Parallel Implementation of Endmember Extraction Algorithms From Hyperspectral Data

Antonio Plaza, *Member, IEEE*, David Valencia, Javier Plaza, and Chein-I Chang, *Senior Member, IEEE*

***Abstract*—Automated extraction of spectral endmembers is a crucial task in hyperspectral data analysis. In most cases, the computational complexity of endmember extraction algorithms is very high, in particular, for very high-dimensional datasets. However, the intrinsic properties of available techniques are amenable to the design of parallel implementations. In this letter, we evaluate several parallel algorithms that represent three representative approaches to the problem of extracting endmembers. Two parallel algorithms have been selected to represent a first class of algorithms based on convex geometry concepts. In particular, we develop parallel implementations of approximate versions of the N-FINDR and pixel purity index algorithms, along with a parallel hybrid of both techniques. A second class is given by algorithms based on constrained error minimization and represented by a parallel version of the iterative error analysis algorithm. Finally, a parallel version of the automated morphological endmember extraction algorithm is also presented and discussed. This algorithm integrates the spatial and spectral information as opposed to the other discussed algorithms, a feature that introduces additional considerations for its parallelization. The proposed algorithms are quantitatively compared and assessed in terms of both endmember extraction accuracy and parallel efficiency, using standard AVIRIS hyperspectral datasets. Performance data are measured on Thunderhead, a parallel supercomputer at NASA's Goddard Space Flight Center.**

***Index Terms*—Beowulf cluster, endmember parallelizable spatial/spectral partition, hyperspectral, parallel computing.**

I. Introduction

SPECTRAL mixture analysis is a commonly used hyperspectral analysis procedure, in which the measured spectrum of a mixed pixel is decomposed into a collection of spectrally pure constituent spectra, or endmembers, and a set of correspondent fractions, or abundances, that indicate the proportion of each endmember present in the pixel [1]. Over the last decade, several algorithms have been developed for automatic extraction of spectral endmembers from hyperspectral datasets [2], including the pixel purity index (PPI) [3], N-FINDR [4], or iterative error analysis (IEA) [5]. The above techniques focus on analyzing the data without incorporating information on the spatially adjacent data, i.e., the hyperspectral data is treated not as an image but as an unordered listing of spectral measurements where the spatial coordinates can be shuffled arbitrarily without affecting the analysis. The automated morphological endmember extraction (AMEE) algorithm was recently developed with the purpose of integrating the spatial and spectral information in the search for spectral endmembers [2]. While integrated spatial/spectral analysis holds great promise for Earth science image analysis, it also introduces new processing challenges [6]. Several applications, however, require a response quickly enough for practical use.

In recent years, high-performance computing systems have become more widespread in remote sensing applications, especially with the advent of relatively cheap Beowulf clusters [7]. The new processing power offered by such commodity systems can be used in a wide range of applications. For instance, the development of parallel techniques for endmember extraction may allow rapid inspection of massive data archives and generation of comprehensive spectral libraries and other useful information. This would offer an unprecedented opportunity to explore methodologies in areas such as multisource data registration and mining [8] that previously looked too computationally intensive for practical applications due to the immense files involved.

In this letter, we discuss parallel implementation strategies for standard endmember extraction algorithms. Both PPI and N-FINDR are used to represent a class of algorithms based on convex geometry concepts, although it should be noted that our parallel implementations are based on approximations to these algorithms. Therefore, it is more accurate to refer to our techniques as parallel PPI- and N-FINDR-like algorithms. Parallelization of these methods introduces less data dependencies than those found in the AMEE. A parallel version of IEA has also been developed to represent a class of algorithms based on constrained error minimization.

The remainder of this letter is organized as follows. Section II discusses data partitioning strategies for the design of parallel endmember extraction algorithms. Section III provides a step-by-step description of the parallel algorithms compared in this work. Section IV conducts a quantitative study, where the proposed parallel algorithms are compared in terms of both the quality of the solutions they provide and the time to obtain them, using the Thunderhead Beowulf cluster at NASA's Goddard Space Flight Center. Section IV concludes with some remarks and hints at plausible future research.

II. Data Partitioning Strategies

Two types of data partitioning strategies can be exploited in the design of hyperspectral algorithms [7]: spatial-domain partitioning and spectral-domain partitioning. The former subdivides an input data cube $\mathbf{F}$ into multiple blocks made up of entire pixel vectors, and assigns one or more blocks to each processing element (PE). On other hand, spectral-domain partitioning subdivides the multiband data into blocks made up of contiguous

Manuscript received August 25, 2005; revised October 18, 2005.

A. Plaza, D. Valencia, and J. Plaza are with the Neural Networks and Signal Processing Group, Department of Computer Science, University of Extremadura, 10071 Cáceres, Spain (e-mail: aplaza@unex.es; davaleco@unex.es; jplaza@unex.es).

C.-I Chang is with the Remote Sensing Signal and Image Processing Laboratory, University of Maryland Baltimore County, Baltimore, MD 21250 USA (e-mail: cchang@umbc.edu).

Digital Object Identifier 10.1109/LGRS.2006.871749

spectral bands (subvolumes), and assigns one or more subvolumes to each PE. This approach breaks the spectral identity of the data as opposed to spatial partitioning, which considers each pixel vector $\mathbf{f}(x,y)$ as a unique entity given by its spatial coordinates (x,y). In this approach, pixel vectors are never split amongst several PEs.

In order to decide whether spatial- or spectral-domain partitioning is more appropriate for the design of parallel endmember extraction algorithms, several considerations need to be made. First, it is clear that spatial-domain partitioning provides a *natural* framework for low level image processing, as many operations require the same function to be applied to a small set of elements around each data element present in the image data structure. As a result, this approach is particularly appropriate for window-moving techniques such as the AMEE algorithm [2]. Another issue is the cost of interprocessor communication. In spectral-based partitioning, the calculations for each hyperspectral pixel need to originate from several PEs, and thus require intensive communications. This is generally perceived as a shortcoming for parallel design, because the communication overhead would increase linearly with the increase in the number of PEs, thus limiting parallel performance. A final issue is code reusability: to enhance portability, it is desirable to reuse much of the code for the sequential algorithm in the design of its correspondent parallel version. As will be shown in Section III, the considered algorithms are designed under the assumption that each pixel vector is uniquely represented by its associated spectral signature. Therefore, the introduction of a spectral domain-based decomposition approach would require additional strategies to combine the partial results from several PEs.

TABLE I
DEFINITIONS OF SYMBOLS

Symbol	Description
$\mathbf{F}$	Hyperspectral data cube
$\mathbf{f}$	Vector of values of spectral channels
$\bar{\mathbf{f}}$	Vector of mean values of spectral channels
$\mathbf{f}(x,y)$	Vector of values of spectral channels at position (x,y)
$\mathbf{E}$	Set of endmembers
$\mathbf{e}$	Individual endmember
$\mathbf{G}$	Data cube pre-processed by dimensional reduction
$\mathbf{g}$	Vector of values of spectral channels (reduced data set)
$\mathbf{T}$	Transformation matrix used in dimensional reduction
N	Number of spectral channels in the input data
K	Number of data partitions = number of processors
P	Number of final endmembers to be found
J	Number of random skewers used by PPI method
M	Number of candidate endmembers used by PPI method
T	Threshold for choosing candidate endmembers for PPI
V	Volume of a simplex calculated by N-FINDR method
B	Structuring element used by AMEE method
I_{max}	Maximum number of iterations executed by AMEE
MEI	Morphological eccentricity index used by AMEE
$\mathbf{PSSP}_k$	Parallelizable spatial/spectral partition used by AMEE, with $k=1,\cdots,K$
$\mathbf{skewer}_j$	Random unit vector with $j=1,\cdots,J$
$S^{(k)}(\mathbf{skewer}_j)$	Set of pixels $\mathbf{g}(x,y)$ in partition k which have the largest absolute value of $\lvert\mathbf{skewer}_j \cdot \mathbf{g}(x,y)\rvert$
$N_{PPI}^{(k)}[\mathbf{g}(x,y)]$	Number of times that pixel $\mathbf{g}(x,y)$ from partition k is selected as extreme pixel by the PPI method

III. PARALLEL ENDMEMBER EXTRACTION ALGORITHMS

While the selection of algorithms for performance comparison is subjective, it is our desire to make our selection as representative as possible. For that purpose, three main classes of endmember extraction algorithms will be considered in this section: convex geometry, constrained error minimization, and integrated spatial/spectral developments. The proposed parallel techniques are based on a master–worker spatial domain-based decomposition paradigm, where the master processor sends partial data to the workers and coordinates their actions. Then, the master gathers the partial results provided by the workers and produces a final result. Table I provides several symbol definitions that will be used in our description of parallel algorithms.

A. Parallel Convex Geometry-Based Endmember Extraction

Two standard approaches are considered to represent this class of algorithms: PPI and N-FINDR. The PPI algorithm cannot identify a final set of endmembers by itself [2] and requires an N-dimensional (N-D) visualization tool, available as a built-in companion piece in Research Systems ENVI software, to carry out such goal. Here, we eliminate such interactive component by introducing our own interpretation of the PPI, which is used as a preprocessor whose output is fed to an N-FINDR-like algorithm. The use of PPI to initialize N-FINDR has been previously suggested [9], but this hybrid approach has never been implemented in parallel. A pseudocode description of our parallel approximation of PPI algorithm (from the viewpoint of the master processor) is given below. The hybrid-PPI algorithm consists of three parallel steps applied sequentially: dimensionality reduction, pixel purity index-based preprocessing, and N-FINDR-like endmember selection.

1) Parallel Dimensionality Reduction:

Input: N-D data cube $\mathbf{F}$, Number of endmembers P.
Output: P-D data cube $\mathbf{G}$.

1) Divide the original data cube $\mathbf{F}$ into K partitions, where K is the number of workers, such that there is no overlapping among different partitions.
2) Calculate the N-D mean vector $\bar{\mathbf{f}}$ concurrently, where each component is the average of the pixel values of each spectral band of the input data. This vector is formed at the master once all the processors have finished their parts.
3) Broadcast vector $\bar{\mathbf{f}}$ to all workers so that each worker computes the covariance component and forms a covariance sum, which is sent to the master.
4) Calculate the covariance matrix sequentially as the average of all the matrices calculated in step 3).
5) Obtain a transformation matrix $\mathbf{T}$ by calculating and sorting the eigenvectors of the covariance matrix according to their eigenvalues, which provide a measure of their variances. As a result, the spectral content is forced into the front components. Since the degree of data dependency of the calculation is high and its complexity is related to the number of spectral bands rather than the image size, this step is done sequentially at the master.
6) Transform each N-D pixel vector in the original image by $\mathbf{g}(x,y) = \mathbf{T} \cdot [\mathbf{f}(x,y) - \bar{\mathbf{f}}]$. This step is done in parallel, where all workers transform their respective data portions. The results are sent to the master, which retains the first P components of the resulting cube $\mathbf{G}$ for subsequent processing [10].

2) Parallel Pixel Purity Index-Like Preprocessing:

Input: P-D cube $\mathbf{G}$, Number of skewers J, Threshold value T.
Output: set of M initial candidate endmembers $\{\mathbf{e}_m^{(0)}\}_{m=1}^{M}$.

1) Generate a set of J randomly generated unit P-D vectors called "skewers," denoted by $\{\mathbf{skewer}_j\}_{j=1}^{J}$, and broadcast the entire set to all the workers.

2) For each $\mathbf{skewer}_j$, project all the data sample vectors at each local partition k onto $\mathbf{skewer}_j$ using $|\mathbf{skewer}_j \cdot g(x,y)|$ to find sample vectors at its extreme positions, and form an extrema set for $\mathbf{skewer}_j$ denoted by $S^{(k)}(\mathbf{skewer}_j)$. Define an indicator function of a set S by $I_S(\mathbf{x}) = \begin{cases} 1; & \text{if } \mathbf{x} \in S \\ 0; & \text{if } \mathbf{x} \notin S \end{cases}$, and use it to calculate $N_{PPI}^{(k)}[\mathbf{g}(x,y)] = \sum_j I_{S^{(k)}(\mathbf{skewer}_j)}[\mathbf{g}(x,y)]$ for each pixel vector $\mathbf{g}(x,y)$ at the local partition (this calculation is equivalent to the PPI count calculation in the original PPI algorithm [3]). Select those pixels with $N_{PPI}^{(k)}[\mathbf{g}(x,y)] > T$ and send them to the master.
3) The master collects all the partial results and merges them together to form a set of candidate endmembers $\{\mathbf{e}_m^{(0)}\}_{m=1}^{M}$.

3) Parallel N-FINDR-Like Endmember Selection:

Input: P-D cube $\mathbf{g}$, set of M candidate endmembers $\{\mathbf{e}_m^{(0)}\}_{m=1}^{M}$.
Output: set of J final endmembers $\{\mathbf{e}_j\}_{j=1}^{J}$.

1) Select a random set of P initial pixels $\{\mathbf{e}_p^{(0)}\}_{p=1}^{P} \subseteq \{\mathbf{e}_m^{(0)}\}_{m=1}^{M}$ and find $V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_P^{(0)})$ the volume of the simplex defined by $\{\mathbf{e}_p^{(0)}\}_{p=1}^{P}$, denoted by $S(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_P^{(0)})$.
2) Calculate the volume of P simplices, $V(\boldsymbol{g}(x,y), \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_P^{(0)}) \ldots, V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \boldsymbol{g}(x,y))$ in parallel each of which is formed by replacing one endmember $\mathbf{e}_p^{(0)}$ with the sample vector $\mathbf{g}(x,y)$. Each worker performs replacements using pixels in its local partition.
3) If none of these P recalculated volumes is greater than $V(\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_P^{(0)})$, then no endmember in $\{\mathbf{e}_p^{(0)}\}_{p=1}^{P}$ is replaced. Otherwise, the master replaces the endmember which is absent in the largest volume among the P simplices with the vector $\mathbf{g}(x,y)$. Let such endmember be denoted by $\mathbf{e}_q^{(0)}$. A new set of endmembers is produced by letting $\mathbf{e}_q^{(1)} = \mathbf{g}(x,y)$ and $\mathbf{e}_p^{(1)} = \mathbf{e}_p^{(0)}$ for $p \neq q$.
4) Repeat from step 2) until no replacements occur.

It should be noted that Section III-A2 could be considered an optional step. If this step is eliminated, then the resulting algorithm can be regarded as a parallel version of N-FINDR algorithm, referred to as P-FINDR from now on. We must emphasize, however, that there are a number of differences between our proposed parallel N-FINDR-like algorithm and the original one, including the original projection via PCA (accomplished by a maximum noise fraction transformation in the original algorithm [4]) and in the endmember finding part of the algorithm itself. Similar comments can be made for our PPI-like algorithm in Section III-A2. To address this issue, we include ENVI's PPI and the N-FINDR software package distributed by Technical Research Associates, Inc. (TRA) in experiments.

B. Parallel Constrained Error Minimization

The IEA algorithm produces a set of endmembers *sequentially* as opposed to PPI and N-FINDR, which generate the final endmember set in single-shot mode. As shown below, parallelization of IEA requires several master–worker communications during the execution.

Parallel Iterative Error Analysis (P-IEA)

Input: N-D data cube $\mathbf{F}$, Number of endmembers P.
Output: set of final endmembers $\{\mathbf{e}_p\}_{j=1}^{P}$.

1) Divide the original data cube $\mathbf{F}$ into K partitions without overlapping and send them to the workers.
2) Calculate the N-D mean vector $\bar{\mathbf{f}}$ concurrently. The master forms $\bar{\mathbf{f}}$ after the workers finish their parts.
3) Broadcast $\bar{\mathbf{f}}$ to all workers so that each worker forms an error image by assuming that all pixels in the local partition are made up of $\bar{\mathbf{f}}$ with 100% abundance.
4) Each worker finds the pixel $\mathbf{f}(x,y)$ in the local partition with the largest abundance estimation error, i.e., the pixel which has largest least squares error if it is represented in terms of a fully constrained linear mixture of the pixels found by previous iterations [5]. This pixel, $\mathbf{p}_k$, and its error score, $\mathcal{S}_k$, are sent to the master.
5) The master selects a first endmember $\mathbf{e}_1$ as the pixel $\mathbf{p}_k$ with the maximum associated error score $\mathcal{S}_k$, for $k = 1, \ldots, K$ and broadcasts $\mathbf{E} = [\mathbf{e}_1]$ to all the workers.
6) Repeat from step 4) using $\mathbf{E}$ instead of $\bar{\mathbf{f}}$, and repeatedly incorporate a new endmember $\mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_P$ to $\mathbf{E}$ until $\mathbf{E} = \{\mathbf{e}_p\}_{p=1}^{P}$, in which case the algorithm is terminated.

There is a salient difference between the parallel algorithms described in Sections III-A and B. Most notably, the methods in Section III-A must re-search all the endmembers when parameter P is changed, because they provide the final set in a single shot. So, a set of $P-1$ endmembers is not necessarily a subset of P endmembers generated by the same algorithm. Quite opposite, the method in Section III-B produces endmembers *sequentially*, so a set of P generated endmembers always includes the set of previously generated $P-1$ endmembers.

C. Parallel Spatial/Spectral Endmember Extraction

A parallel morphological endmember extraction technique is discussed in this subsection [11]. In order to extend morphological operations to hyperspectral imagery, we impose an ordering relation in terms of spectral purity in the set of pixel vectors lying within a spatial neighborhood B, known as structuring element (SE), which is translated over the spatial domain. To do so, we define a cumulative distance between each pixel vector $\mathbf{f}(x,y)$ and all pixel vectors in the spatial neighborhood given by B as follows:

$$D_B[\mathbf{f}(x,y)] = \sum_s \sum_t \mathrm{SAM}[\mathbf{f}(x,y), \mathbf{f}(s,t)],\ (s,t) \in Z^2(B) \tag{1}$$

where SAM is the spectral angle mapper [1], defined by $\mathrm{SAM}[\mathbf{f}(x,y), \mathbf{f}(s,t)] = \cos^{-1}[(\mathbf{f}(x,y) \cdot \mathbf{f}(s,t))/(\|\mathbf{f}(x,y)\| \cdot \|\mathbf{f}(s,t)\|)]$. Based on the above distance, extended morphological erosion and dilation can be respectively used to extract the most highly pure and the most highly mixed pixel in the B-neighborhood, defined by $Z^2(B)$, as follows:

$$(\mathbf{f} \Theta B)(x,y) = \underset{(s,t) \in Z^2(B)}{\operatorname{argmin}} \{D_B[\mathbf{f}(x+s, y+t)]\} \tag{2}$$

$$(\mathbf{f} \oplus B)(x,y) = \underset{(s,t) \in Z^2(B)}{\operatorname{argmax}} \{D_B[\mathbf{f}(x+s, y+t)]\}. \tag{3}$$

Morphological operations rely on a window-moving strategy that introduces additional considerations for parallelization. In previous work [12], we have defined the concept of parallelizable spatial/spectral partition (PSSP) as a partition of the input data in the spatial domain that can be processed in parallel without the need for additional communication and/or synchronization. Therefore, a PSSP may be seen as a block of data that can be processed independently, using smaller, window-based (SE) elements. In order to avoid interprocessor communication when the SE computation is split among processing nodes, a data-replication function is implemented to avoid accesses outside the local domain of each partition. Fig. 1 illustrates the function using a toy example based on two data partitions.

As shown in Fig. 1, a *scratch line* is replicated in the two partitions, thus allowing SE-based parallel processing of the two PSSPs without accesses outside their local domain. It should be

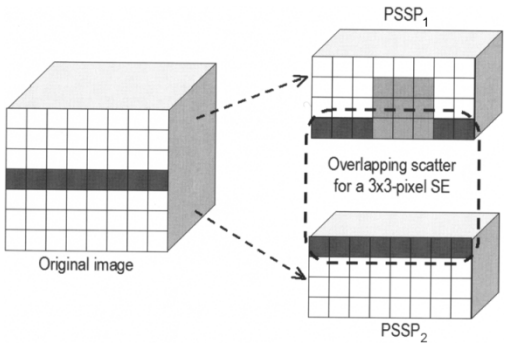

Fig. 1. Overlapping scatter to avoid interprocessor communication.

noted that the example in Fig. 1 gives a simplified view. Depending on how many adjacent partitions are involved (and the size of the SE), it may be necessary to place a *scratch border* of increased width around each partition. It is worth noting that our implementation makes use of a constant 3×3-pixel SE through iterations. Instead of increasing the size of the SE, we replace the original cube $\mathbf{F}$ or, equivalently, a local PSSP in parallel processing, by the resulting cube after a dilation operation using B. This allows us to perform multiscale analysis without increasing the scratch border size between iterations, thus minimizing redundant computations.

AMEEPAR Algorithm

Input: N-D cube $\mathbf{F}$, Iterations $I_{\max}$, Number endmembers P.
Output: set of final endmembers $\{\mathbf{e}_p\}_{p=1}^{P}$.

1) Scatter K partial data structures $\{\mathbf{PSSP}_k\}_{k=1}^{K}$ of $\mathbf{F}$ (with their scratch borders) to each of the K workers.
2) Using parameters $I_{\max}$ and P, each worker executes the sequential AMEE algorithm locally at each processor for the corresponding $\mathbf{PSSP}_k$:
   - 2.1) Set $i = 1$ and initialize a morphological eccentricity index score $MEI(x, y) = 0$ for each local pixel.
   - 2.2) Move B (a fixed $3 \times$ three-pixel SE) through all the pixels of the local $\mathbf{PSSP}_k$ and calculate the maximum and the minimum pixels using (2) and (3), respectively. Update the MEI score at each pixel with the SAM between the maximum and the minimum.
   - 2.3) Set $i = i + 1$. If $i = I_{\max}$ then go to step 2.4). Otherwise, replace $\mathbf{PSSP}_k$ by its dilation using B and go to step 2.2).
   - 2.4) Select a set of P pixel vectors in the local partition with the highest MEI scores.
3) The master gathers all the individual endmember sets provided by the workers and forms a unique set $\{\mathbf{e}_p\}_{p=1}^{P}$ by calculating the SAM for all possible pairs in parallel.

IV. Experimental Results

The proposed parallel algorithms were compared on Thunderhead, a Beowulf cluster at NASA's Goddard Space Flight Center. It is composed of 256 Intel Xeon 2.4-GHz nodes, each with 1 GB of local memory. The implementations were carried out using the C++ programming language with calls to message passing interface (MPI) [13]. The hyperspectral dataset used in experiments is the well-known AVIRIS Cuprite scene, available online in reflectance units from http://aviris.jpl.nasa.gov/html/aviris.freedata.html. The sector selected for experiments is the one labeled as f970619t01p02_r02_sc03.a.rfl. This scene has been widely used to validate the performance of endmember extraction algorithms. The scene comprises a relatively large area (614 lines by 512 samples and 20-m pixels) and 224 spectral bands between 0.4 and 2.5 μm, with nominal spectral resolution of 10 nm. Bands 1–3, 105–115 and 150–170 were removed prior to the analysis due to water absorption and low SNR in those bands. The site is well understood mineralogically, and has several exposed minerals of interest including alunite, buddingtonite, calcite, kaolinite and muscovite. Reference ground signatures of the above minerals, available in the form of a U.S. Geological Survey library (USGS) (http://speclab.cr.usgs.gov/spectral-lib.html) will be used to assess endmember signature purity in this work.

TABLE II
SAM-Based Scores Between USGS Signatures and Algorithm-Derived Endmember Signatures

Algorithm	Alunite	Buddint.	Calcite	Kaolinite	Muscovite
Hybrid-PPI (2745)	0.084	0.106	0.105	0.136	0.108
P-FINDR (695)	0.094	**0.052**	**0.065**	0.105	0.098
P-IEA (9567)	0.091	0.103	0.093	0.078	0.081
AMEEPAR (1874)	**0.073**	0.071	0.084	**0.064**	**0.077**

An experiment-based cross-examination of algorithm endmember extraction accuracy is presented in Table II, which tabulates the SAM scores obtained after comparing library spectra with the corresponding endmembers extracted by the parallel algorithms. The smaller the SAM values across the five minerals considered in Table II, the better the results.

It should be noted that Table II only displays the smallest SAM scores of all endmembers with respect to each USGS signature for each algorithm. For illustrative purposes, single-processor times for all parallel algorithms are given in Table II (in the parentheses). In all cases, the number of endmembers to be extracted, P, was set to 22 after using the virtual dimensionality (VD) concept in [1]. For the Hybrid-PPI, parameter T was set to the mean of N_{PPI} scores after $k = 103$ iterations. A previous performance study of our AMEEPAR algorithm revealed that the most satisfactory results for the considered scene were found by setting $I_{\max} = 7$ iterations. The parameter values above are in agreement with those used before in [2]. It is important to note that the output produced by all parallel methods was verified using not only our own serial implementations, but the original versions of the algorithms as well (using the same parameters in both cases). The endmembers found by our parallel implementations were exactly the same as the ones found by our serial implementations of the original algorithms. In the cases of PPI and N-FINDR, we made sure that the same random skewers and random initial endmembers were used, respectively, to guarantee that both the serial and parallel versions were exactly the same. While both P-IEA and AMEEPAR produced the same results as the original algorithms, our parallel PPI- and N-FINDR-like implementations produced slightly different results than those found by ENVI's PPI and TRA's N-FINDR. In particular, 4 out of 22 endmembers produced by ENVI's PPI and 6 out of 22 endmembers produced by TRA's N-FINDR were not included in the respective final endmember sets produced by our Hybrid-PPI and P-FINDR. However, we experimentally tested that the SAM scores between the endmembers that were different between the original and approximate algorithms were always very low (below 0.015), a fact that reveals that the final endmembers sets were almost identical, spectrally. Also, both the original and approximate parallel versions of PPI and

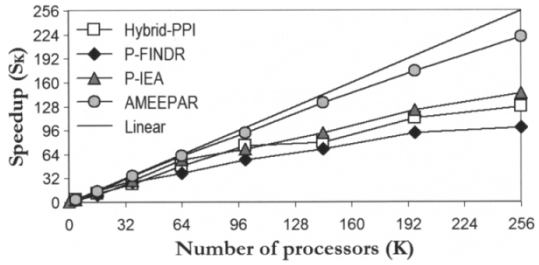

Fig. 2. Performance of parallel algorithms on Thunderhead.

TABLE III
EXECUTION TIME AND SEQUENTIAL PORTION (PARENTHESES) IN SECONDS USING DIFFERENT NUMBERS OF PROCESSORS ON THUNDERHEAD

Algorithm	4	16	36	64	100	144	196	256
H brid-PPI	807(208)	244(64)	112(25)	56(16)	36(11)	34(9)	24(7)	21(6)
P-FINDR	302(81)	73(21)	25(9)	18(7)	12(5)	10(4)	8(3)	7(3)
P-IEA	2733(680)	628(161)	336(87)	168(48)	134(35)	104(29)	77(21)	66(18)
AMEEPAR	580(57)	132(14)	53(8)	30(6)	19(4)	14(3)	11(2)	8(2)

N-FINDR produced exactly the same endmembers for the five minerals listed in Table II.

To empirically investigate the scaling properties of the parallel algorithms, their performance was tested by timing the programs over various numbers of processors. If we approximate the total time required to complete a task on K processors by $T(K) = A_K + (B_K/K)$, where A_K is the sequential (nonparallelizable) portion of the computation and B_K is the parallel portion, then we can simply define the speedup for K processing units as $S_K = T(1)/T(K) \approx (A_K + B_K)/(A_K + (B_K/K))$, where $T(1)$ is the single-processor time. The relationship above, usually expressed as an inequality to account for parallelization overhead, is known as Amdahl's law [14]. The sequential portion A_K is proportionally more important as the number of processors increase and, thus, the performance of the parallelization is generally degraded for a large number of PEs. Fig. 2 plots the speedups achieved by the parallel algorithms as a function of the number of processors.

Results in Fig. 2 reveal that the performance drop from linear speedup in the Hybrid-PPI and P-FINDR algorithms increases as the number of processors is higher. Although the involved calculations were sufficient to gain an impact from using a large number of processors, the sequential computations represented a significant portion of the total processing time in both cases, in particular, when the number of processors was increased (see Table III). A similar effect was also observed for the P-IEA. Although AMEEPAR introduces redundant information, expected to slow down the computation *a priori*, sequential computations for this algorithm were low when compared to the total times, which resulted in closer to linear speedups in Fig. 2. This comes at no surprise since AMEEPAR is a windowing type algorithm as opposed to the other tested algorithms and, therefore, it is expected to scale better. This property allows the algorithm to process the full AVIRIS scene (137 MB in size) in only a few seconds using a moderate number of processors. To conclude this section, we must also acknowledge that TRA's N-FINDR software was able to process the Cuprite scene in 16 s on a standard desktop machine with an AMD Athlon processor at 2.6 GHz and 256 MB of RAM. It is to the credit of N-FINDR programmers that the software package is so fast, although our research indicates that significant opportunities for parallelization of this algorithm are still open.

V. CONCLUSION

This letter has examined several strategies for parallelization of endmember extraction algorithms. Although future work may incorporate additional techniques, three highly representative classes of algorithms were considered in this study. Most available approaches do not take into account the spatial information explicitly, a fact that has generally been perceived as an advantage for the development of parallel implementations. However, experimental results in this letter suggest that spatial/spectral endmember extraction approaches may be more "pleasingly parallel," mainly due to the windowing nature of such algorithms, but also because they can reduce sequential computations at the master node and involve only minimal communication between the parallel tasks, namely, at the beginning and ending of such tasks.

REFERENCES

[1] C.-I Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York: Kluwer, 2003.
[2] A. Plaza, P. Martínez, R. Pérez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 650–663, Mar. 2004.
[3] J. W. Boardman, F. A. Kruse, and R. O. Green, "Mapping target signatures via partial unmixing of AVIRIS data," in *Summaries of JPL Airborne Earth Science Workshop*, Pasadena, CA, 1995.
[4] M. E. Winter, "N-FINDR: an algorithm for fast autonomous spectral endmember determination in hyperspectral data," in *Proc. SPIE Conf. Imaging Spectrometry V*, 1999, vol. 3753, pp. 266–277.
[5] R. A. Neville, K. Staenz, T. Szeredi, J. Lefebvre, and P. Hauff, "Automatic endmember extraction from hyperspectral data for mineral exploration," in *21st Can. Symp. Remote Sensing*, Jun. 1999, pp. 21–24.
[6] J. C. Tilton, "Method for implementation of recursive hierarchical segmentation on parallel computers," U.S. Pat. Office, Washington, DC U.S. Pending Published Application 09/839147, 2005 [Online]. Available: http://www.fuentek.com/technologies/rhseg.htm.
[7] J. A. Gualtieri, "A parallel processing algorithm for remote sensing classification," in *Proc. Airborne Earth Science Workshop*, Pasadena, CA, 2004 [Online]. Available: http://aviris.jpl.nasa.gov/html/aviris.documents.html.
[8] J. Le Moigne, W. J. Campbell, and R. F. Cromp, "An automated parallel image registration technique based on the correlation of wavelet features," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 8, pp. 1849–1864, Aug. 2002.
[9] J. Theiler, D. Lavernier, N. R. Harvey, S. J. Perkins, and J. J. Szymanski, "Using blocks of skewers for faster computation of pixel purity index," in *Proc. SPIE*, 2000, vol. 4132, pp. 61–71.
[10] T. El-Ghazawi, S. Kaewpijit, and J. Le Moigne, "Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality," in *Proc. 3rd IEEE Int. Conf. Cluster Computing (Cluster'01)*, 2001, pp. 102–112.
[11] A. Plaza, P. Martínez, R. Pérez, and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 9, pp. 2025–2041, Sep. 2002.
[12] D. Valencia, A. Plaza, P. Martínez, and J. Plaza, "On the use of cluster computing architectures for implementation of hyperspectral analysis algorithms," in *Proc. 10th IEEE Symp. Computers Communications*, 2005, pp. 995–1000.
[13] W. Gropp, E. Lust, and A. Skjellum, *Using MPI: Portable Parallel Programming With Message Passing Interface*. Cambridge, MA: MIT Press, 1996.
[14] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Mateo, CA: Morgan Kaufmann, 2002.