEEE Trans Geosci Remote Sens 48(10):3747–3762
38. Bascoy PG, Quesada-Barriuso P, Heras DB, Argüello F, Demir B, Bruzzone L (2019) Extended attribute profiles on gpu applied to hyperspectral image classification. J Supercomput 75:1565–1579
39. Molero JM, Garzón EM, García I, Quintana-Ortí ES, Plaza A (2014) Efficient implementation of hyperspectral anomaly detection techniques on gpus and multicore processors. IEEE J Select Top Appl Earth Observ Remote Sens 7(6):2256–2266
40. Wu Y, Gao L, Zhang B, Yang B, Chen Z (2015) Embedded gpu implementation of anomaly detection for hyperspectral images. In: High-Performance Computing in Remote Sensing 9646: 66–71 SPIE
41. Du Q, Tang B, Xie W, Li W (2021) Parallel and distributed computing for anomaly detection from hyperspectral remote sensing imagery. Proc IEEE 109(8):1306–1319
42. Caba J, Díaz M, Barba J, Guerra R, Escolar S, López S (2022) Low-power hyperspectral anomaly detector implementation in cost-optimized fpga devices. IEEE J Select Top Appl Earth Observ Remote Sens 15:2379–2393
43. Coca M, Datcu M (2023) Fpga accelerator for meta-recognition anomaly detection: Case of burned area detection. IEEE J Select Top Appl Earth Observ Remote Sens
44. Díaz M, Guerra R, Horstrand P, López S, Sarmiento R (2019) A line-by-line fast anomaly detector for hyperspectral imagery. IEEE Trans Geosci Remote Sens 57(11):8968–8982
45. Tarabalka Y, Haavardsholm TV, Kåsen I, Skauli T (2009) Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and gpu processing. J Real-Time Image Proc 4:287–300
46. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybernet 9(1):62–66
47. Chang C-I, Lin C-Y, Chung P-C, Hu PF (2023) Iterative spectral-spatial hyperspectral anomaly detection. IEEE Trans Geosci Remote Sens 61:1–30
48. CESGA: FinisTerrae-II supercomputer. Accessed: 03 Jan 2022. https://www.cesga.es/en/infrastructures/computing/
49. NVIDIA CORPORATION 'I &' AFFILIATES: Jetson Orin Nano Series, Jetson Orin NX Series and Jetson AGX Orin Series: Supported Modes and Power Efficiency. https://docs.nvidia.com/jetson/archives/r35.3.1/DeveloperGuide/text/SD/PlatformPowerAndPerformance/JetsonOrinNanoSeriesJetsonOrinNxSeriesAndJetsonAgxOrinSeries.html Accessed 2023-09-15
50. Han C, Rundo L, Murao K, Noguchi T, Shimahara Y, Milacski ZÁ, Koshino S, Sala E, Nakayama H, Satoh S (2021) Madgan: unsupervised medical anomaly detection gan using multiple adjacent brain mri slice reconstruction. BMC Bioinformat 22(2):1–20
51. Han Y, Li W, Liu M, Wu Z, Zhang F, Liu X, Tao L, Li X, Guo X (2021) Application of an anomaly detection model to screen for ocular diseases using color retinal fundus images: design and evaluation study. J Med Int Res 23(7):27822
52. Huang C, Xu Q, Wang Y, Wang Y, Zhang Y (2022) Self-supervised masking for unsupervised anomaly detection and localization. IEEE Trans Multimed