Journal of
Applied Remote Sensing

Parallel hyperspectral image processing on distributed multicluster systems

Fangbin Liu
Frank J. Seinstra
Antonio Plaza

SPIE

Parallel hyperspectral image processing on distributed multicluster systems

Fangbin Liu,[a] Frank J. Seinstra,[b] and Antonio Plaza[c]
[a] Universiteit van Amsterdam, Informatics Institute,
Science Park 107, 1098 XG Amsterdam, The Netherlands
[b] Vrije Universiteit, Department of Computer Science,
De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands
[c] University of Extemadura, Hyperspectral Computing Laboratory,
Department of Technology of Computers and Communications,
Avenida de la Universidad s/n, E-10003 Caceres, Spain
aplaza@unex.es

Abstract. Computationally efficient processing of hyperspectral image cubes can be greatly beneficial in many application domains, including environmental modeling, risk/hazard prevention and response, and defense/security. As individual cluster computers often cannot satisfy the computational demands of emerging problems in hyperspectral imaging, there is a growing need for distributed supercomputing using multicluster systems. A well-known manner of obtaining speedups in hyperspectral imaging is to apply data parallel approaches, in which commonly used data structures (e.g., the image cubes) are being scattered among the available compute nodes. Such approaches work well for individual compute clusters, but—due to the inherently large wide-area communication overheads—these are generally not applied in distributed multicluster systems. Given the nature of many algorithmic approaches in hyperspectral imaging, however, and due to the increasing availability of high-bandwidth optical networks, wide-area data parallel execution may well be a feasible acceleration approach. This paper discusses the wide-area data parallel execution of two realistic and state-of-the-art algorithms for endmember extraction in hyperspectral unmixing applications: automatic morphological endmember extraction and orthogonal subspace projection. It presents experimental results obtained on a real-world multicluster system, and provides a feasibility analysis of the applied parallelization approaches. The two parallel algorithms evaluated in this work had been developed before for single-cluster execution, and were not changed. Because no further implementation efforts were required, the proposed methodology is easy to apply to already available algorithms, thus reducing complexity and enhancing standardization. [DOI: 10.1117/1.3595292]

Keywords: hyperspectral image processing; distributed multicluster systems; optical networks.

Paper 11017SSR received Feb. 1, 2011; revised manuscript received Apr. 1, 2011; accepted for publication May 3, 2011; published online Nov. 18, 2011.

1 Introduction

Hyperspectral imaging involves the measurement, analysis, and interpretation of spectra acquired from a given scene by an airborne or satellite sensor. A major challenge for the development of efficient hyperspectral imaging systems in the context of remote sensing is how to deal with the extremely large volumes of data produced by the imaging spectrometers.[1] For instance, the NASA Jet Propulsion Laboratory's Airborne Visible Infra-red Imaging Spectrometer (AVIRIS)[2] is able to record the visible and near-infrared spectrum (wavelength region from 0.4 to 2.5 μm) of the reflected light of an area 2 to 12 km wide and several kilometers long using 224

1931-3195/2011/$25.00 © 2011 SPIE

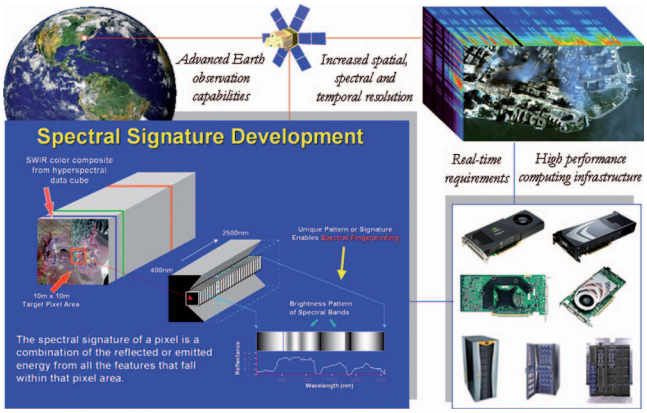

Fig. 1 An illustration of the processing demands introduced by the ever increasing dimensionality of remotely sensed hyperspectral imaging instruments.

spectral bands. The resulting hyperspectral data cube is a stack of images in which each pixel (vector) is represented by a spectral signature that uniquely characterizes the underlying scene (see Fig. 1). With the resulting multidimensional data volume typically comprising many Gbytes per flight, and individual compute clusters either being too small or largely being occupied by other users, there is a growing need for distributed supercomputing using (large) collections of compute clusters.

It is well-known that data parallel approaches in the field of hyperspectral image processing can provide efficient acceleration.[3] In such approaches, the input data cube is scattered uniformly among the available compute nodes, such that each node processes the allocated partial data structure only. Internode data dependencies are then resolved by communication between the nodes. In recent years, numerous (hyperspectral) imaging applications have been executed successfully in this manner, in particular using compute clusters, which are now widely available for cost-effecive exploitation.[3,4]

For fine-grain data parallel execution internode communication is the major hurdle for obtaining high speedups. With the advent of low-latency, high-bandwidth optical networks, however, wide-area data parallel execution may become a feasible acceleration approach. In this paper we apply wide-area data parallelism to a specific class of fine-grained, tightly-coupled, data parallel regular domain problems, represented by two state-of-the-art algorithms for spectral unmixing of hyperspectral data. In recent years, many hyperspectral imaging algorithms have been researched extensively for their high computational demands.[3,5,6] One of the most relevant techniques for information extraction from hyperspectral data is spectral unmixing, which aims to overcome problems that occur when the spatial resolution of a remote sensor is not high enough to separate different materials.[6] Figure 2(a) illustrates the problem of mixed pixels containing different macroscopically pure spectral substances (endmembers) that need to be found by a certain algorithm, in order to interpret each mixed pixel as a combination of endmembers weighted by their respective abundance fractions. A second important problem is that of finding spectrally distinct signatures in the scenes. Depending on the complexity and dimensionality of the input scene, these problems may be computationally very expensive, limiting the possibility of utilizing the algorithms in— for example— time-critical applications.

Two representative algorithms to address the problem of finding spectrally pure signatures (endmembers) for unmixing hyperspectral images are the automated morphological endmember extraction [AMEE Ref. 3 and 7], and the orthogonal subspace projection [OSP Ref. 5 and 8]. Although several other approaches are described in the literature (Ref. 9)—most notably the spectral-based N-FINDR algorithm for endmember extraction [Ref. 10]– previous parallel versions of N-FINDR for single cluster systems show that this algorithm is hard to parallelize efficiently.[11] In turn, efficient implementations for this algorithm have been developed based

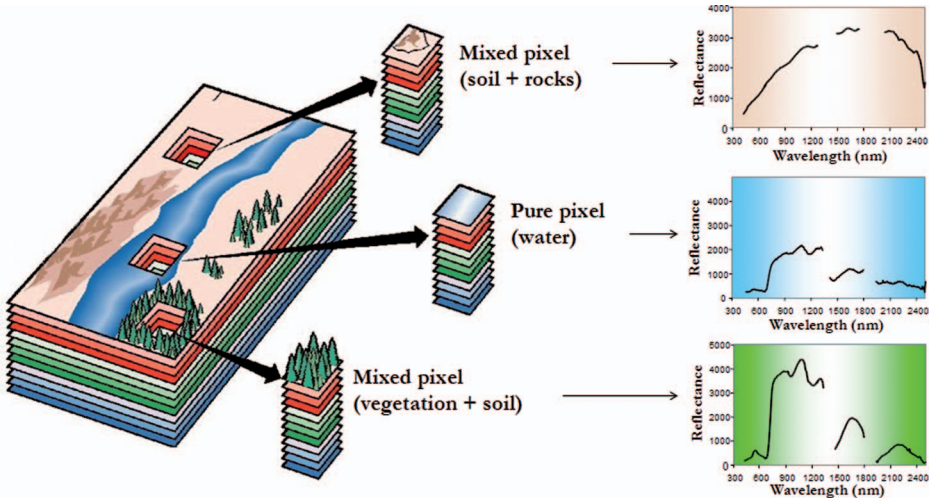

Fig. 2 The mixture problem in hyperspectral data analysis.

on software optimizations.[12] This supports our selection of AMEE (representative of a class of spatial-spectral algorithms) and OSP (a spectral-based algorithm, much like N-FINDR, but with more intensive communications than the AMEE).

The two selected algorithms represent two widely used design choices in the endmember extraction community: i. use of spectral information alone (OSP), and ii. use of spatial and spectral information in combined fashion (AMEE). Both methods are dominated by regular computations which make them appealing for parallel processing in commodity clusters. Also, these methods are well-established in the literature and have been used by many authors in different application contexts. For instance, different applications of OSP have been explored in Refs. 1, 5, 6, 8, and 13 while other applications of AMEE have been explored in Refs. 3, 7, 9, 11, and 14. The algorithms are applied to real AVIRIS hyperspectral data sets on a distributed multicluster system which—apart from being equipped with conventional Internet connections—comprises a dedicated wide-area optical interconnect. We present experimental results, and provide a feasibility analysis of the applied parallelization approach leading to findings which can be generalized to other fine-grained, tightly-coupled, data parallel regular domain problems.

This paper is organized as follows. Section 2 describes the two considered hyperspectral imaging algorithms. Section 3 describes the data parallel execution of the two algorithms. Section 4 describes the distributed multicluster system applied in the experiments. Section 5 gives an evaluation of the performance and speedup results. Section 6 concludes with some remarks and hints at plausible future research.

2 Hyperspectral Imaging Algorithms

In recent years, many hyperspectral imaging algorithms have been researched extensively for their high computational demands.[3,5,6] The following describes two important, and computationally expensive, techniques for information extraction from hyperspectral data by way of spectral unmixing: AMEE and OSP.

2.1 AMEE

The AMEE (Ref. 7) algorithm runs on the full data cube with no dimensional reduction, and begins by searching spatial neighborhoods around each pixel vector in the image for the

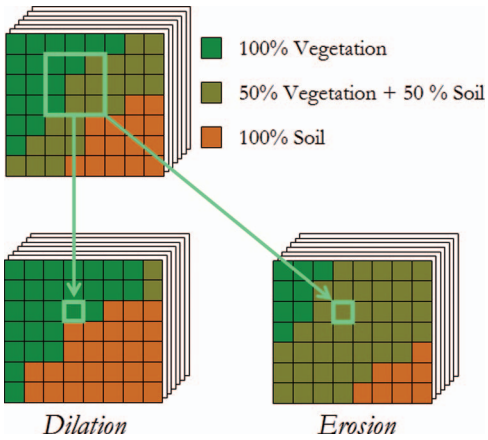

Fig. 3 Example illustrating extended morphological operations of dilation and erosion.

most spectrally pure and most highly mixed pixel. This task is performed by using extended mathematical morphology operators of dilation and erosion,[7] which are graphically illustrated in Fig. 3. Here, dilation selects the most spectrally pure pixel in a local neighborhood around each pixel vector, while erosion selects the most highly mixed pixel in the same neighborhood. Each spectrally pure pixel is then assigned an eccentricity value, which is calculated as the spectral angle distance between the most spectrally pure and most highly mixed pixel for each given spatial neighborhood. This process is repeated iteratively for larger spatial neighborhoods up to a maximum size that is pre-determined. At each iteration, the eccentricity values of the selected pixels are updated. The final end member set is obtained by selecting the set of pixel vectors with higher associated eccentricity scores and removing those pixels which are not sufficiently distinct among each other in a spectral sense.

The inputs to AMEE are the full hyperspectral data cube $\mathbf{F}$, a structuring element B (used to define the spatial context around each image pixel), a maximum number of algorithm iterations $I_{\max}$, and a number of endmembers (pure spectral signatures) to be extracted, p. The output is an endmember set, $\{\boldsymbol{E}_i\}_{i=1}^{q}$, with $q \leq p$. The algorithm consists of the following steps:[7,14]

1. Set $i = 1$ and initialize a morphological eccentricity index (MEI) $(x, y) = 0$ for each hyperspectral image pixel $\mathbf{F}(x, y)$.
2. Move B through all the pixels of the input data, defining a local spatial search area around each pixel $\mathbf{F}(x, y)$, and calculate the maximum and minimum pixel vectors at each B neighborhood using extended morphological erosion and dilation, respectively, defined as follows:

$$(\mathbf{F} \Theta B)(x, y) = \underset{(i,j) \in Z^2(B)}{\operatorname{argmin}} \{\mathrm{DB}[\mathbf{F}(x+i, y+j)]\}, \tag{1}$$

$$(\mathbf{F} \oplus B)(x, y) = \underset{(i,j) \in Z^2(B)}{\operatorname{argmin}} \{\mathrm{DB}[\mathbf{F}(x+i, y+j)]\}. \tag{2}$$

3. Update the MEI at each spatial location (x, y) using:

$$\mathrm{MEI}(x, y) = \mathrm{Dist}[(\mathbf{F} \Theta B)(x, y), (\mathbf{F} \oplus B)(x, y)] \tag{3}$$

4. Set $i = i + 1$. If $i = I_{\max}$, then go to step 5. Otherwise, replace the original image with its dilation using B using $\mathbf{F} = \mathbf{F} \oplus B$. This represents an optimization of the algorithm that propagates only the purest pixels at the local neighborhood to the following algorithm iteration. Then, go to step 2.

5. Select the set of p pixel vectors with higher associated MEI scores (called endmember candidates) and form a unique spectral set of $\{\mathbf{E}_i\}_{i=1}^{q}$ pixels, with $q \leq p$, by calculating the spectral angle[1] for all pixel vector pairs.
6. Finally, a fully constrained unmixing procedure[15] is accomplished by applying a least-squares-based technique to estimate the abundance fractions of each of the p derived endmembers in each pixel of the hyperspectral image $\mathbf{F}$.

2.2 OSP

The OSP was developed to find spectrally distinct pixel vectors in the scene.[13] The algorithm first calculates the pixel vector with maximum length in the hyperspectral image and labels it as an initial endmember $\mathbf{E}_1$ [see Fig. 4(a)] using the following expression:

$$\mathbf{E}_1 = \underset{(x,y)\in Z^2(\mathbf{F})}{\operatorname{argmax}} \{\boldsymbol{F}_{(x,y)} \cdot \mathbf{F}(x,y)^T\}, \tag{4}$$

where the superscript T denotes the vector transpose operation. Once an initial endmember has been identified, the algorithm assigns $\mathbf{U}_1 = [\mathbf{E}_1]$ and applies an orthogonal projector[13] to all image pixel vectors, thus calculating the second endmember as the one with the maximum projection value [see Fig. 4(b)] as follows:

$$\begin{aligned} \mathbf{E}_2 = & \underset{(x,y)\in Z^2(\mathbf{F})}{\operatorname{argmax}} \left\{ \left[P_{U_1}^{\perp} F(x,y) \right]^T \left[P_{U_1}^{\perp} F(x,y) \right] \right\} \\ & \text{with } p_{U_1}^{\perp} = I - U_1 (U_1^T U_1)^{-1} U_1^T, \end{aligned} \tag{5}$$

where $\mathbf{I}$ is the identity matrix and the matrix $\mathbf{U}_1$ will be iteratively expanded with the inclusion of the additional endmembers identified in the process, i.e., at the next iteration $\mathbf{U}_2 = [\mathbf{E}_1, \mathbf{E}_2]$. As a result, $\mathbf{U}_j$ is simply a matrix that stores the j endmembers which have been already extracted in iterative fashion in the process. With this in mind, the algorithm now iterates to find new

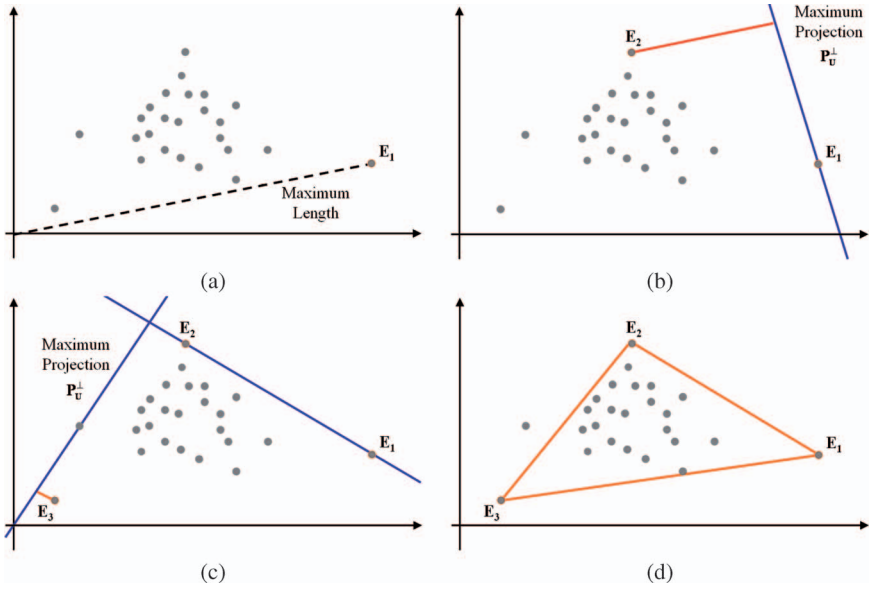

Fig. 4 Graphical interpretation of the OSP algorithm in a two-dimensional space. (a) Initial endmember $\mathbf{E}_1$ selected as the pixel with maximum length. (b) Orthogonal projection to find second endmember $\mathbf{E}_2$. (c) Orthogonal projection to find third endmember $\mathbf{E}_3$. (d) The algorithm terminates for a given number of endmembers (in this case, $p = 3$).

endmember pixels iteratively [see Fig. 4(c)] as the pixels with maximum projection value:

$$\mathbf{E}_{j+1} = \underset{(x,y)\in Z^2(F)}{\operatorname{argmax}} \left\{ \left[P_{U_j}^{\perp} F(x,y) \right]^T \left[P_{U_j}^{\perp} F(x,y) \right] \right\} \quad \text{with} \quad p_{U_j}^{\perp} = I - U_j (U_j^T U_j)^{-1} U_j^T. \tag{6}$$

The iterative procedure is terminated once a predefined number of p endmembers, $\{\mathbf{E}_i\}_{i=1}^{p}$, has been identified [see Fig. 4(d)].

3 Parallel Implementations

The parallel implementations of the two algorithms are based on a data-driven partitioning strategy in which different parts of the image cube are assigned to different compute nodes. Essentially, two approaches for data partitioning exist, namely spectral-domain decomposition and spatial-domain decomposition. Previous experimentation with both strategies have shown that spatial-domain decomposition can significantly reduce inter-processor communication, resulting from the fact that a single pixel vector is never partitioned and communication is not needed at the pixel level.[3] Another important advantage of spatial domain partitioning is that the problem of load-balancing on multicluster systems is not more complex in comparison with the one on a single cluster system. Before describing the parallel algorithms, we emphasize that the total number of iteration counts performed by each node/cluster in the parallel versions summarized below are exactly the same. As a result, the computation loads are equally balanced among nodes and also among clusters (i.e., when the algorithms are run in a multicluster environment).

3.1 Parallel AMEE

1. The master processor partitions the data into K spatial-domain partitions. To avoid repetitive communication steps taking place, we make sure that each node is provided with a partial data structure that contains a so-called "scratch border" with size depending on the structuring element, containing all data that otherwise would need to be communicated during the course of the execution of the algorithm (see Fig. 5).[3] These parallel spatial-spectral partitions are denoted by $\{\mathbf{PSSP}_i\}_{i=1}^{K}$.

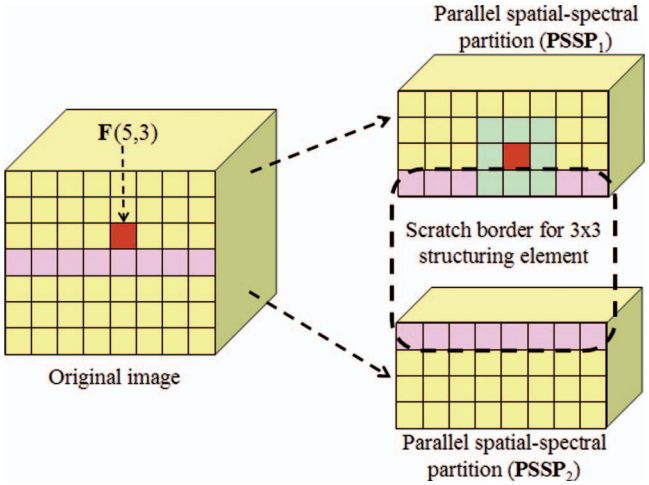

Fig. 5 Scratch border for a 3 × 3-pixel structuring element in the partitioning of a hyperspectral image into two parallel spatial-spectral partitions.

2. Using parameters $I_{\max}$ (maximum number of iterations) and p (maximum number of endmembers to be extracted), each worker executes (in parallel) steps 1–5 of the sequential AMEE algorithm for its corresponding $\mathbf{PSSP}_i$, thus obtaining a MEI score for each pixel in the local partition and obtaining a local set of unique endmembers.
3. The master gathers all the local endmember sets provided by the workers and forms a global set of endmembers $\{\mathbf{E}_i\}_{i=1}^{p}$, with $q \leq p$, by calculating the spectral angle for all possible endmember pairs in parallel.
4. Using the sets $\{\mathbf{PSSP}_i\}_{i=1}^{K}$ and $\{\mathbf{E}_i\}_{i=1}^{p}$, perform step 6. of the sequential AMEE algorithm locally at each worker. This is an embarrassingly parallel computation, since the abundance estimations can be performed independently at each worker. In the end, the master gathers the information provided by the workers, and forms a final output.

While several steps are purely sequential (performed by the master alone), the AMEE algorithm has high potential for parallel speedups and scalability in the number of workers. This is because step 3—which is by far the most time-consuming part of the algorithm—is fully parallel and can be performed without the need for internode communication.

3.2 Parallel OSP

1. As above, the master node first splits the input image cube into K spatial-domain partitions (without scratch borders, as this algorithm is a pixel-based technique), and distributes these among the workers. The communication overhead is based on broadcast communication performed by the master node, and not on peer-to-peer communication. Note that the communication overhead generated by this step will grow linearly with an increasing number of nodes.
2. Each worker calculates the brightest pixel in its local partition using Eq. (4) and then sends the spatial location of the local brightest pixel back to the master. Note that the calculation time in this step will decrease linearly with an increasing number of nodes.
3. The master computes the brightest pixel of the complete input scene, $\mathbf{t}_1$, by applying the operator of step 2 to all pixels at the spatial locations received from the workers. Then, the master sets $\mathbf{U}_1 = [\mathbf{E}_1]$ and broadcasts this matrix to all workers. This communication overhead generated by this step grows linearly with an increasing number of nodes.
4. Each worker finds the pixels in its local partition which are most orthogonal relative to the pixel vectors in $\mathbf{U}_1$. Each worker then sends the spatial locations of these local pixels to the master. Note that the calculation time in this step grows linearly with the increasing number of computing nodes applied.
5. The master now finds a second endmember $\mathbf{E}_2$ by applying an orthogonal subspace projection operator to the pixel vectors at the spatial locations provided by the workers, and selecting the one with the maximum score as indicated in Eq. (5). The master now sets $\mathbf{U}_1 = [\mathbf{E}_1, \mathbf{E}_2]$ and broadcasts this matrix to all workers. This communication step will increase proportionally with an increasing number of nodes used.
6. Repeat iteratively from step 4 by applying Eq. (6) until a set of p endmember pixels, $\{\mathbf{E}_i\}_{i=1}^{p}$, is extracted from the input data.

Similar to the AMEE algorithm, several steps of the algorithm are purely sequential. Moreover, as the OSP requires more communication steps between master and workers to be performed, there seems less potential for significant speedups and scalability than in the AMEE case. Figure 6 presents a graphical summary of the parallel OSP algorithm.

3.3 Distributed MultiCluster Execution

The presented parallel algorithms initially were developed for execution on single compute clusters. Often, the use of individual compute clusters is not sufficient, however. First, this is

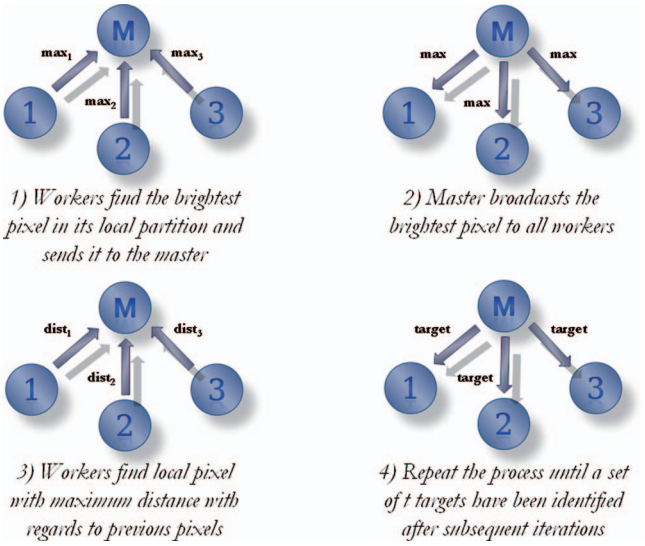

Fig. 6 Parallel OSP using 1 master and 3 workers.

because the shear volume of hyperspectral image data may be too large for a single-cluster system. Similarly, in time-critical applications, a single cluster may not provide the necessary compute power. Finally, even if a single-cluster system in principle would adhere to the algorithmic requirements in terms of available compute power, memory, networking, and storage, such a system often is heavily used for other tasks (by other users)—effectively rendering its use impossible in practice.

To overcome these problems, we apply the presented parallel algorithms unchanged in a multicluster setting. This approach has the advantage that no further implementation effort is required from the application user (in general a domain expert having little or no expertise in high-performance and distributed computing). As communication within a cluster is generally an order of magnitude faster than communication between clusters, results obtained with this approach will indicate a lower bound on the obtainable speedup. Nevertheless, we aim to show that the presented single-cluster algorithms already show close-to-optimal performance in a multicluster setting. Given the nature of the presented parallel algorithms, the need for intercluster communications is expected to be limited, to the effect that the negative impact of such slow wide-area connectivity may be insignificant. In particular, we expect this to be the case in the presence of a (dedicated) low-latency, high-bandwidth optical wide-area network.

If our expectations indeed are supported by actual results obtained in a distributed multicluster setting, this will open up important new possibilities for the field of hyperspectral imaging. First, in particular for time-critical applications it will become possible to scale up to problem sizes that better match the required performance (e.g., in terms of obtained detection results). This will support a trend toward designing algorithms with high potential for efficient parallelization over commonly used, but hard-to-parallelize, ones. Also, for many hyperspectral imaging algorithms (i.e., those which are appealing for parallel implementations) it may become feasible to scale up to much larger processor counts than is currently commonplace, which (a.o.) will allow for much "deeper" analysis of the image data than before.

4 Experimentation Platform: DAS-3

In real-world distributed systems, many fluctuations exist in the performance, load, and availability of the geographically dispersed nodes. As a result, the repeatability of experiments is hard to guarantee. It is therefore essential to perform experiments on a testbed that shows the key

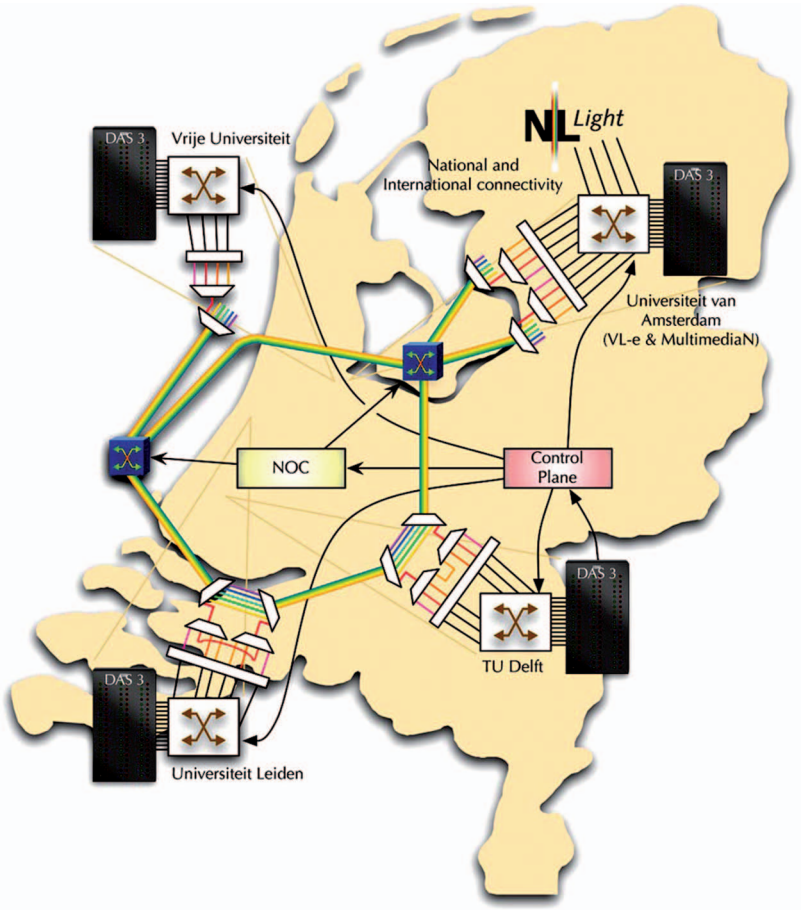

Fig. 7 DAS-3 and the StarPlane Optical Interconnect.

characteristics of a real-world distributed multicluster system on the one hand, yet that allows for controlled experiments on the other.

To meet these requirements, we perform all our evaluations on the DAS-3 testbed system [distributed ASCI Supercomputer 3; see Fig. 7 and also Table 1 (http://www.cs.vu.nl/das3/)]. DAS-3 is a five-cluster wide-area distributed system, with individual clusters located at four different universities in The Netherlands: Vrije Universiteit Amsterdam (VU), University of Amsterdam (UvA), Leiden University (LU), and Delft University of Technology (TUD). The MultimediaN Consortium (UvA-MN) also participates with one cluster.

Besides the ubiquitous 1 and 10 GBit/s Ethernet networks, four of the five DAS-3 clusters incorporate the high-speed Myri-10G interconnect technology from Myricom as a high-speed connection between the nodes within a cluster (as the TUD cluster does not have a Myri-10G interconnect, we do not consider its use in our experiments). Also, apart from being connected by conventional Internet connections, the DAS-3 clusters are interconnected via a 10 GBit/s

Table 1 Specification of four of the five DAS-3 clusters used in our experiments.

Cluster name	No. of nodes (and type)	CPU type	CPU (Ghz)	Memory (GB)	Network
VU	85 dual	Dual-core	2.4	4	Myri-10G
UvA	41 dual	Dual-core	2.2	4	Myri-10G
LU	32 dual	Single-core	2.6	4	Myri-10G
UvA-MN	46 dual	Single-core	2.4	4	Myri-10G

wide-area network based on light paths [StarPlane (http://www.straplane.org)]. The rationale of the StarPlane optical interconnect is to allow part of the photonic network infrastructure of the Dutch SURFnet6 network to be manipulated by distributed applications to optimize performance. The novelty of StarPlane is that it gives flexibility directly to applications by allowing them to choose one of the multiple logical network topologies in real time, ultimately with subsecond switching times.

5 Performance Evaluation

To perform experiments on DAS-3, we have compiled our applications with OpenMPI.[16] This communication library provides the correct communication setup for message passing programs that need to use multiple heterogeneous networks. For highest performance, we have initialized OpenMPI such that we use optical links between the clusters, and the local Myri-10G network for intracluster communication. We do not use any of the GbE networks. For comparison, we also perform experiments using conventional Internet links between the clusters.

To avoid results that need normalization to a "base node" of a certain speed and capacity, we limit ourselves to scenarios in which the set of compute elements is relatively homogeneous. Also, we do not include scenarios involving multiple cores per CPU, as this would allow us to scale up to two clusters only (i.e., VU and UvA; the remaining clusters have single-core CPUs). Given these restrictions, we perform experiments using three different (multi-)cluster configurations (see also Table 2). Single-cluster runs are always performed at the VU, using a maximum of 64 nodes (= 128 CPUs). Dual-cluster runs are always performed using VU and UvA simultaneously, each with an equal number of nodes. Because it was not possible to use more than 28 UvA nodes, dual-cluster runs have been performed using a maximum of 56 nodes (= 112 CPUs) in total. Four-cluster runs have been performed in a similar manner, for runs using a maximum of 96 nodes (= 192 CPUs).

Keeping in mind the marginal differences between the four cluster systems used in our experiments, we have obtained the sequential run-times for both our algorithms using a single CPU of the VU cluster. These sequential runtimes are used as the base reference for our speedup calculations in all experiments. Also, despite the fact that the nodes in the different DAS-3 clusters run at slightly different clock speeds, in multicluster runs the total workload is spread such that each node is provided with an equal-sized partition of the original input data. Dealing with situations in which much larger speed differences exist between the nodes in a multicluster system is left as future work (and briefly discussed in Sec. 6).

Finally, the image cubes used to evaluate our two algorithms were standard hyperspectral data sets commonly used as testbeds in spectral unmixing applications. Specifically, the AMEE algorithm was applied to the well-known AVIRIS Cuprite data set, available online in reflectance units (http://aviris.jpl.nasa.gov/html/aviris.freedata.html). The portion used in experiments corresponds to the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data, with 614×512 pixels, 224 spectral bands, and a total size of approximately 80 MBytes. The number of endmembers in this scene was set to $p = 19$ after estimating the number of endmembers using the HySime method.[17] On the other hand, the image cube selected for the OSP algorithm consists of a full AVIRIS flightline comprising 4230×750 pixels and 224 spectral bands (approximately 1.5 Gbytes), collected over the World Trade Center (WTC) in New York

Table 2 Specification of the three (multi-)cluster configurations used in our experiments. In every multicluster run, each participating cluster contributed with an equal number of nodes.

No. of clusters	Names of cluster(s) used	Max. no. of nodes used
1	VU	64 (= 128 CPUs)
2	VU, UvA	56 (= 112 CPUs)
4	VU, UvA, LU, UvA-MN	96 (= 192 CPUs)

City, just 5 days after the terrorist attacks that collapsed the two towers at theWTC area. The number of OSP endmember pixels to be detected was set to $p = 30$. In both cases, the accuracy of AMEE and OSP for spectral unmixing purposes has already been widely reported in the literature.[5,14]

5.1 Performance and Speedup

A purely sequential execution revealed that the AMEE algorithm takes over 85 min and 40 s to complete on a single CPU; the OSP algorithm takes over 24 min and 3 s. A parallel single-cluster run using 64 nodes (= 128 CPUs) for the AMEE algorithm resulted in a total execution time of 47.3 s, indicating a close-to-linear speedup of almost 109. A similar single-cluster run for OSP gave a runtime of 20.2 s, and a speedup of over 71.

These results confirm our initial expectations expressed in Sec. 3, and are in line with results reported earlier in Refs. 3 and 5; while it is possible to obtain high speedups for both algorithms, OSP is more difficult to parallelize efficiently than AMEE. In both cases, even if the executions are (almost) perfectly load balanced, efficiency loss is due to the usual causes for parallelization overhead: communication (e.g., to resolve data-dependencies between nodes) and additional memory operations (e.g., to create, copy, and delete intermediate partial data structures).

The speedup graph of Fig. 8 shows that the AMEE algorithm indeed benefits from an increasing number of nodes, even when spread over multiple clusters. When considering multicluster scenarios with all wide-area communication taking place over the StarPlane optical interconnect, obtained results are very similar to the single-cluster measurements. For example, a run using 128 CPUs spread over 4 clusters takes 48.9 s (giving a speedup of 105), as opposed to the above-mentioned 47.3 s for a single-cluster run with the same CPU count. As stated in Sec. 3, this is because the required amount of wide-area communication is very limited. Also note that part of the marginal performance drop can be attributed to the lower clock-speed (2.2 Ghz) of the UvA CPUs.

The most important result, however, is that a four-cluster run allows for concurrent use of more CPUs than can be provisioned by any of the available clusters alone: the use of 192 CPUs spread over four clusters results in a runtime of 32.4 s, giving a speedup of over 158. The speedup graph of Fig. 8 also shows that runs over conventional links are less efficient, albeit only marginally so. Using 192 CPUs over four clusters results in a runtime of 35.2 s, giving a slightly lower speedup of 146. Based on these results, we conclude that the AMEE algorithm indeed can

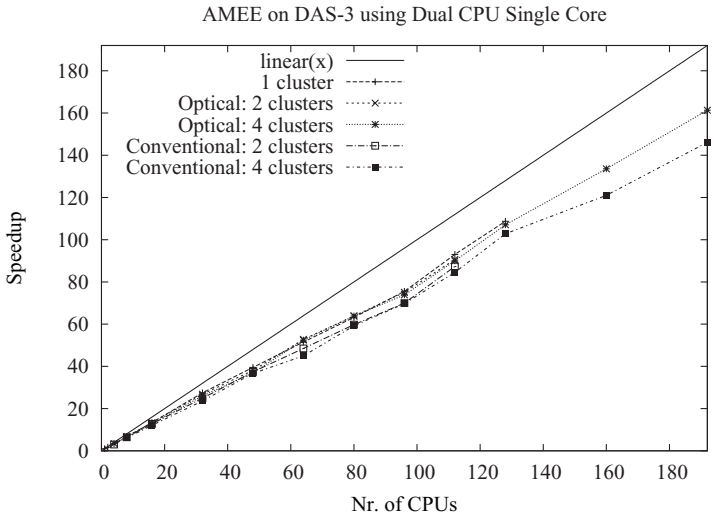

Fig. 8 Speedup results for AMEE on DAS-3.

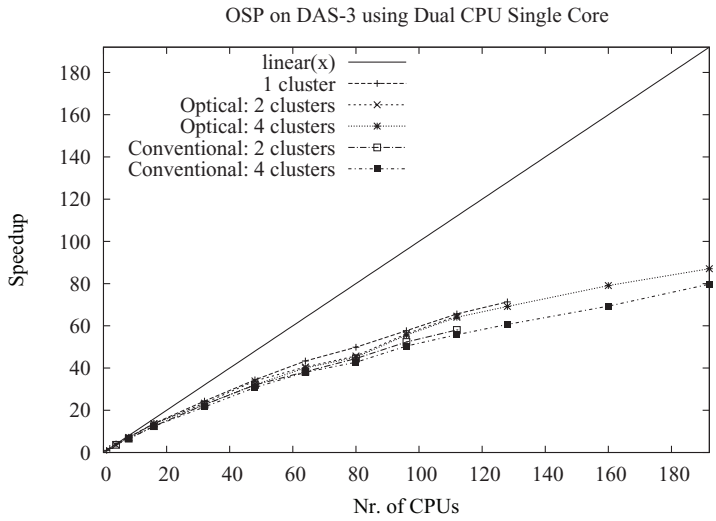

Fig. 9 Speedup results for OSP on DAS-3.

benefit significantly from multicluster execution, with close-to-linear speedup obtained for all measured scenarios. The availability of a high-speed optical wide-area interconnect only gives a limited additional benefit. Given the fact that the input data used for the AMEE algorithm was relatively small, these results also provide excellent prospects for larger input sizes.

Although OSP requiresmore communication, the overall results (shown in Fig. 9) are rather similar. First, the optical multicluster runs are almost identical compared to the single-cluster execution: a run using 128 CPUs spread over 4 clusters takes 20.9 s (for a speedup of 69), where a single-cluster run took 20.2 s. Again, this is due to the limited need for wide-area communication. Scaling up to 192 CPUs spread over four clusters gives a runtime of 16.6 s, and a speedup of 87. As before, the negative impact of conventional wide-area communication is quite limited: a four-cluster run using 192 CPUs takes 18.1 s, with a speedup of just below 80. Given these results, we conclude that—despite the higher need for communication— the OSP algorithm benefits significantly from multicluster execution as well. The speedup graph of Fig. 9 does indicate, however, that scaling up to even larger CPU counts (i.e., 200 CPUs or more) may not provide further benefits.

6 Conclusions and Future Work

In this paper we have applied a wide-area data parallelization approach to two well-known end-member extraction algorithms for hyperspectral unmixing. The considered parallel algorithms, which are representative of a class of fine-grained, tightly-coupled, data parallel regular domain problems, have been tested on a real-world distributed and homogeneous multicluster system, a.o. using a dedicated wide-area optical interconnect. Through experimental evaluation, we have indicated the feasibility of the applied approach.

An important feature of the presented work is that the parallel algorithms were not changed and no further implementation efforts were needed. This means that the parallel implementations of the algorithms were previously developed (for MPI clusters) and now adapted to multicluster environments. As a result, the proposed methodology is easy to apply to already available algorithms, which reduces complexity and increases standardization.

We believe that the presented results are important in hyperspectral imaging for several reasons. First, for many hyperspectral imaging applications it may become feasible to scale up to much larger processor counts than can be provided by a single compute cluster (this performance increase may even come at no cost, since free access to clusters is widely available).

Also, for time-critical applications it will become possible to scale up to problem sizes that better match the specific data analysis problems at hand. Moreover, when available cluster systems are heavily used by others, wide-area data parallel execution may well be the only way to perform realistic experiments.

The presented work is part of a much larger strive to bring the benefits of high-performance and distributed computing to the hyperspectral imaging community. Future work will therefore include the integrated use of state-of-the-art many-core technologies (e.g., Graphics Processing Units, or GPUs) in single- and multicluster execution scenarios, as will be possible with—for example— the state-of-the-art DAS-4 system (http://www.cs.vu.nl/das4/). We also aim to expand our earlier work on hyperspectral imaging for heterogeneous cluster systems to multi-cluster environments. One immediate possibility would be to make use of HeteroMPI (http://hcl.ucd.ie/project/HeteroMPI), a parallel programming library which can automatically balance the load in heterogeneous execution settings. Alternative solutions could come from techniques that partition the input data structures in a dynamic fashion, either relative to the (known) clock-speeds of the participating nodes, or based on problem-specific performance models. Finally, another important future research direction is the consideration of MPI-2 parallel I/O functionality and other collective operations that could speed up several steps of the considered parallel algorithms.

Acknowledgments

This work has been supported by the Netherlands Organization for Scientific Research (NWO) under Grant No. 643.000.602 (JADE-MM: Adaptive High-Performance Distributed Multimedia Computing). This work also has been supported by the European Community's Marie Curie Research Training Networks Programme under reference MRTN-CT-2006–035927, Hyperspectral Imaging Network (HYPER-I-NET). Finally, this work has been supported by the Spanish Ministry of Science and Innovation with the HYPERCOMP/EODIX (AYA2008-05965-C04-02) project. The authors gratefully thank the guest editor and the three anonymous reviewers for their outstanding and highly constructive comments and suggestions, which greatly helped us to improve the quality and presentation of the manuscript.

References

1. C. Chang, *Hyperspectral Data Exploitation: Theory and Applications*, Wiley, New York (2007).
2. R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, M. R. Olah, and O. Williams, "Imaging specroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Env.* **65**(3), 227–248 (1998).
3. A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *J. Parallel Distrib. Comput.* **66**, 345–358 (2006).
4. F. J. Seinstra, J. M. Geusebroek, D. Koelma, C. G. M. Snoek, M. Worring, and A. W. M. Smeulders, "High-performance distributed video content analysis with parallelhorus," *IEEE Multimedia* **14**(4), 64–75 (2007).
5. A. Paz, A. Plaza, and J. Plaza, "Clusters versus GPUs for parallel automatic target detection in remotely sensed hyperspectral images," *EURASIP J. Appl. Signal Process.* (2010).
6. A. Plaza, J. Plaza, and A. Paz, "Parallel heterogeneous CBIR system for efficient hyperspectral image retrieval using spectral mixture analysis," *Concurrecy Comput.: Pract. Exper.* **22**(9), 1138–1159 (2010).
7. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Trans. Geosci. Remote Sens.* **40**, 2025–2041 (2002).

8. H. Ren and C.-I. Chang, "Automatic spectral target recognition in hyperspectral imagery," *IEEE Trans. Aerosp. Electron. Syst.* **39**, 1232–1249 (2003).
9. A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.* **42**, 650–663 (2004).
10. M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," *Proc. SPIE* **3753**, 266–277 (2003).
11. A. Plaza, D. Valencia, J. Plaza, and C.-I. Chang, "Parallel implementation of endmember extraction algorithms from hyperspectral data," *IEEE Geosci. Remote Sens. Lett.* **3**(3), 334–338 (2006).
12. C.-C. Wu, H.-M. Chen, and C.-I. Chang, "Real-time N-finder processing algorithms for hyperspectral imagery," *J. Real-Time Image Process*, http://www.springerlink.com/content/6542658g41xl4168 (2010).
13. J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.* **32**, 779–785 (1994).
14. A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Env.* **113**, 110–122 (2009).
15. D. Heinz and C.-I. Chang, "Fully constrained least squares linear mixture analysis for material quantification in yperspectral imagery," *IEEE Trans. Geosci. Remote Sens.* **39**, 529–545 (2001).
16. R. L. Graham, G. M. Shipman, B. W. Barrett, R. H. Castain, G. Bosilca, and A. Lumsdaine, "Open MPI: A high-performance, heterogeneous MPI," in *Proceedings of HeteroPar'06*, Barcelona, Spain, (2006).
17. J. M. Bioucas-Dias and J. M. P. Nascimento, "Hyperspectral subspace identification," *IEEE Trans. Geosci. Remote Sens.* **46**(8), 2435–2445 (2008).