





## Article

# A Parallel Unmixing-Based Content Retrieval System for Distributed Hyperspectral Imagery Repository on Cloud Computing Platforms

Peng Zheng <sup>1</sup> , Zebin Wu <sup>1,\*</sup> , Jin Sun <sup>1</sup> , Yi Zhang <sup>1</sup>, Yaoqin Zhu <sup>1</sup>, Yuan Shen <sup>1</sup>, Jiandong Yang <sup>2</sup>, Zhihui Wei <sup>1</sup> and Antonio Plaza <sup>3</sup> 

<sup>1</sup> School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China; 318106020161@njjust.edu.cn (P.Z.); sunj@njjust.edu.cn (J.S.); yzhang@njjust.edu.cn (Y.Z.); zhuyaoqin@njjust.edu.cn (Y.Z.); shenyuan@njjust.edu.cn (Y.S.); gswei@njjust.edu.cn (Z.W.)

<sup>2</sup> China Satellite Maritime Tracking and Control Department, Jiangyin 214431, China; yjdjs250@126.com

<sup>3</sup> Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, 10071 Cáceres, Spain; aplaza@unex.es

\* Correspondence: wuzb@njjust.edu.cn

**Abstract:** As the volume of remotely sensed data grows significantly, content-based image retrieval (CBIR) becomes increasingly important, especially for cloud computing platforms that facilitate processing and storing big data in a parallel and distributed way. This paper proposes a novel parallel CBIR system for hyperspectral image (HSI) repository on cloud computing platforms under the guide of unmixed spectral information, i.e., endmembers and their associated fractional abundances, to retrieve hyperspectral scenes. However, existing unmixing methods would suffer extremely high computational burden when extracting meta-data from large-scale HSI data. To address this limitation, we implement a distributed and parallel unmixing method that operates on cloud computing platforms in parallel for accelerating the unmixing processing flow. In addition, we implement a global standard distributed HSI repository equipped with a large spectral library in a software-as-a-service mode, providing users with HSI storage, management, and retrieval services through web interfaces. Furthermore, the parallel implementation of unmixing processing is incorporated into the CBIR system to establish the parallel unmixing-based content retrieval system. The performance of our proposed parallel CBIR system was verified in terms of both unmixing efficiency and accuracy.

**Keywords:** content-based image retrieval; cloud computing; unmixing; hyperspectral images



**Citation:** Zheng, P.; Wu, Z.; Sun, J.; Zhang, Y.; Zhu, Y.; Shen, Y.; Yang, J.; Wei, Z.; Plaza, A. A Parallel Unmixing-Based Content Retrieval System for Distributed Hyperspectral Imagery Repository on Cloud Computing Platforms. *Remote Sens.* **2021**, *13*, 176.  
<https://doi.org/10.3390/rs13020176>

Received: 18 November 2020

Accepted: 4 January 2021

Published: 6 January 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Hyperspectral remote sensing, which is capable of measuring and interpreting on earth objects without contact detection, has been employed as an important technique for governments and scientific research institutions to achieve earth object information directly and quickly [1]. Today, hyperspectral remote sensing has played a critically important role in many fields, such as mineral detection [2], ecological environment investigation and detection [3] (water environment [4], air pollution [5], soil erosion [6], vegetation detection [7], etc.), dynamic monitoring of marine [8], atmospheric [9], land [10], and military application [11].

With the innovation of hyperspectral remote sensing technology, the spatial resolution and spectral resolution of hyperspectral images (HSIs) are continuously enhanced [12]. At present, there are hundreds of multi-resolution hyperspectral remote sensing satellites and sensors running in the air and space to provide continual hyperspectral remote sensing data. As far as the spatial resolution is concerned, it ranges from a few kilometers to a few meters. For instance, National Oceanic and Atmospheric Administration (NOAA) [13] and Moderate Resolution Imaging Spectroradiometer (MODIS) [14], which belong to the

category of low-resolution satellites, operate with spatial resolutions of 1 km and 250 m, respectively. Orbview-5 operates with a higher spatial resolution of 0.41 m and, thus, can be classified into high-resolution satellites. For the spectral resolution, the HSI resolution has even reached nanoscale. The spectrum ranges are broadened to near infrared and short waves infrared, which are beyond the spectral bands of ultraviolet rays and visible light. Additionally, the bandwidth of the HSI is narrower than before, resulting in a richer information vector for the HSI. For instance, the resolution of 224 spectral bands gained through Airborne Visible Infrared Imaging Spectrometer (AVIRIS) by National Aeronautics and Space Administration (NASA) is 0.2  $\mu\text{m}$  to 2.4  $\mu\text{m}$ , whereas the data offered by CHRIS that contains 63 bands with spectral resolutions of 1.3  $\mu\text{m}$  to 12  $\mu\text{m}$ . Motivated by the above-mentioned factors, the hyperspectral remote sensing data has developed to a huge scale after many years of accumulation.

It is obvious that we have entered an era of big hyperspectral remote sensing data. With the rapid advances in sensors and hyperspectral techniques, the hyperspectral remote sensing data develops in a multi-directional way gradually. They spread around the world and are in requirement of unity for data types and storage standard, which has brought inconvenience to the transmission and storage of data. A standard hyperspectral imagery repository for large-scale data with retrieval function is urgently required. Sevilla and Plaza proposed a common repository of hyperspectral data including a spectral unmixing-guided content-based image retrieval (CBIR) functionality on graphics processing units (GPUs) intended to distribute and share free hyperspectral data sets in the community [15]. However, the file transfer protocol (FTP) used in this method is not suitable for the storage of large-scale data which exceeds the volume that a single machine can hold. The management and process abilities of hyperspectral data are further influenced.

Undoubtedly, the high diversity and volume of hyperspectral data put us into a dilemma when we process and storage the data of large-scale HSI data. The bottleneck may appear in processing hyperspectral remote sensing data (e.g., classification [16] and unmixing [17]) in traditional single-server environment. In recent years, great efforts have been made towards the research on high-performance computing (HPC) of specialized hardware devices (e.g., field-programmable gate arrays (FPGAs), Beowulf clusters and distributed computers, multicore central processing units (CPUs), and graphics processing units (GPUs)) in hyperspectral applications [18]. It is an effective way to construct HPC systems to improve computational efficiency of hyperspectral data processing, especially when the hyperspectral data processing algorithms and applications are highly heterogeneous. For example, Zheng et al. [19] have designed and implemented parallel computing methods of pre-processing the time-consuming interpolation based on OpenMP programming model. These methods can significantly improve CPU efficiency and reduce the runtime of the aerosol optical depth retrieval application. In contrast, benefiting from GPUs' intensive computation capacity, GPU-based parallel computing is a more powerful parallel way to accelerate the HSI processing. For instance, Sevilla et al. [15] have developed efficient implementations of spectral unmixing algorithms on GPUs to obtain HSI spectral information.

Although HPC systems have a powerful computing capacities, there are still many challenging problems to tackle. Today, the hyperspectral remote sensing data are experiencing rapid growth and constantly increasing daily analyzing and processing amount. Existing HPC systems have severe bottlenecks and difficulties in the storage of increasing HSI data scale even they have enhanced computing capacity. To overcome the inherent weakness and deficiency in traditional data storage mechanisms, extensive researches on big data storage have been conducted in academic and industrial fields. To make the file systems store more and larger-scale data, block storage is typically used in storage-area network (SAN) [20] environments, among which redundant arrays of independent drives (RAID) [21] technique is a representative use case. With RAID, multiple independent disks are combined for data protection and performance. However, the high cost and poor scalability of SAN systems make them no longer meet the requirements of cluster

equipped with thousands of CPUs. Another way of shared storage is network attached storage (NAS) [22] that achieves more powerful expandability and higher cost-effectiveness. At present, NFS file systems are widely used in computing clusters. However, the high protocol overhead, low bandwidth, and large latency of NAS have limited the application of high-performance clusters.

It is challenging to deal with the storage and processing of big remote sensing data simultaneously [23]. Fortunately, cloud computing emerges in recent years. Cloud computing is highly promising in the hyperspectral field and offers an effective solution to the storage and processing of HSIs due to its scalable storage capabilities and high-performance computing capacity [24]. Wu et al. [25] have developed a parallel and distributed implementation of principal component analysis (PCA), which is a widely-used technique for hyperspectral dimensionality reduction. This implementation is built upon cloud computing and has demonstrated very high performance. In our previous work [23], a parallel CBIR system is proposed, but it is lack of more dynamic cloud resource management (e.g., using restricted virtual resources management softwares) and more endmember information extraction algorithms. Nevertheless, to the best of our knowledge, there are very few efforts in the literature on constructing unified and general-purpose cloud computing architecture for hyperspectral remote imagery.

In this paper, we propose a novel parallel CBIR system for hyperspectral image (HSI) repository on cloud computing platforms under the guide of unmixed spectral information as a extension of [23]. To accelerate spectral information extraction, we implement a distributed and parallel unmixing method that operates on cloud computing platforms for accelerating the unmixing processing flow. In addition, we implement a global standard distributed HSI repository equipped with a large spectral library in a software-as-a-service mode, providing users with HSI storage, management, and retrieval services through web interfaces. Moreover, the unmixing processing is verified to achieve high efficiency with an up to  $31.5\times$  speedup.

The rest of this paper is organized as follows. Section 2 describes the hyperspectral imagery CBIR system based on cloud computing architecture released in SaaS mode, which is composed of three main layers: (1) the device layer, in charge of providing various underlying compute and storage resources; (2) the service layer, which is responsible for storage, management and processing of HSIs; (3) and the application layer, which provides users with secure remote access to the repository. Section 3 presents a method of hyperspectral imagery distributed retrieval based on parallel unmixing algorithms on Spark. Section 4 presents an experimental validation of the CBIR system. Finally, Section 5 summarizes the whole paper and discusses future work directions.

## 2. Cloud-Computing-Based CBIR Framework in SaaS Mode for Hyperspectral Remote Imagery Repository

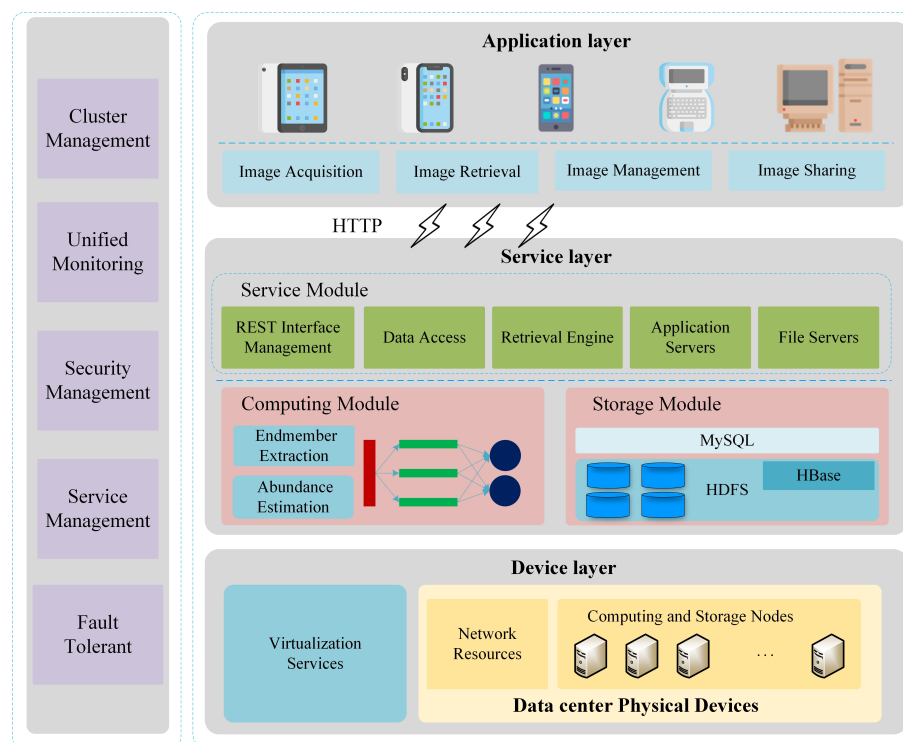
This section describes our study focused on the HSI repository and parallel CBIR method on cloud-computing-based architecture in SaaS mode.

### 2.1. Cloud-Computing-Based CBIR Architecture in SaaS Mode

In order to overcome the shortcomings of existing hyperspectral image retrieval systems, Figure 1 illustrates a cloud-computing-based CBIR architecture in SaaS mode for HSI repository. As a well-developed cloud computing model, SaaS is generally applied in various fields. SaaS applications are hosted in the cloud and support effective storage and computing capacity. In the SaaS mode, software application services for hyperspectral remote imagery, e.g., HSI storage and processing are provided to users through Internet by using a thin client, e.g., a web browser, which can only purchase software services on demand. In this mode, the users can ignore the problems, such as deployment, operations, and software upgrades.

Our proposed architecture can be divided into two parts: management module and service module. The main functionality of management module is to ensure that the whole cloud computing center can operate safely and stably. The management module

plays an important role in applying service management, security, monitoring, resource maintenance, etc. Service module mainly provides applications of hyperspectral image retrieval to customers in a Web-based manner. To make every part more clear and to make the service module simple yet with high maintainability, we divide the entire framework into three layers: device layer, service layer, and application layer.



**Figure 1.** The cloud-computing-based content-based image retrieval (CBIR) architecture in SaaS mode for hyperspectral image (HSI) repository.

The cloud computing framework starts at the most fundamental device layer of storage and server infrastructure. The device layer provides necessary physical resources for the whole cloud platform. It consists of physical assets, such as servers, network devices, and storage disks. Through virtualization technology [26], these physical resources form a virtual resources pool from physics. In virtualization technology, a physical server can provide user with several virtual machines, which have diverse computing capability and are independent from each other. In this manner, the framework's efficiency, flexibility and resource utilization can be enhanced to lower IT costs. In addition, the software and hardware are separated after virtualization so that users can deploy their applications without considering the underlying architecture. In brief, the device layer offers back-end support for the upper layers and provides a highly available and scalable resource environment for upper-layer applications.

The service layer of the framework is mainly composed of hyperspectral remote sensing big data storage module, computing module and service module. This layer is the most important part of the architecture and is mainly responsible for executing parallel unmixing algorithms with high computational cost and storage of huge HSIs. For distributing computing, we sort to Apache Spark as the computing engine for parallel processing. Apache Spark [27] is a versatile large-scale data rapid computing framework for big data on large clusters and has been widely used in academia and industry. It inherits the advantages of Apache Hadoop, a previous distributed computing framework, and is far more powerful than Apache Hadoop. Spark's computational model is similar to Hadoop's, using Map-Reduce. But considering the low efficiency of the Map-Reduce iterative algorithms due to frequent IO operations, Spark builds an integrated and diversified big data processing system which implements a fault-tolerant abstraction for

in-memory cluster and proposes a creative data structure namely resilient distributed datasets (RDDs). To obtain the computing power of Spark cluster, developers can only call uniform standard application programming interfaces (APIs) to do RDDs action and transformation in Scala, Python, or Java which improves the convenience and effectiveness greatly. Through the action and transformation operations, RDDs can form a series of dependent lineage. If a data in the lineage is incorrect or missing, lineage can implement transaction rollback through a checkpoint mechanism with high fault tolerance. Through the distributing computing power provided by Spark cluster, the overhead of the Web servers is reduced considerably. For storage, we sort to Hadoop Distributed File System (HDFS) [28] as bottom support for the distributing storage. HDFS is the core sub-project and an important part of Hadoop. It is deployed in hundreds of low-cost servers and provides outstanding scalability and capacity with high throughput IO. A HDFS cluster runs in master/slaves mode and primarily consists of a NameNode that manages the file system metadata, a Secondary NameNode that helps NameNode and DataNodes that store the actual data. Through common interfaces, Spark imports data from HDFS and transforms to RDDs for the following execution. Besides, the layer provides Web service and Database service. It plays a significant part as a bond between the service layer and the application layer through calling program. The application service archives user requests from the application layer and invokes the computing and storage services by Shell. The executions results will be delivered to the application layer.

The application layer of the framework is responsible for the deployment and execution of CBIR applications. This layer defines the interaction between the end users and CBIR system over the HTTP protocol, and provides users with secure remote access to the system including image acquisition, image retrieval, image management, image sharing, etc., in a dashboard through a Web browser. The application is deployed in a Tomcat servlet container running in web servers and is implemented with SpringMVC, an original web framework built on the Servlet API. As for the front end, a progressive framework named Vue for building user interfaces is selected to form an interactive and a beautiful user interface. The frontend communicates data with the backend through unified restful style service's APIs with JavaScript Object Notation (JSON) format asynchronously.

## 2.2. Storage Method for Hyperspectral Remote Images

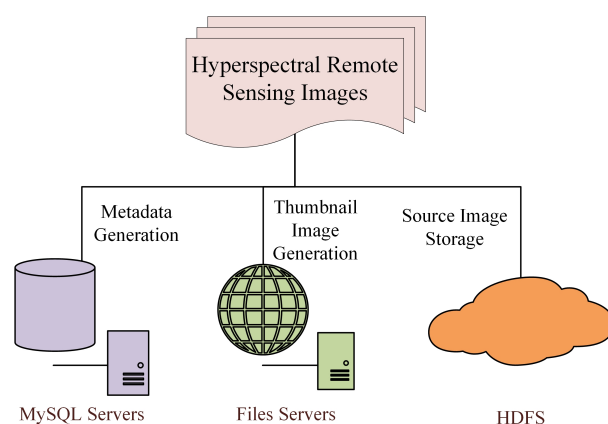
The main HSIs storage structure is presented in Figure 2. In order to build a scalable and efficient HSI repository, it is essential to have data well organized. The storage structure consists of two parts: The first part is the design of metadata storage of HSIs, and the other part is the design of the storage of original HSIs based on HDFS. To facilitate the reading of original files, the relationship between HDFS and file servers is established through a hyperlink stored in MySQL, whereas the HSI thumbnails are stored in files servers.

Metadata information is descriptive information for hyperspectral data and plays a very important role in the field of remote sensing data especially when the data is of large size and hard to analyze in a timely way. Massive HSIs have a wide range of sources and the data standard from different HSIs providers is not unified. Most providers provide HSIs with specific independent header files to store these metadata. Therefore, it is particularly important to design a unified metadata structure of HSIs reasonably in the process of collecting hyperspectral data through analyzing the header file format of HSIs from different organizations. There are many choices of storage types, such as JSON, which is convenient for Web operation. In order to access these relatively small metadata efficiently, we carefully analyze the characteristics of metadata access and consider the relational database MySQL to store HSIs metadata.

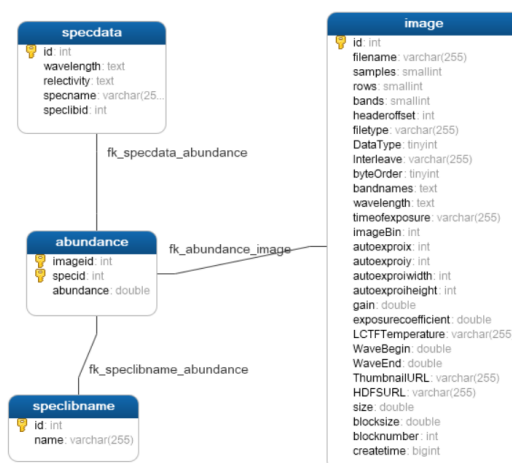
The main database tables for HSIs retrieval and relations between them are shown in Figure 3. In the database, not only general information including image name, number of samples, data type, bands, lines, wave information, thumbnail, etc. (refer table *image* in Figure 3), but also the metadata of endmembers and abundances information (refer to table *abundance* in Figure 3) is stored. The spectral information and relevant basic



information about the scene are obtained by parallel unmixing algorithms and stored in MySQL automatically in advance.



**Figure 2.** The storage framework structure of HSI repository.



**Figure 3.** The main database tables relations for hyperspectral remote sensing images retrieval.

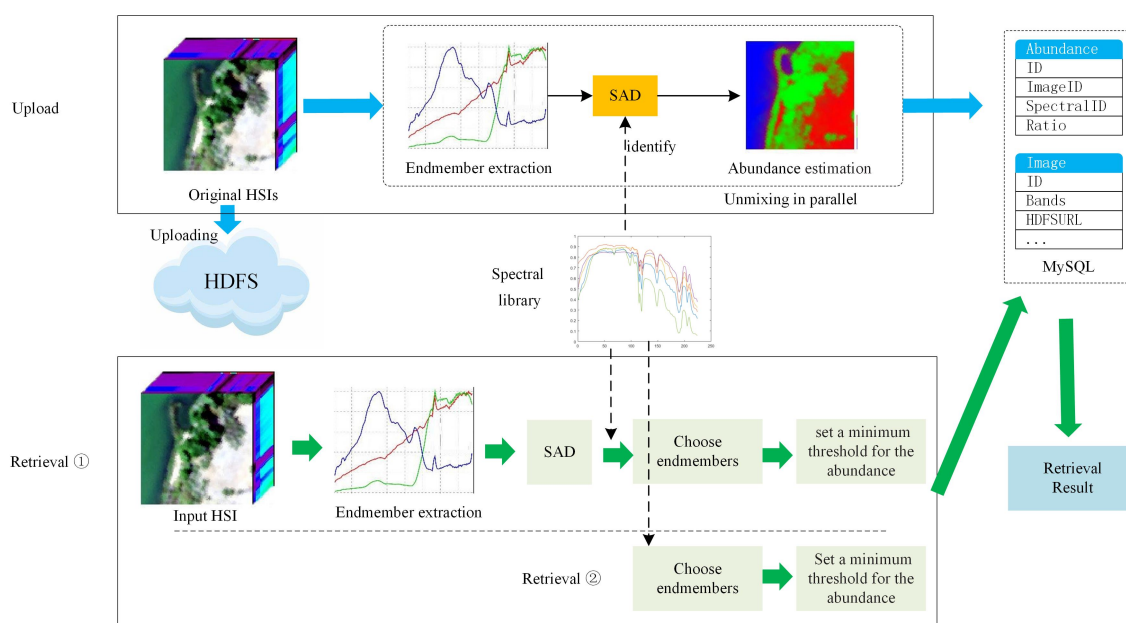
The other part is the storage of original HSIs in HDFS. This part provides distributed storage support of massive HSIs for parallel computing in HSIs retrieval process. Considering the diversity of the data sources, we attentively analyze the storage format of different HSIs. There are mainly three kinds of storage format including band interleaved by pixel format (BIP), band interleaved by line format (BIF), and band sequential format (BSQ) [29]. Specific algorithms need data in different formats that fit them, so data conversion between BIP, BSQ, and BIF will bring additional overhead and greatly reduce the efficiency of unmixing execution. In order to reduce the consumption of data conversion, as well, we pre-convert the hyperspectral remote sensing images in three data formats. Although there will be some data redundancy in this way, the cost of disks is negligible.

### 2.3. Method of the CBIR System

The early research of HSI repository, which basically relied on text to retrieve hyperspectral imagery data [24], was limited by the computer performance. It mainly archived and indexed information, such as the location and time of image acquisition, which is regarded as the metadata of hyperspectral imagery data to facilitate the research and analysis of remote sensing image retrieval. With the continuous development of content-based image retrieval of traditional images, the content-based image retrieval technique is also applied to hyperspectral remote imagery repository since the mid-1990s [30].

Up to present, content-based image retrieval is still a research hotspot in hyperspectral field. Although HSIs are different from traditional three-channel images, they still maintain the characteristics of traditional image. Therefore, some methods and techniques based on traditional image retrieval can still be applied to hyperspectral remote image retrieval. Combined with the unique characteristics of hyperspectral remote image, such as spectral characteristics, the research on hyperspectral remote image retrieval makes good effect. One popular strategy is proposed in Reference [31], which present a spectral/spatial CBIR system for HSIs. One main problem the author solved involved in the CBIR system is spectral unmixing, which includes two steps: (1) using endmember extraction algorithms to extract a set of image spectral features which called endmembers; (2) complete abundance inversion to obtain spatial features. And then these two kinds of features are integrated together, which guides to the queries to the database to retrieve the image meeting the spectral of certain conditions.

In our paper, we follow the strategy in Reference [31] and design an unmixing-based parallel CBIR method based on cloud computing for higher efficient. First, the HSIs' information extraction is accomplished in advance whiling uploading HSIs to the repository. The endmember signatures are extracted using pixel purity index (PPI) [32] in parallel and then the spectral angle distance (SAD) algorithm is considered to identify the pure endmembers by comparing with the spectral library. A particular issue may arise that the wavelengths of the HSIs are different from the wavelengths of the spectral library used as input. For this purpose, we have implemented a linear interpolation strategy that looks for wavelength values which are present in both the extracted endmembers and the spectral library. Further the abundance estimation is conducted using sum-to-one constrained least squares (SCLS) in parallel. The information provided by endmember identification and abundance estimation is stored in table image and abundance as metadata to catalog each image. Hence when retrieving hyperspectral remote sensing image in the repository, just start with querying the metadata information stored in MySQL database (Figure 4).



**Figure 4.** The unmixing-based CBIR flowchart of HSIs in the proposed system.







There are two retrieval options available in the proposed system. The first one is query by hyperspectral remote sensing image directly. From the user's perspective, a standard retrieval procedure in the system can be summarized as follows.

1. Image input. In this step, an image to be retrieve is uploaded to HDFS.
2. Endmember extraction. The system extracts the image's endmember information automatically.

3. Signature comparison. For all spectral signatures in the spectral library, the system calculates the SAD with all the endmembers extracted in step 2 and identifies the extracted endmembers according to SAD scores.
4. Abundance filtration. One or many endmembers interested in are chosen and minimum abundance filters for each endmember are defined. In this case, the image is retrieved only if the matched endmember contains a higher total abundance in the scene than the predefined minimum abundance threshold.
5. Results display and manual selection. The retrieved hyperspectral remote images are showed in table form in the web for users to select.

Another option is query by endmembers directly provided by the spectral library. Compared with the previous option, this option skips steps 1 and 2 and avoids the complicated process of unmixing, which makes query more convenient and efficient.

In what follows, we provide a simple step-by-step example to demonstrate how to perform a simple hyperspectral image retrieval in our system. Figure 5a shows the list of HSIs in our system with their general metadata. Click the *View* command button and then the HSI's information including all metadata, thumbnail, its endmembers' spectrum, and the ratio of every endmember will be displayed in detail, as shown in Figure 5b. To expand the HSI repository, the system provides an interface for uploading. Once an HSI is uploaded, the users can decide which unmixing algorithms to extract the spectral information. As shown in Figure 5c, the PPI, SAD, and SCLS algorithms based on Spark were chose which can be executed in appointed cluster in the cloud framework. In addition, the parameters including Driver-Memory, Executor-Memory, and Executor-Cores for task execution can also be specified in the web page. After the unmixing algorithms have been executed, the HSI spectral information obtained as a complement will be automatically cataloged in the database described in Figure 4. Figure 5d shows a querying example, in which we specified two spectral signatures (clinoclione\_Fe GDS157 and Eugsterite GDS140 Syn) from United States Geological Survey (USGS) library as the query criteria. The minimum abundance is set to 0.1 that means the HSIs we are looking forward to contain at least 10% clinoclione\_Fe GDS157 and 10% Eugsterite GDS140 Syn. The query result will be shown as a list, and corresponding HSI's information can be viewed by click the *View* button. In this case, the HSI with id 41 contains 21.05% clinoclione\_Fe GDS157 and 18.25% Eugsterite GDS140 Syn, which satisfies the query criteria accurately. Readers who are interested in this work and would like to know more details are encouraged to download and run the source codes of the project hosted on Github [Github:git@github.com: ZpWaitingForSunshine/CBIR.git](https://github.com/ZpWaitingForSunshine/CBIR.git).

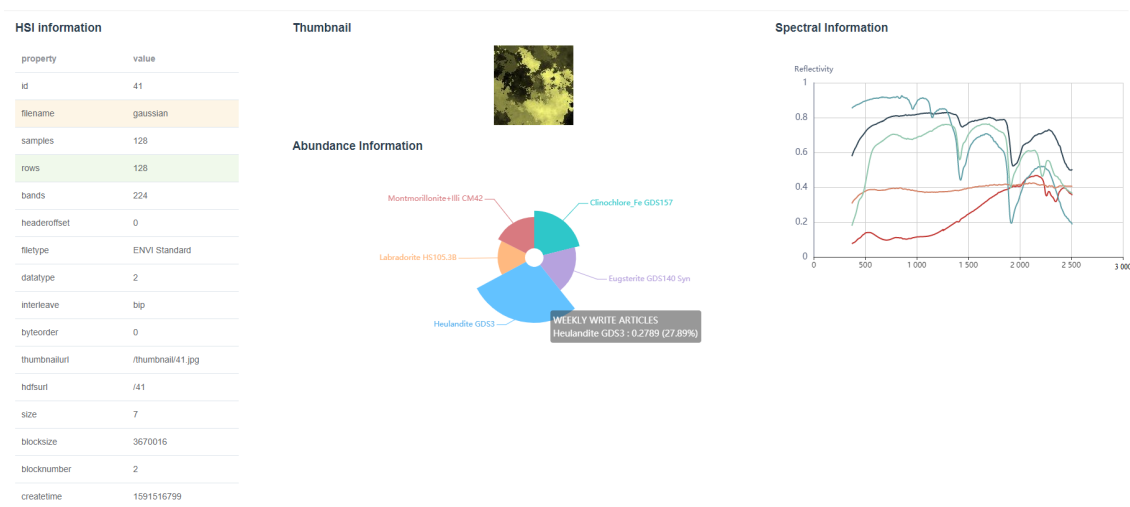
id	filename	samples	rows	bands	headeroffset	filetype	datatype	interleave	createTime	thumbnailurl	Actions
41	gaussian	128	128	224	0	ENVI Standard	2	bip	2020-06-07 15:59		<a href="#">View</a> <a href="#">Delete</a>
47	samson_2	95	95	156	0	ENVI Standard	2	bip	2020-07-08 12:22		<a href="#">View</a> <a href="#">Delete</a>
49	c350bip	350	350	192	0	ENVI Standard	2	bip	2020-07-08 12:32		<a href="#">View</a> <a href="#">Delete</a>
51	Indian_pines	145	145	200	0	ENVI Standard	12	bip	2020-07-08 12:58		<a href="#">View</a> <a href="#">Delete</a>
53	c188	190	250	188	0	ENVI Standard	2	bip	2020-07-08 13:04		<a href="#">View</a> <a href="#">Delete</a>
54	PaviaU	610	340	103	0	ENVI Standard	12	bip	2020-07-08 13:06		<a href="#">View</a> <a href="#">Delete</a>

共 46 条 10条/页 < 1 2 3 4 5 前往 5 页

(a) List of HSIs

Figure 5. Cont.





(b) HSI metadata display

**Image ID**

41

**Algorithm Selection**

Endmember Extraction: PPI

Signature Comparison: SAD

Abundance Fractions: SCLS

**Resource Allocation**

Master: spark://192.168.1.107

Diver Memory(G): 4

Executor Memory(G): 4

Executor-Cores: 2

**Submit**

(c) Unmixing algorithm execution settings

**Spectral Library**

- United States Geological Survey, USGS\_min
  - ☒ Clinocllore\_Fe GDS157
  - ☒ Eugsterite GDS140 Syn
  - ☐ Heulandite GDS3
  - ☐ Labradorite HS105\_3B
  - ☐ Montmorillonite+Illite CM42
- Jet Propulsion Laboratory, JPL
- International Gorilla Conservation Programme, IGCP-264
- Johns Hopkins University, JHU
- United States Geological Survey, USGS
- United States Geological Survey, USGS-veg

**Spectral Selected**

**Retrieval Result**

ImageID	Filename	Action
41	gaussian	<a href="#">View</a>

**Abundance Filter**

Clinocllore\_Fe: 0.1

GDS157

Eugsterite GDS140: 0.1

Syn

**Search**

(d) Example of a query

Figure 5. The CBIR system presentation.

### 3. Hyperspectral Imagery Distributed Retrieval

In this section, as a case study, we use a Spark-based parallel unmixing flow, including PPI, SAD, and SCLS algorithms, which are the key steps of hyperspectral remote sensing image distributed retrieval chain. The spectral features are the key information we make use of, so the hyperspectral remote sensing imagery unmixing is accomplished in advanced.

### 3.1. Parallel PPI

Let us assume that  $\mathcal{X}_{N \times L}$  is a hyperspectral image with  $N$  pixel vectors and  $L$  spectral bands and  $S$  is an  $L$ -dimensional random vector. The traditional PPI algorithm can be formulated as follows:

1. Generate a set  $\{skewer_j\}_{j=1}^Y$  of  $Y$  random skewers where  $skewer_j$  denotes a random vector.
2. Iterate  $Y$  times, for each  $skewer_j$ , we project all the pixel vectors onto this  $skewer_j$  to record those sample vectors that are at its extreme positions denoted as  $max_j = \max_{1 \leq j \leq N} \{X_{i \times L}^T s_j\}$  and  $min_j = \min_{1 \leq j \leq N} \{X_{i \times L}^T s_j\}$ . Let  $PPI(X_{i \times L}) = \{skewer_j, 1 \leq j \leq Y | X_{i \times L}^T \cdot skewer_j = max_j \text{ or } min_j\}$ .
3. Remove those pixels with low frequency, and spectral angle distance (SAD) is used to calculate the similarity between any two pixels in  $PPI(X_{i \times L})$  to remove those similar pixels. Finally, the all remaining pixels in  $PPI(X_{i \times L})$  are endmembers  $M$ .

In the processing flow of PPI algorithm [33], a large amount of calculation occurs in Step 2. And the computational complexity of this part is closely related to the number of random vectors  $skewer_j$ , and the number of random vectors directly determines the execution effort of the PPI algorithm. Therefore, we mainly optimize the calculation in Step 2 in parallel based on Spark, summarized in Algorithm 1. Built upon the cloud-computing-based CBIR framework described in Section 2 and the map/reduce paradigm, the distributed PPI algorithm can be implemented for endmember extraction of massive hyperspectral images by the following steps, which is shown in Figure 6.

1. We divide the original hyperspectral remote sensing image  $X_{N \times L}$  into  $m$  partitions and store them on HDFS in a distributed way automatically.
2. We read the hyperspectral dataset from HDFS to *ByteRdd* as a stream of bytes using built-in *newAPIHadoopFile()* method. After that, *ByteRdd* is converted into *DataRDD* according to the format of remote sensing image. It is noteworthy that the spectral data in *DataRDD* is complete spectral.
3. We generate  $Y$  random skewers in the driver and broadcast them to all executors through resource manager. Thus, the data  $X_{i \times L}$ ,  $i = \{1, 2, \dots, m\}$  in each partition shares the same skewers.
4. The following steps are similar to the steps of the traditional PPI algorithm. Perform the map operation on *DataRDD* and, in each executor, project the pixel onto each *skewer* to find pixels that are at its extreme positions to form a set  $E^i$  denoted by  $\{tmax_k, tmax\_p_k, tmin_k, tmin\_p_k\}_{1 \leq k \leq Y}^i$ , where  $tmax_k$  and  $tmin_k$  denote the maximal and minimal projections on every skewer, and  $tmax\_p_k$  and  $tmin\_p_k$  denote their corresponding positions in the entire HSI, respectively.
5. The reduce operation is conducted to submit all  $E^i$  in each partition to driver and compute the maximal and minimal projects on every skewer over again. The result will be cached in driver, denoted by  $E^{final} = \{tmax_k, tmax\_p_k, tmin_k, tmin\_p_k\}_{1 \leq k \leq Y}^{final}$ .
6. Count pixel purity index from  $\{tmax_k, tmax\_p_k, tmin_k, tmin\_p_k\}_{1 \leq k \leq Y}^{final}$  denoted by  $M_{all}$ , including and filter a part of endmembers in  $M_{all}$ , which is similar through SAD algorithm. Final, find the sample vectors  $M$  which are repeated most often.

**Algorithm 1:** Parallel PPI Based on Spark.

---

Map Stage:  
**Input:**  $X_{N \times L}$ ,  $S'_{Y \times L}$ ,  $b$   
**Output:**  $(E^i)$

```

1 for  $(i = 1; i \leq y; i++)$  do
2    $t = 0, tmax = 0, tmin = 0, tmax\_p = b, tmin\_p = b$ 
3   for  $j = 1; j \leq N/m; j++$  do
4     for  $p = 1; p \leq L; p++$  do
5        $t = t + X[j][p] \times S'[i][p]$ 
6     end
7     if  $tmax < t$  then
8        $tmax = t, tmax\_p = b + j$ 
9     end
10    if  $tmin > t$  then
11       $tmin = t, tmin\_p = b + j$ 
12    end
13  end
14   $E_{1i} = tmax; E_{2i} = tmax\_p; E_{3i} = tmin; E_{4i} = tmin\_p$ 
15 end

```

// Reduce Stage:  
**Input:**  $E^1, E^2, \dots, E^i, \dots, E^m$   
**Output:**  $(E)$

```

16 for  $i=1; i \leq m; i++$  do
17   for  $j=1; j \leq Y; j++$  do
18     if  $i == 1$  then
19        $E_{1j} = E_{1j}^i; E_{2j} = E_{2j}^i; E_{3j} = E_{3j}^i; E_{4j} = E_{4j}^i;$ 
20     end
21     else
22       if  $E_{1j} < E_{1j}^i$  then
23          $E_{1j} = E_{1j}^i, E_{2j} = E_{2j}^i$ 
24       end
25       if  $E_{3j} > E_{4j}^i$  then
26          $E_{3j} = E_{3j}^i, E_{4j} = E_{4j}^i$ 
27       end
28     end
29   end
30 end

```

---

**3.2. Parallel SAD Implemented**

After the processing in the previous step is completed, spectral feature matching of the endmember extraction results with the spectral library should be carried out. The spectral feature matching algorithm used in the system is based on spectral angle distance (SAD) method, which measures the distance between two spectral by calculating the SAD between two spectral vectors. Let  $e_i$  and  $e_j$  be two spectral vectors. The SAD value can be defined as

$$d_{SAD}(e_i, e_j) = \cos^{-1} \frac{e_i \cdot e_j}{\|e_i\| \cdot \|e_j\|}. \quad (1)$$

We assume  $\{e_i\}_{i=1}^N$  be the endmembers set of the spectral library and  $\{e_j\}_{j=1}^R$  be endmembers set of the extraction results where  $N$  is the total of the spectral library endmembers and  $R$  is the total of the extraction endmembers. The SAD algorithm can be implemented in parallel and distributed form by the following steps, which are graphically summarized in Figure 7.

1. First, the spectral library  $\{e_i\}_{i=1}^N$  is read from the MySQL server to a *LibRDD* instance which is divided into  $n$  partitions. Every partition contains a list of  $N/n$  elements which are tuples, including name, wavelength, and reflectance. It is obvious that  $R$  is always much smaller than  $N$ , so we broadcast  $\{e_j\}_{j=1}^R$  to computing nodes through Spark broadcast mechanism.
2. Perform the map operation on *LibRDD*. The map operation is in charge of calculating the SAD value between  $e_i$  in each partition and  $e_j$  in broadcast. In order to solve the problem that the wavelengths of the results are different from the wavelengths of the spectral library, we take wavelengths in the same range and align the spectrum by linear interpolation.
3. Conduct the reduce operation on *LibRDD* to filter out the minimum SAD value for each spectral vector  $e_j$  and identify the spectrum type.

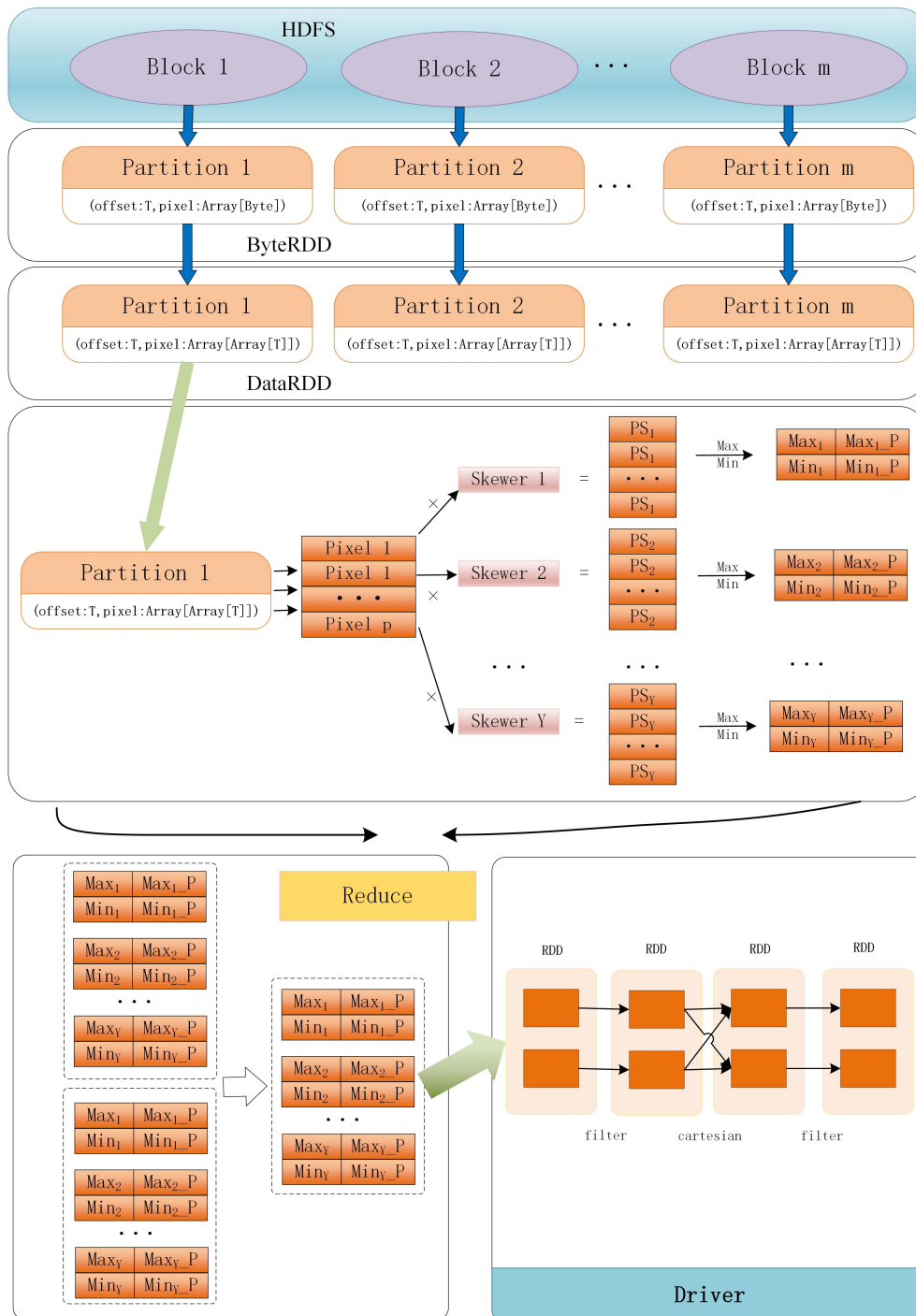
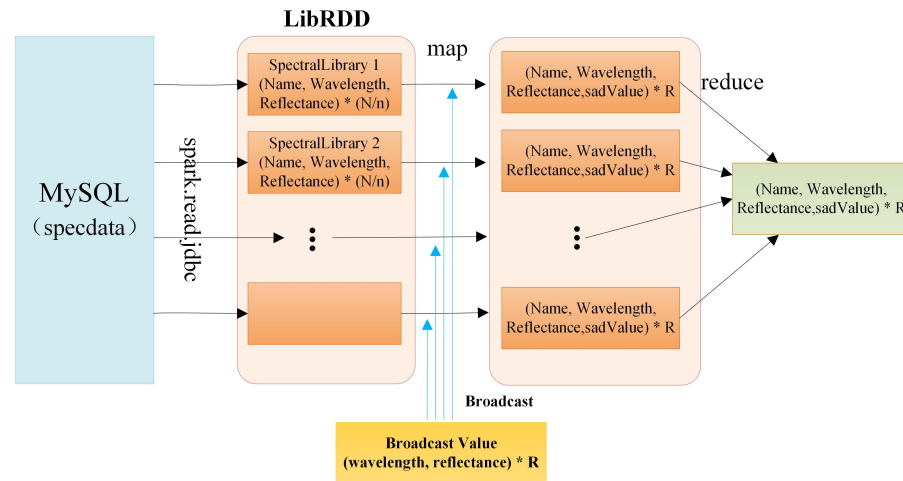


Figure 6. Parallel and distributed implementations of pixel purity index (PPI).



**Figure 7.** Parallel and distributed implementation of spectral angle distance (SAD).

### 3.3. Parallel Sum-to-One Constrained Least Squares (PSCLS) Abundance Inversion

The sum-to-one constrained least squares (SCLS) method based on linear mixed models is widely applied in abundance inversion. The mixing pixel reflectivity can be simply regarded as a linear combination of multiple endmembers' reflectivities, which can be formulated as

$$r = \sum_{i=1}^n m_i \alpha_i + \epsilon = M\alpha + \epsilon$$

$$s.t. \sum_{i=1}^n \alpha_i = 1, \alpha_i > 0,$$
(2)

where  $r$  is any  $L$ -dimensional vector in a hyperspectral image,  $M = [m_1, m_2, \dots, m_n]$  is endmember matrix with  $L$  spectral bands and  $n$  endmembers, and  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)^T$  represents abundance coefficient of a pixel, while  $\epsilon$  is a nonnegative and  $L$ -dimensional error term. Here, we must guarantee that the sum of the abundance coefficient is 1.

In this approach,  $\epsilon$  is expected as small as possible. Therefore, the optimization problem is formulated as:

$$\arg \min_{\theta} |\epsilon| = \arg \min_{\theta} \|r - M\alpha\| = \arg \min_{\theta} (r - M\alpha)^T (r - M\alpha)$$

$$s.t. \sum_{i=1}^n \alpha_i = 1, \alpha_i > 0.$$
(3)

Then, we get the estimation of abundance coefficient:

$$\hat{\alpha} = (M^T M)^{-1} M^T r.$$
(4)

After that, we introduce matrix  $N$  and vector  $s$  to convert Formula 5 as follows:

$$N = \begin{bmatrix} \delta M \\ I \end{bmatrix}, s = \begin{bmatrix} \delta r \\ 1 \end{bmatrix},$$
(5)

where  $I = (1, 1, \dots, 1)$  and  $\delta$  is used to normalize  $M$ , which we set  $10^{-5}$ . It looks like the hyperspectral image embraces an extra band, all of which are 1.

Correspondingly, we get the solution:

$$\hat{\alpha} = (N^T N)^{-1} N^T s.$$
(6)

In order to accelerate SCLS algorithm, the data parallel strategy is used based on Spark as shown in Figure 8 and Algorithm 2. First, we pre-process the endmember matrix  $M$



and  $\delta$  to form  $N$  in the driver. Then, we prepare  $pr_{NN} = (N^T N)^{-1} N^T$ , which is a fixed value. In order to reduce network traffic and the memory overhead, we broadcast  $pr_{NN}$  to computing nodes through Spark broadcast mechanism. Since the origin image data is divided into many partitions, we read every pixel data  $r$  in every partition on HDFS to form *DataRDD* instances. The pixels data are transformed into  $X'_{p \times L}$ , where  $p$  denotes the number of pixels in the partition. After that, perform the map operation to get *SCLSRDD*. The map operation is in charge of pre-processing  $X'_{p \times L}$  into  $X'_{p \times (L+1)}$  (which is similar to the operation on endmember matrix  $M$ ), and multiplying  $pr_{NN}$  broadcasted with each column in  $X'_{p \times (L+1)}$  to get the abundance coefficient of per pixel.

---

**Algorithm 2:** Parallel Sum-to-one Least Squares abundance inversion on Spark.

---

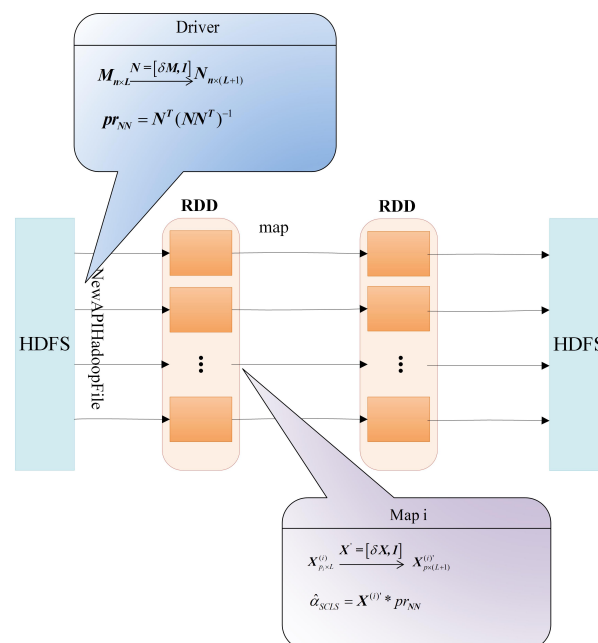
Map Stage:  
**Input:**  $X_{p \times L}, N_{n \times (L+1)}, \delta, n$   
**Output:**  $A$

```

1  $X'[1 \dots p][1 \dots L+1];$ 
2 for ( $i = 0; i < p; i++$ ) do
3   for ( $j = 0; j < L; j++$ ) do
4      $X'[i][j] = X[i][j] \times \delta$ 
5   end
6 end
7  $X'[:, L+1] = 1$ 
8 for  $i = 0; i < p; i++$  do
9   for  $j = 0; j < n; j++$  do
10     $t = 0; \text{for } t = 0; t < L+1; t++ \text{ do}$ 
11       $t = t + X'[i][j] \times N[t][j];$ 
12     $A[i][j] = t$ 
13  end
14 end
15 end
16 Reduce Stage:
Input:  $A^1, A^2, \dots, A^m$ 
Output:  $A$ 
17  $A = \{A^1, A^2, \dots, A^m\}$ 

```

---



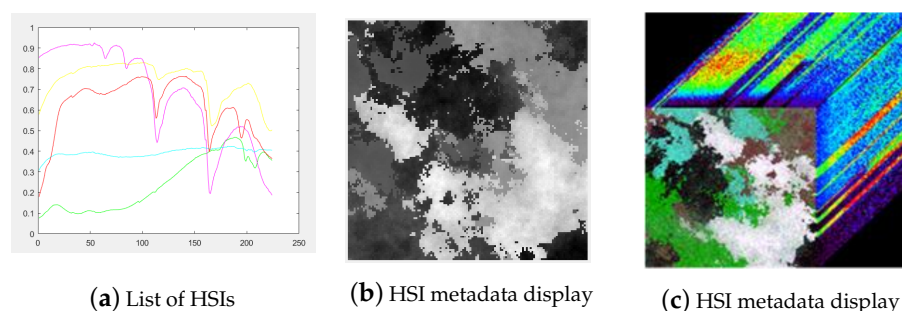
**Figure 8.** Parallel and distributed implements of sum-to-one constrained least squares (SCLS).

## 4. Experiments

### 4.1. Experiment Setup

We use a synthesized HSI dataset to verify our proposed parallel CBIR system. The images in the dataset are generated in a fully controlled environment for our evaluation of retrieval accuracy so that the accuracy and efficiency of the parallel algorithms can be effectively tested and validated. In this work, we randomly select five spectra from the digital spectrum library the United States Geological Survey (USGS) [34], including Clinocllore\_Fe, Eugsterite GDS140 Syn, Heulandite GDS3, Labradorite HS105.3B, and Montmorillonite+Illite CM42, as shown in Figure 9a. Each spectral contains a total of 224 bands range of 0.4 to 2.5  $\mu\text{m}$ . Then, the Gaussian field method [35] is used to generate the simulated abundance diagram of 5 minerals with the size of  $128 \times 128$ , as shown in Figure 9b. Finally, the simulated Dataset\_S is generated in a linear hybrid model, as shown in Figure 9c.

The other hyperspectral image scene used in our experiments in this work was a subset of the well-known Airborne Visible Infrared Imaging Spectrometer (AVIRIS) Cuprite image [36], available online in reflectance units from <http://aviris.jpl.nasa.gov/data/freedata.html>. This image was collected over the Cuprite mining site, Nevada, in 1995, which has been widely used to validate the performance of endmember extraction algorithms. It consists of 224 spectral channels range of 0.4 to 2.5  $\mu\text{m}$ . A part of bands including 1–3, 107–114, 153–169, and 221–224 were removed prior for analysis due to water absorption and low signal-to-noise ratio. The final Dataset\_C 1 comprises  $350 \times 350$  pixels, 192 bands, and a total size of about 44.86MB. In order to evaluate the performance on big datasets of different sizes, five larger datasets have been generated using the Mosaicking function of ENVI software denoted as follows: Dataset\_C 2:  $11,200 \times 350$  with the size of about 1.40 GB, Dataset\_C 3:  $22,400 \times 350$  with the size of about 2.80 GB, Dataset\_C 4:  $44,800 \times 350$  with the size of about 5.60 GB, Dataset\_C 5:  $89,600 \times 350$  with the size of about 11.20 GB, and Dataset\_C 6:  $179,200 \times 350$  with the size of about 22.40 GB.



**Figure 9.** A synthetic HSI including its spectral signatures and fractal image.

To verify the accuracy and efficiency of the unmixing-based content retrieval method for hyperspectral imagery repository on cloud computing platform based on an OpenStack cluster which contains a control node equipped with an eight-core i7-9700 k CPU at 3.6 GHz with 16 GB memory and a 500 GB solid state Disk (SSD), a network node equipped with the same configuration as the control node, and five computing nodes of each equipped with a sixteen-core i9-9960X CPU at 3.1 GHz with 128 GB memory and 500 GB SSD. After virtualization, a Hadoop and Spark cluster is built with a master and eight slaves. The master node is allocated with 16 cores (logic processors) and each Slave node is allocated with 64 cores and 128 GB memory. All nodes have Java 1.8.0\_201, Apache Spark 2.3.3, Hadoop 2.7.3, Tomcat 9.0, and Centos 7.0 as operating system installed. The distributed parallel version is implemented by Java and Scala hybrid programming. The web services version is realized by Spring, SpringMVC, MyBatis (SSM), MySQL, and Vue. All software and their versions used in experiments are listed in Table 1.

**Table 1.** Software description of experimental environments.

Software	Release Version
Spark	2.3.3
Hadoop	2.7.3
OpenStack	Queens
MySQL	5.7.1
Java	1.8.0_201
Scala	2.11.8
Tomcat	9.0
Spring	4.3.1.RELEASE
MyBatis	1.3.1

#### 4.2. Evaluation of Parallel Unmixing Accuracy

At first, we evaluate the accuracy of our parallel and distributed implementation of the unmixing (PPI and SCLS as a case) on the original Dataset\_S and Dataset\_C 1-6. The serial, pseudo-distributed and distributed unmixing process are performed in the same execution environment. In order to quantitatively evaluate the accuracy of image unmixing, we use the SAD scores in endmember comparison to measure the spectral similarity and root-mean-square error (RMSE) in the estimated abundance fractions to measure the abundance similarity. The SAD scores are calculated as Formula (1) and the RMSE scores is calculated as follows:

$$RMSE = \sqrt{\frac{1}{p \times L} \|X - \hat{a}M\|_F^2}, \quad (7)$$

where  $L$  and  $p$  stand for the number of bands and pixels of image  $X$  of size  $p \times L$ , respectively.  $\hat{a}$  represents the abundance coefficient matrix of size  $p \times n$ , and  $M$  is an endmember matrix of size  $n \times L$ .

The serial, pseudo-distributed and distributed unmixing process give the same SAD and RMSE scores of endmember extraction and abundance inversion, which demonstrates the accuracy of the distributed parallel version of unmixing algorithm. In addition, we can conclude that our system can effectively retrieve the images consisting of endmembers which are closely similar.

#### 4.3. Computational Performance Evaluation

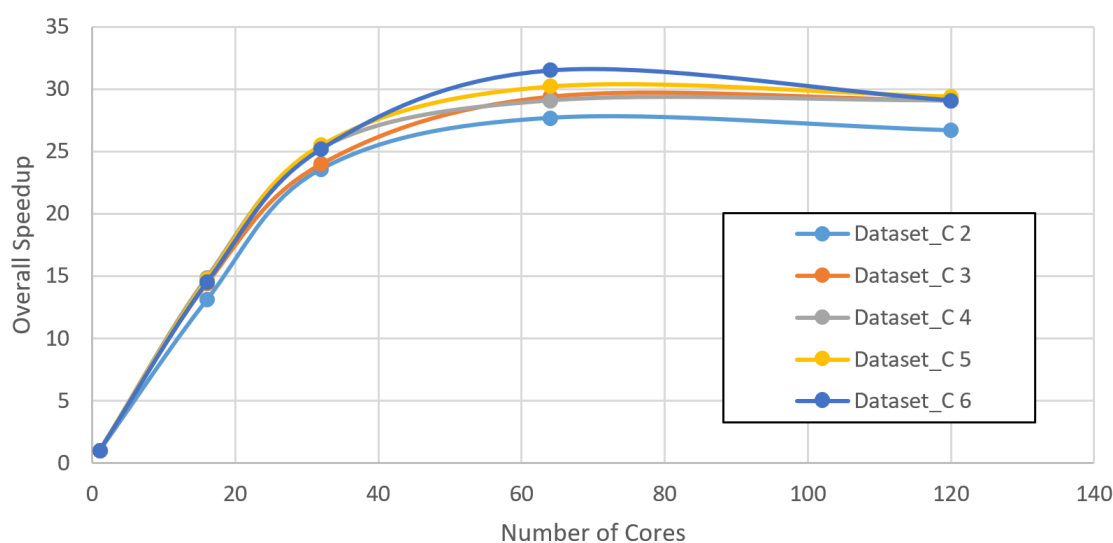
In this section, we performed the proposed distributed parallel unmixing processing on Dataset\_C 2-6 described in Section 4.1 to further evaluate the computational efficiency. In order to describe the acceleration result of unmixing on Spark in detail, the individual parallel PPI and SCLS on Spark were separated from the unmixing chain and were performed, respectively. The execution time and speedup comparison between the serial and parallel versions are given in Tables 2 and 3, respectively, in which cores denote the sum of the executors' cores. It is worth mentioning that we upload all datasets into HDFS with same *blocksize* 11,760,000 bytes (integral multiple of bands); thus, every pixel vector is stored in the same node. Therefore, the parallel PPI and SCLS on Spark can read the original data and form an RDD of which every partition corresponds to a block.

We further analyze the processing time and speedup of the serial and parallel versions. Benefiting from virtualization technology, we can apply for enough memory and execute the serial version to process big data. When counting the execute time, the initialization time of SparkContext and data reading is also included since these times are generally higher in a Spark cluster than in single machine, which is due to the delay in the communication between the nodes in the Spark cluster and between the storage module and computing module.

**Table 2.** Execution time and speedup of the serial and distributed parallel versions of PPI with dataset\_C 2–6.

Version	Cores	Dataset_C 2 (1.40 GB)		Dataset_C 3 (2.80 GB)		Dataset_C 4 (5.60 GB)		Dataset_C 5 (11.20 GB)		Dataset_C 6 (22.40 GB)	
		Time (min)	Speedup (x)	Time (min)	Speedup (x)	Time (min)	Speedup (x)	Time (min)	Speedup (x)	Time (min)	Speedup (x)
Serial version	1	144	-	288	-	582	-	1146	-	2268	-
Distributed version	16	11	13.1	20	14.4	39	14.9	78	14.7	156	14.5
	32	6.1	23.6	12	24.0	23	25.3	45	25.5	90	25.2
	64	5.2	27.7	9.8	29.4	20	29.1	38	30.2	72	31.5
	120	5.4	26.7	9.9	29.1	20	29.1	39	29.4	78	29.1

Referring to the execution time and speedup statistics of parallel PPI reported in Table 2 and Figure 10, we observe that when the number of cores is less than a certain value, the speedup is nearly linear regarding to the core count. However, if we continue to increase the number of cores, the growth of speedup becomes slower or even negative. This observation is due to the fact that as the increasing number of cores, the communication cost between the nodes increases and the cost growth rate is superlinear, which leads to no longer negligible communication cost. For example, when the data is Dataset\_C 2 (1.4 GB), employing 16 cores leads to a  $13.1\times$  speedup and the speedup has been increased by 80.15% through doubling the number of cores (32), but, when the number of cores reaches 64, the speedup is only 17.37% higher than that by using 32 cores.

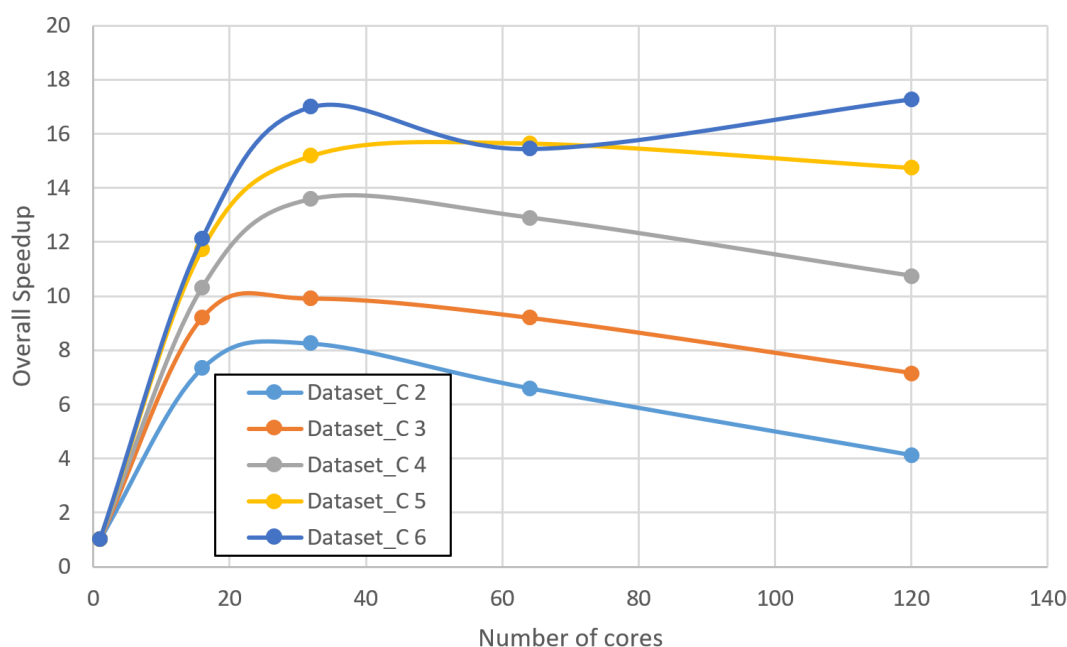
**Figure 10.** Speedup achieved by the parallel PPI on Spark for various data sizes with different numbers of cores.

Referring to the execution time and speedup statistics of parallel SCLS reported in Table 3 and Figure 11, the similar speedup trends with regard to the number of cores for all datasets can be observed. The only difference is that when the number of cores is fixed, the speedup of parallel SCLS increases with the data size, while the speedup of parallel PPI keeps at the same. As shown in Table 2 and 3, the row in which the cores number is fixed at 32 lists the spent time and speedups achieved for different datasets sizes. The parallel PPI's speedups are near  $25\times$ , while the parallel SCLS's are  $8.25\times$ ,  $9.92\times$ ,  $13.58\times$ ,  $15.18\times$ , and  $17.00\times$ . Since the amount of calculation of parallel SCLS is far less than that of parallel PPI. The cost of parallel PPI is dominated by  $O(YLN)$ , while the cost of parallel SCLS is dominated by  $O(n(L+1)N)$  where  $n \ll Y$ . The map operation to compute the maximal and minimal projects accounts for a majority of the overall execution time, the parallel PPI's speedups are relatively constant when the cores number is fixed. In addition, the improvement of parallel SCLS is because the system and communication

overhead depends more on the number of computing nodes than on data size [37,38]. With the increasing in data size, the proportion of the actual computation time goes up since its ratio to the communication and system overhead scales up. The results also explain why the parallel PPI's computational efficiency is better than the parallel SLCS's.

**Table 3.** Execution time and speedup of the serial and distributed parallel versions of SCLS with dataset\_C 2–6.

Version	Cores	Dataset_C 2 (1.40 GB)		Dataset_C 3 (2.80 GB)		Dataset_C 4 (5.60 GB)		Dataset_C 5 (11.20 GB)		Dataset_C 6 (22.40 GB)	
		Time (s)	Speedup (x)	Time (s)	Speedup (x)	Time (s)	Speedup (x)	Time (s)	Speedup (x)	Time (s)	Speedup (x)
Serial version	1	144	-	288	-	582	-	1146	-	2268	-
Distributed version	16	11	13.1	20	14.4	39	14.9	78	14.7	156	14.5
	32	8	8.25	13	9.92	19	13.58	34	15.18	60	17.00
	64	10	6.60	14	9.21	20	12.9	33	15.64	66	15.45
	120	16	4.13	18	7.17	24	10.75	35	14.74	59	17.29



**Figure 11.** Speedup achieved by the parallel SCLS on Spark for various data sizes with different numbers of cores.

## 5. Discussion

### 5.1. Method Advantage Analysis

The experimental results presented in Section 4 justify the accuracy and computational efficiency of the parallel unmixing-based CBIR system. This depends largely, we believe, on the cloud-computing-based CBIR architecture in SaaS mode. The storage, management, and retrieval services for HSIs are provided for free through a web browser, which has brought great convenience of maintenance and use. In its concrete implementation, we first propose a standard HSI repository on cloud computing platforms. In the repository, we intend to keep the standard format for HSI data in different formats from all over the world. In this way, we can distribute those hyperspectral data among the potential users in time and even improve users' work efficiency because of the reduced overhead of format conversion. In addition, we design a unified hyperspectral retrieval process based unmixing including two retrieval process. The first way is retrieving images with a image, and the second way is retrieving images with spectrums directly. A spectral library is built which covers every substance in ideal situation firstly. The spectral/spatial information is



extracted by parallel unmixing method and then stored in MySQL database. The spectrums of the data are indexed by the spectrums of the library. Therefore, when users retrieve images, especially in the second way, band-level comparisons, e.g., SAD can be omitted and results can be obtained quickly compared to traditional methods. As a result, the HSI storage, metadata storage, data management, and retrieval form a closed loop.

Moreover, we make use of the powerful storage and computation power of cloud computing to store various HSIs and accelerate the unmixing processing flow. More HSI data can be stored and processed than traditional platforms, such as single machine and GPUs clusters, due to the scalability of cloud computing. To speed up the processing speed and shorten the period of endmember extraction, we implement the distributed parallel unmixing including parallel PPI and SCLS algorithms based on Spark. As shown in Figures 10 and 11, we obtain a good acceleration ratio both for synthetic data and real hyperspectral images in different size. In addition, the unmixing algorithms are extensible if we code in uniform standards. As a result, unmixing algorithms are selectable for higher computing speed and accuracy.

### 5.2. Limitation Analysis

In fact, the parallel unmixing-based content retrieval system for distributed hyperspectral imagery repository on cloud computing platforms offer our users with the convenience of use, but substantial computer resources are needed including computing power, storage space, network traffic, etc. Besides, in the real world, there are too many uncertainties that lead to poor HSI quality resulting high requirements for unmixing algorithms. If the spectral information extracted is inaccurate, the accuracy of the retrieval is further affected.

Furthermore, the acceleration ratio cannot increase linearly as the number of cores increases in the process of unmixing acceleration. It is caused by the built-in task scheduling and data migration of Spark. Therefore, the parallel strategy for unmixing algorithms should be well designed.

## 6. Conclusions

In this paper, we propose a novel parallel CBIR system for hyperspectral image (HSI) repository on cloud computing platforms under the guide of unmixed spectral information. To accelerate spectral information extraction, we implement a distributed and parallel unmixing method that operates on cloud computing platforms for accelerating the unmixing processing flow. In addition, we implement a global standard distributed HSI repository equipped with a large spectral library in a software-as-a-service mode, providing users with HSI storage, management, and retrieval services through web interfaces. As a case study, we deploy a Spark-based parallel unmixing flow including PPI, SAD, and SCLS to verify the accuracy and efficiency of the proposed architectures. Experimental results demonstrate that the proposed parallel and distributed implementation is effective in terms of both unmixing accuracy and computational efficiency. In future work, we plan to implement more unmixing algorithms for meta-data extraction and provide more retrieval methods for more types of remote sensing images.

**Author Contributions:** Conceptualization, P.Z., Z.W. (Zebin Wu), and A.P.; methodology, Z.W. (Zebin Wu); software, Y.S.; validation, P.Z. and Y.S.; formal analysis, P.Z. and J.Y.; investigation, P.Z. and J.S.; resources, Y.Z. (Yaoqin Zhu); data curation, P.Z., Y.S., and Z.W. (Zebin Wu); writing—original draft preparation, P.Z.; writing—review and editing, Y.Z. (Yi Zhang) and J.S.; visualization, P.Z. and Y.S.; supervision, A.P. and Z.W. (Zhibin Wei); project administration, Z.W. (Zebin Wu); funding acquisition, Z.W. (Zebin Wu). All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded in part by the National Natural Science Foundation of China under Grant 61772274, Grant 61976117, Grant 61701238, Grant 61501241, Grant 61671243, and Grant 61802190, in part by the Jiangsu Provincial Natural Science Foundation of China under Grant BK20180018, Grant BK20170858, and Grant BK20191409, in part by the Fundamental Research Funds for the Central Universities under Grant 30917015104, Grant 30919011103, and Grant 30919011402, Grant 30919011234.

**Institutional Review Board Statement:** The study did not involve humans or animals.

**Informed Consent Statement:** The study did not involve humans.

**Data Availability Statement:** The datasets in this study are available online from <http://aviris.jpl.nasa.gov/data/freedata.html> and <https://github.com/ZpWaitingForSunshine/CBIR/tree/master/data>.

**Acknowledgments:** We are grateful to United States Geological Survey (USGS), who provided the spectral library for this research. We would also like to thank NASA, who provided Cuprite data for the experimental validation.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Bioucas-Dias, J.M.; Plaza, A.; Camps-Valls, G.; Scheunders, P.; Nasrabadi, N.; Chanussot, J. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geosci. Remote Sens. Mag.* **2013**, *1*, 6–36. [\[CrossRef\]](#)
2. Kruse, F.A.; Boardman, J.W.; Huntington, J.F. Comparison of airborne hyperspectral data and EO-1 Hyperion for mineral mapping. *IEEE Trans. Geosci. Remote Sens.* **2003**, *41*, 1388–1400. [\[CrossRef\]](#)
3. Liu, X.; Gao, Z.; Ning, J.; Yu, X.; Zhang, Y. An improved method for mapping tidal flats based on remote sensing waterlines: A case study in the Bohai Rim, China. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 5123–5129. [\[CrossRef\]](#)
4. Zhao, J.; Xu, T.; Xiao, J.; Liu, S.; Mao, K.; Song, L.; Yao, Y.; He, X.; Feng, H. Responses of water use efficiency to drought in southwest China. *Remote Sens.* **2020**, *12*, 199. [\[CrossRef\]](#)
5. Cao, L.; Chen, X.; Zhang, C.; Kurban, A.; Qian, J.; Pan, T.; Yin, Z.; Qin, X.; Ochege, F.U.; Maeyer, P.D. The global spatiotemporal distribution of the mid-tropospheric CO<sub>2</sub> concentration and analysis of the controlling factors. *Remote Sens.* **2019**, *11*, 94. [\[CrossRef\]](#)
6. Kupilik, M.; Witmer, F.D.; MacLeod, E.; Wang, C.; Ravens, T. Gaussian process regression for arctic coastal erosion forecasting. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 1256–1264. [\[CrossRef\]](#)
7. Cao, R.; Feng, Y.; Liu, X.; Shen, M.; Zhou, J. Uncertainty of vegetation green-up date estimated from vegetation indices due to snowmelt at northern middle and high latitudes. *Remote Sens.* **2020**, *12*, 190. [\[CrossRef\]](#)
8. Zhang, Z.; Shao, Y.; Tian, W.; Wei, Q.; Zhang, Y.; Zhang, Q. Application potential of GF-4 images for dynamic ship monitoring. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 911–915. [\[CrossRef\]](#)
9. Arias, L.; Cifuentes, J.; Marín, M.; Castillo, F.; Garcés, H. Hyperspectral imaging retrieval using MODIS satellite sensors applied to volcanic ash clouds monitoring. *Remote Sens.* **2019**, *11*, 1393. [\[CrossRef\]](#)
10. Yu, W.; Ma, M. Scale mismatch between in situ and remote sensing observations of land surface temperature: Implications for the validation of remote sensing LST products. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 497–501.
11. Ke, C. Military object detection using multiple information extracted from hyperspectral imagery. In Proceedings of the IEEE International Conference on Progress in Informatics and Computing, Nanjing, China, 27–29 October 2017; pp.124–128.
12. Rathore, M.M.U.; Paul, A.; Ahmad, A.; Chen, B.; Huang, B.; Ji, W. Real-time big data analytical architecture for remote sensing application. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4610–4621. [\[CrossRef\]](#)
13. Arrow, K.; Solow, R.; Portney, P.R.; Leamer, E.E.; Radner, R.; Schuman, H. Report of the NOAA panel on contingent valuation. *Fed. Regist.* **1993**, *58*, 4601–4614.
14. Justice, C.; Giglio, L.; Korontzi, S.; Owens, J.; Morisette, J.T.; Roy, D.; Descloitres, J.; Alleaume, S.; Petitcolin, F.; Kaufman, Y. The MODIS fire products. *Remote Sens. Environ.* **2002**, *83*, 244–262. [\[CrossRef\]](#)
15. Sevilla, J.; Plaza, A. A new digital repository for hyperspectral imagery with unmixing-based retrieval functionality implemented on GPUs. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2267–2280. [\[CrossRef\]](#)
16. Yu, H.; Gao, L.; Liao, W.; Zhang, B.; Aleksandra, P.; Wilfried, P. Multiscale superpixel-level subspace-based support vector machines for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 2142–2146. [\[CrossRef\]](#)
17. Luo, W.; Gao, L.; Zhang, R.; Andrea, M.; Zhang, B. Bilinear normal mixing model for spectral unmixing. *IET Image Processing* **2019**, *13*, 344–354. [\[CrossRef\]](#)
18. Wu, Z.; Shi, L.; Li, J.; Wang, Q.; Sun, L.; Wei, Z.; Plaza, J.; Plaza, A. GPU parallel implementation of spatially adaptive hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *4*, 1–13. [\[CrossRef\]](#)
19. Zheng, X.; Xue, Y.; Guang, J.; Liu, J. Remote sensing data processing acceleration based on multi-core processors. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Beijing, China, 10–15 July 2017; pp. 641–644.
20. Rabe, B.R.; Clifford, M.; Miles, N. Storage Area Network (SAN) Management System for Discovering SAN Components Using a SAN Management Server. U.S. Patent 7194538B1, 20 March 2007.
21. Chen, P.M.; Lee, E.K.; Gibson, G.A.; Katz, R.H.; Patterson, D.A. RAID: High-performance, reliable secondary storage. *ACM Comput. Surv.* **1994**, *26*, 145–185. [\[CrossRef\]](#)
22. Gibson, G.A.; Van, M.R. Network attached storage architecture. *Commun. ACM* **2000**, *43*, 37–45. [\[CrossRef\]](#)

23. Zheng, P.; Wu, Z.; Zhang, W.; Li, M.; Yang, J.; Zhang, Y.; Wei, Z. An unmixing-based content retrieval method for hyperspectral imagery repository on cloud computing platform. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 23–27 July 2018; pp. 3583–3586.
24. Zinno, I.; Elefante, S.; Mossucca, L.; De, L.C.; Manunta, M.; Terzo, O.; Lanari, R.; Casu, F. A first assessment of the P-SBAS DInSAR algorithm performances within a cloud computing environment. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4675–4686. [[CrossRef](#)]
25. Wu, Z.; Li, Y.; Plaza, A.; Li, J.; Xiao, F.; Wei, Z. Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 2270–2278. [[CrossRef](#)]
26. Cafaro, M.; Giovanni, A. Grids, clouds, and virtualization. In *Grids, Clouds and Virtualization*; Springer: London, UK, 2011; pp. 1–21.
27. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [[CrossRef](#)]
28. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The hadoop distributed file system. In Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies, Lake Tahoe, NV, USA, 3–7 May 2010; pp. 1–10.
29. Kruse, F.A.; Lefkoff, A.B.; Boardman, J.W.; Heidebrecht, K.B.; Shapiro, A.T.; Barloon, P.J.; Goetz, A. The Spectral Image Processing System (SIPS): Software for integrated analysis of AVIRIS data. In Summaries of the Third Annual JPL Airborne Geoscience Workshop, Pasadena, CA, USA, 1–5 June 1992.
30. Molinier, M.; Laaksonen, J.; Hame, T. Detecting man-made structures and changes in satellite imagery with a content-based information retrieval system built on self-organizing maps. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 861–874. [[CrossRef](#)]
31. Veganzones, M.A.; Grana, M. A spectral/spatial CBIR system for hyperspectral images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 488–500. [[CrossRef](#)]
32. Chang, C.I.; Plaza, A. A fast iterative algorithm for implementation of pixel purity index. *IEEE Geosci. Remote Sens. Lett.* **2006**, *3*, 63–67. [[CrossRef](#)]
33. Chaudhry, F.; Wu, C.; Liu, W.; Chang, C.; Plaza, A. Pixel purity index-based algorithms for endmember extraction from hyperspectral imagery. *Recent Adv. Hyperspectral Signal Image Process.* **2006**, *37*, 29–62.
34. Wild, E.C. USGS library training and outreach: Finding and using scientific literature and data. In *Geoscience Information Services: “Peak” Performances*; Geoscience Information Society: Alexandria, VA, USA, 2010; Volume 51.
35. Bertschinger, E. Multiscale Gaussian random fields and their application to cosmological simulations. *Astrophys. J. Suppl. Ser.* **2001**, *137*, 1. [[CrossRef](#)]
36. Vane, G.; Green, R.O.; Chrien, T.G.; Enmark, H.T.; Hansen, E.G.; Porter, W.M. The airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sens. Environ.* **1993**, *44*, 127–143. [[CrossRef](#)]
37. Sun, J.; Zhang, Y.; Wu, Z.; Zhu, Y.; Yin, X.; Ding, Z.; Wei, Z.; Plaza, J.; Plaza, A. An efficient and scalable framework for processing remotely sensed big data in cloud computing environments. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 4294–4308. [[CrossRef](#)]
38. Wu, Z.; Sun, J.; Zhang, Y.; Zhu, Y.; Li, J.; Plaza, A.; Benediktsson, J.A.; Wei, Z. Scheduling-guided automatic processing of massive hyperspectral image classification on cloud computing. *IEEE Trans. Cybern.* **2020**, doi: 10.1109/TCYB.2020.3026673 [[CrossRef](#)]