EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms

Alfredo Remón[1*], Sergio Sánchez[2], Sergio Bernabé[2], Enrique S Quintana-Ortí[1] and Antonio Plaza[2]

Abstract

Hyperspectral imaging is a growing area in remote sensing in which an imaging spectrometer collects hundreds of images (at different wavelength channels) for the same area on the surface of the Earth. Hyperspectral images are extremely high-dimensional, and require on-board processing algorithms able to satisfy near real-time constraints in applications such as wildland fire monitoring, mapping of oil spills and chemical contamination, etc. One of the most widely used techniques for analyzing hyperspectral images is spectral unmixing, which allows for sub-pixel data characterization. This is particularly important since the available spatial resolution in hyperspectral images is typically of several meters, and therefore it is reasonable to assume that several spectrally pure substances (called *endmembers* in hyperspectral imaging terminology) can be found within each imaged pixel. There have been several efforts towards the efficient implementation of hyperspectral unmixing algorithms on architectures susceptible of being mounted onboard imaging instruments, including field programmable gate arrays (FPGAs) and graphics processing units (GPUs). While FPGAs are generally difficult to program, GPUs are difficult to adapt to onboard processing requirements in spaceborne missions due to its extremely high power consumption. In turn, with the increase in the number of cores, multi-core platforms have recently emerged as an easier to program platform compared to FPGAs, and also more tolerable radiation and power consumption requirements. However, a detailed assessment of the performance versus energy consumption of these architectures has not been conducted as of yet in the field of hyperspectral imaging, in which it is particularly important to achieve processing results in real-time. In this article, we provide a thoughtful perspective on this relevant issue and further analyze the performance versus energy consumption ratio of different processing chains for spectral unmixing when implemented on multi-core platforms.

1 Introduction

Hyperspectral imaging instruments are capable of collecting hundreds of images, corresponding to different wavelength channels, for the same area on the surface of the Earth [1]. For instance, NASA is continuously gathering imagery data with instruments such as the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS), which operates in the 0.4–2.5 μm spectral range, with 10 nm spectral resolution and 30 m spatial resolution [2]. As a follow-up to the success of AVIRIS (an airborne instrument), a new generation of satellite instruments for Earth observation are operating or under development (see Table 1). As indicated there, most current hyperspectral missions are spaceborne in nature [3].

One of the main problems in the analysis of hyperspectral data cubes is the presence of mixed pixels [4,5], which arise when the spatial resolution of the sensor is not fine enough to separate spectrally distinct materials (see Figure 1). Spectral unmixing [6-8] is one of the most popular techniques to analyze hyperspectral data. It involves the separation of a pixel spectrum into its pure component (endmember) spectra [9,10], and the estimation of the abundance value for each endmember [11,12]. The linear mixture model assumes single scattering between the endmember substances resulting from the fact that they are sitting side-by-side within the field of view of the imaging instrument (see Figure 2a). On the other hand, the nonlinear mixture model [13-15] assumes nonlinear

*Correspondence: remon@icc.uji.es
[1]Department of Engineering and Computer Sciences, University Jaime I, Castellon, Spain
Full list of author information is available at the end of the article

Table 1 Overview of some present and future remote sensing missions including hyperspectral sensors

	EO-1 Hyperion[a]	Prisma[b]	EnMAP[c]	HyspIRI[d]
Country of origin	USA	Italy	Germany	USA
Spatial resolution (m)	30	5–30	30	60
Revisit time (days)	16	3/7	4	18
Spectral range (nm)	400–2,500	400–2,500	420–2,450	380–2,500
Spectral resolution (nm)	10	10	6.5–10	10
Swath width (km)	7.7	30	30	120
Earth coverage	Partial	Full	Full	Full
Launch	2000	2012	2015	2018

[a]http://eo1.gsfc.nasa.gov.
[b]http://www.asi.it/en/flash_en/observing/prisma
[c]http://www.enmap.org.
[d]http://hyspiri.jpl.nasa.gov.

interactions and multiple scattering between endmember substances (see Figure 2b). In practice, the linear model is more flexible and can be easily adapted to different analysis scenarios [16]. It can be simply defined as follows [17]:

$$\mathbf{y} = \mathbf{E}\mathbf{a} + \mathbf{n} = \sum_{i=1}^{p} \mathbf{e}_i a_i + \mathbf{n}, \tag{1}$$

where $\mathbf{y}$ is an n-dimensional pixel vector given by a collection of values at different wavelengths, $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$ is a matrix containing p endmembers, $\mathbf{a} = [a_1, a_2, \ldots, a_p]$ is a p-dimensional vector containing the abundance fractions for each of the p endmembers in $\mathbf{y}$, and $\mathbf{n}$ is a noise term. Generalizing this expression for all the hyperspectral pixels in the scene (in compact matrix notation) yields $\mathbf{Y} = \mathbf{EA} + \mathbf{N}$, where $\mathbf{Y}$ is the full hyperspectral image with m pixels, each with n bands, $\mathbf{E}$ is the endmember matrix with dimensions $n \times p$, $\mathbf{A}$ is an $p \times m$ matrix containing the endmember abundances for each pixel of the scene, and $\mathbf{N}$ is a $n \times m$ noise matrix. With the aforementioned notation in mind, solving the linear mixture model involves: (1) estimating the number of endmembers, p, in the hyperspectral scene; (2) identifying a collection of $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{p}$ endmembers; and (3) estimating the fractional abundances of the p endmembers for each pixel in the hyperspectral data set.

Several techniques have been proposed to solve this problem under the linear mixture model assumption in recent years (see [18-43], among several others), but all of them are quite expensive in computational terms. Although these techniques map nicely to high performance computing platforms such as commodity clusters [44], these systems are difficult to adapt to on-board processing requirements introduced by applications with real-time constraints such as wild land fire tracking,

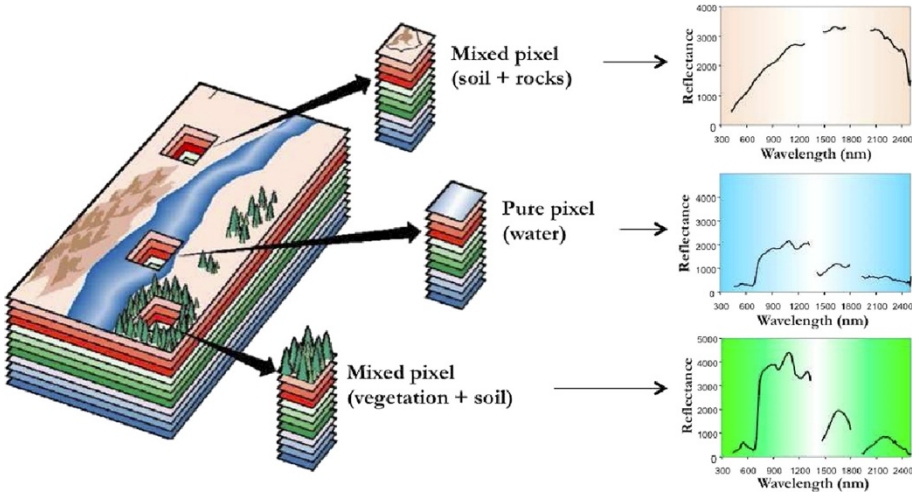


Figure 1 Presence of mixed pixels in remotely sensed hyperspectral images.

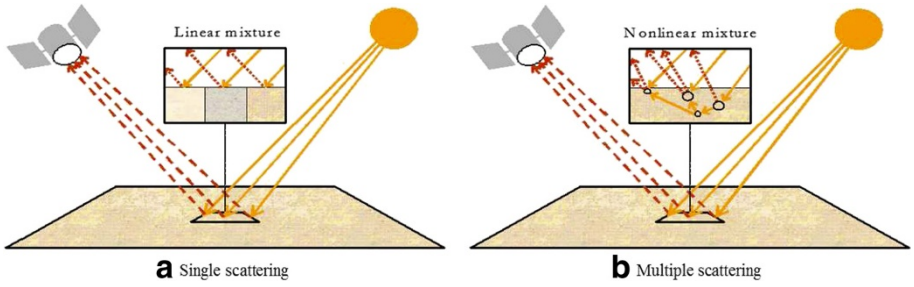


Figure 2 Mixture models in remotely sensed hyperspectral imaging. (**a**) Linear and (**b**) nonlinear.

biological threat detection, monitoring of oil spills and other types of chemical contamination [45-47]. In those cases, low-weight integrated components such as field programmable gate arrays (FPGAs) [48-50] and graphics processing units (GPUs) [51,52] have the potential to reduce payload in current and future Earth observation missions, which are mainly spaceborne in nature as indicated by Table 1. Furthermore, GPUs offer fast processing at low cost (in the literature, so far GPUs have been the only platform shown to be able to process hyperspectral images in real-time) and easy programmability which are very appealing for future remote sensing missions [53-58]. On the negative side, FPGAs are difficult to program and there is currently a lack of efficient implementations of a full spectral unmixing chain for hyperspectral image processing in these type of architectures. Besides, GPUs are currently not suitable for spaceborne missions due to their high power consumption and the lack of radiation tolerance. These aspects are critical for the definition of the mission (payload) and its overall success and lifetime. In turn, with the increase in the number of cores, multi-core platforms have recently emerged as an easier to program platform as compared to FPGAs, and also more flexible to accomodate to radiation tolerance and power consumption requirements. Nevertheless, a detailed assessment of the performance versus energy consumption of these architectures has not been conducted as-of-yet in the field of hyperspectral imaging, in which it is particularly important to achieve processing results in real-time with low energy cost.

In this article, we provide a thoughtful perspective on this relevant issue and further analyze the performance versus energy consumption ratio of different processing chains for spectral unmixing when implemented on multi-core platforms. This kind of analysis has not been previously conducted in the literature, and in our opinion it is very important in order to really calibrate the possibility of using multi-core platforms for efficient hyperspectral image processing in real remote sensing missions. The remainder of the article is organized as follows. Section 2 reviews the different modules that conform the considered unmixing chains discussed in this work. Section 3 describes the parallel implementation of these modules on multi-core platforms. Section 4 presents an experimental evaluation of the proposed implementations in terms of unmixing accuracy, parallel performance and energy consumption, reporting several multi-core implementations able to provide real-time analysis performance and discussing their energy consumption requirements. Section 5 concludes the article with some remarks and hints at plausible future research lines.

2 Spectral unmixing modules

In this section, we describe different modules for spectral unmixing of hyperspectral data. These modules will be then used to define spectral unmixing chains, which are composed of three main stages: (1) estimating the number of endmembers in the original hyperspectral scene; (2) identifying a collection of endmembers in the scene; and (3) estimating the fractional abundances of endmembers in each pixel of the scene. In the following, we describe different methods in each category, offering a few remarks that describe computational aspects such as the operations that are involved and the cost of the algorithm in terms of floating-point arithmetic operations (flops). In the following cost expressions, for simplicity, we neglect lower order terms, taking into account that in practice $p, n \ll m$.

2.1 Methods for estimating the number of endmembers

This section introduces two different methods for estimating the number of endmembers: virtual dimensionality (VD) [59] and hyperspectral signal identification by minimum error (HySime) [60].

2.1.1 Virtual dimensionality (VD)

The VD method first calculates the eigenvalues of the covariance matrix $\mathbf{K}_{L\times L} = 1/N(\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$ and the correlation matrix $\mathbf{R}_{L\times L} = \mathbf{K}_{L\times L} + \overline{\mathbf{Y}}\overline{\mathbf{Y}}^T$, referred to as covariance-eigenvalues and correlation-eigenvalues, for each of the spectral bands in the original hyperspectral

image $\mathbf{Y}$. The VD concept follows the "pigeon-hole principle". If we represent a signal source by a pigeon and a spectral band by a hole, we can use a spectral band to accommodate one source. Thus, if a distinct spectral signature makes a contribution to the eigenvalue-represented signal energy in one spectral band, then its associated correlation eigenvalue will be greater than its corresponding covariance eigenvalue in this particular band. Otherwise, the correlation eigenvalue would be very close to the covariance eigenvalue, in which case only noise would be present in this particular band. By applying this concept, a Neyman–Pearson detector [59] is introduced to formulate the issue of whether a distinct signature is present or not in each of the spectral bands of $\mathbf{Y}$ as a binary hypothesis testing problem. Here, the decision is made based on an input parameter of the algorithm that is called the false alarm probability or P_F, which is used to establish the sensitivity of the algorithm in terms of how much error can be tolerated in the identification of the actual number of endmembers in the image data. With this interpretation in mind, the issue of determining an estimate $\hat{p}$ for the number of endmembers is further simplified and reduced to a specific value of P_F that is preset by the Neyman–Pearson detector.

From the computational point of view, the most complex operation in this algorithm is related with the calculation of the covariance and correlation matrices which need to be compared in order to determine the number of endmmebers. If we recall that the number of bands of the hyperspectral image is denoted by n, the total cost of each calculation is given by n^2 flops.

2.2 HySime method

The HySime method consists of two parts. Algorithm 1 describes the noise estimation part, which obtains an $N \times L$ matrix $\widehat{\xi}$ containing an estimation of the noise present in the original hyperspectral image $\mathbf{Y}$ [60]. This algorithm follows an approach which addresses the high correlation exhibited by close spectral bands. The main advantage of Algorithm 1 is that the computational complexity is substantially lower than that of other algorithms for noise estimation in hyperspectral data in the literature. Additional details about Algorithm 1 can be found in [60] and we do not repeat them for space considerations. On the other hand, Algorithm 2 describes the signal subspace identification part of the algorithm, which first computes the noise correlation matrix $\widehat{\mathbf{R}}_n$ and then computes the signal correlation matrix $\widehat{\mathbf{R}}_x$. Next, the eigenvectors of the signal correlation matrix are obtained and sorted in ascending order. Finally, a minimization function is applied to obtain an estimate $\hat{p}$ of the number of endmembers in the subspace $\widehat{\mathbf{X}}$. The main purpose of this algorithm is to select the subset of eigenvectors that best represents the signal subspace in the minimum mean squared error sense. As in the case of the previous algorithm, the most complex operations are due to the calculation of the covariance and correlation matrices. Again, the total cost of each calculation is given by n^2 flops, where n is the number of bands of the hyperspectral image.

Algorithm 1 Noise estimation

1: **INPUT**: $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$
2: $\mathbf{Z} := \mathbf{Y}^T$, $\widehat{\mathbf{R}} := (\mathbf{Z}^T\mathbf{Z})$
3: $\mathbf{R}' := \widehat{\mathbf{R}}^{-1}$
4: **for** i:=1 to L **do**
5: $\widehat{\beta}_i := ([\mathbf{R}']_{\alpha_i,\alpha_i} - [\mathbf{R}']_{\alpha_i,i}\,[\mathbf{R}']_{i,\alpha_i}/[\mathbf{R}']_{i,i})[\widehat{\mathbf{R}}]_{\alpha_i,i}$ {Note that $\alpha_i = 1, \ldots, i-1, i+1, \ldots, L$ }
6: $\widehat{\xi}_i := \mathbf{z}_i - \mathbf{Z}_{\alpha_i}\widehat{\beta}_i$
7: **end for**
8: **OUTPUT**: $\widehat{\xi}$ {$N \times L$ matrix with the estimated noise}

Algorithm 2 Signal subspace estimation

1: **INPUTS**: $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$, $\widehat{\mathbf{R}}_y \equiv, (\mathbf{Y}\mathbf{Y}^T)/N, \widehat{\xi}$
2: $\widehat{\mathbf{R}}_n := 1/N \sum_i (\widehat{\xi}_i \widehat{\xi}_i^T)$
3: $\widehat{\mathbf{R}}_x := 1/N \sum_i ((\mathbf{y}_i - \widehat{\xi}_i)(\mathbf{y}_i - \widehat{\xi}_i^T))$ {estimate of $\widehat{\mathbf{R}}_x$}
4: $\mathbf{E} := [\mathbf{e}_1, \ldots, \mathbf{e}_L]$ {$\mathbf{e}_i$ are eigenvectors of $\widehat{\mathbf{R}}_x$}
5: $\delta := [\delta_1, ..., \delta_L]$
6: $(\widehat{\delta}, \widehat{\pi})$:= sort(δ) {sort δ_i in ascending order of the corresponding eigenvalues and save the permutation $\widehat{\pi}$}
7: $\widehat{p}$:= number of terms $\widehat{\delta}_i < 0$ {This is a simplified description of the minimization function, which is fully described in [60] and ommitted here for space considerations.}
8: **OUTPUT**: $\widehat{\mathbf{X}} = \langle [\widehat{\mathbf{e}}_{i1}, \cdots, \widehat{\mathbf{e}}_{ip}] \rangle$ {Estimated subspace}

2.3 Methods for endmember identification

This section introduces two different methods for identifying the endmember signatures in the hyperspectral data: orthogonal subspace projection with Gram-Schmidt orthogonalization (OSP-GS) [18] and N-FINDR [23].

2.3.1 Orthogonal subspace projection with Gram-Schmidt orthogonalization (OSP-GS)

The OSP algorithm [18] was originally developed to find spectrally distinct signatures using orthogonal projections. For this work, we have used an optimization of this algorithm (see [61,62]) which allows calculating the OSP without requiring the computation of the inverse of the matrix that contains the endmembers already identified in the image. This operation, which is difficult to implement in parallel, is accomplished using the Gram-Schmidt method for orthogonalization. This process selects a finite set of linearly independent vectors $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_p\}$ in the inner product space $\mathbf{R}^n$ in which the original hyperspectral image is defined, and generates an orthogonal

set of vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_p\}$ which spans the same *p-dimensional* subspace of $\mathbf{R}^n$ ($p \leq n$) as $\mathbf{A}$. In particular, $\mathbf{B}$ is obtained as follows:

$$\begin{aligned}
&\mathbf{b}_1 = \mathbf{a}_1, && \mathbf{e}_1 = \frac{\mathbf{b}_1}{\|\mathbf{b}_1\|}\\
&\mathbf{b}_2 = \mathbf{a}_2 - \text{proj}_{\mathbf{b}_1}(\mathbf{a}_2), && \mathbf{e}_2 = \frac{\mathbf{b}_2}{\|\mathbf{b}_2\|}\\
&\mathbf{b}_3 = \mathbf{a}_3 - \text{proj}_{\mathbf{b}_1}(\mathbf{a}_3) - \text{proj}_{\mathbf{b}_2}(\mathbf{a}_3), && \mathbf{e}_3 = \frac{\mathbf{b}_3}{\|\mathbf{b}_3\|}\\
&\mathbf{b}_4 = \mathbf{a}_4 - \text{proj}_{\mathbf{b}_1}(\mathbf{a}_4) - \text{proj}_{\mathbf{b}_2}(\mathbf{a}_4) - \text{proj}_{\mathbf{b}_3}(\mathbf{a}_4), && \mathbf{e}_4 = \frac{\mathbf{b}_4}{\|\mathbf{b}_4\|}\\
&\vdots && \vdots\\
&\mathbf{b}_p = \mathbf{a}_p - \textstyle\sum_{j=1}^{p-1} \text{proj}_{\mathbf{b}_j}(\mathbf{a}_p), && \mathbf{e}_p = \frac{\mathbf{b}_p}{\|\mathbf{b}_p\|},
\end{aligned} \tag{2}$$

where the projection operator is defined as

$$\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{< \mathbf{a}, \mathbf{b} >}{< \mathbf{b}, \mathbf{b} >}\mathbf{b}, \tag{3}$$

and $< \mathbf{a}, \mathbf{b} >$ denotes the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$.

The sequence $\mathbf{b}_1, \ldots, \mathbf{b}_p$ in Equation (2) represents the set of orthogonal vectors generated by the Gram-Schmidt method, and thus, the normalized vectors $\mathbf{e}_1, \ldots, \mathbf{e}_p$ in (2) form an orthonormal set. As far as $\mathbf{B}$ spans the same p-dimensional subspace of $\mathbf{R}^n$ as $\mathbf{A}$, an additional vector $\mathbf{b}_{p+1}$ computed by following the procedure stated at (2) is also orthogonal to all the vectors included in $\mathbf{A}$ and $\mathbf{B}$. This algebraic assertion constitutes the cornerstone of the OSP method with Gram-Schmidt orthogonalization.

From the computational point of view, this algorithm has to be augmented with some sort of column pivoting that, at each step of the orthogonalization, detects the pixel with maximum projection value among those of the image (see [63] for details). Unfortunately, this requires that each projector is applied to all pixels of the scene, not only to p, yielding a significant increase in the arithmetic cost of the algorithm. Given the $3n$ flops required to apply the projector (3) to one pixel, and the p endmembers that have to be identified, the result is a total cost for the algorithm of $3mnp$ flops.

2.3.2 N-FINDR

The N-FINDR algorithm [23] is one of the most widely used and successfully applied methods for automatically determining endmembers in hyperspectral image data without using *a priori* information. This algorithm looks for the set of pixels with the largest possible volume by *inflating* a simplex inside the data. The procedure begins with a random initial selection of pixels (see Figure 3a). Every pixel in the image must be evaluated in order to refine the estimate of endmembers, looking for the set of pixels that maximizes the volume of the simplex defined by the selected endmembers. The mathematical definition of the volume of a simplex formed by a set of endmember candidates is proportional to the determinant of the set augmented by a row of ones. The determinant is only defined in the case where the number of features is $p - 1$, p being the number of desired endmembers [9]. Since in hyperspectral data typically $n \gg p$, a transformation that reduces the dimensionality of the input data is required. In this work, we use the principal component transform (PCT) [64] for this purpose. The corresponding volume is calculated for every pixel in each endmember position by replacing that endmember and finding the resulting volume. If the replacement results in an increase of volume, the pixel replaces the endmember. This procedure is repeated in iterative fashion until there are no more endmember replacements (see Figure 3b). The method can be summarized by a step-by-step algorithmic description which is given below for clarity:

1. *Feature reduction*. Apply a dimensionality reduction transformation such as PCT to reduce the dimensionality of the data from n to $d = p - 1$, where p is an input parameter to the algorithm (number of endmembers to be extracted). The basic idea of PCT is to orthogonally project the data into a new coordinate system, defined by the variance of the original data, i.e. the direction that accounts for the greatest variance of the original data will be the first coordinate (the principal component) of the transformed system, the second dimension will be the direction with the second largest variance, and so on. PCT requires the computation of the singular values and right singular vectors of $\tilde{\mathbf{Y}} = (\mathbf{Y} - \overline{\mathbf{Y}})^T(\mathbf{Y} - \overline{\mathbf{Y}})$. In particular, consider the singular value decomposition (SVD) $\tilde{\mathbf{Y}} = \mathbf{U}\Sigma\mathbf{V}^T$ [64,65]. Then, the PCT performs the dimension reduction $n \rightarrow (d = p - 1)$ by replacing $\mathbf{Y}$ with the first $p - 1$ columns of $\tilde{\mathbf{Y}}\mathbf{V}$.
2. *Initialization*. Let $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$ be a set of endmembers randomly extracted from the input data.
3. *Volume calculation*. At iteration $k \geq 0$, calculate the volume defined by the current set of endmembers as follows:

$$V\left(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right) = \frac{\left|\det\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \mathbf{e}_1^{(k)} & \mathbf{e}_2^{(k)} & \ldots & \mathbf{e}_p^{(k)} \end{bmatrix}\right|}{(p-1)!}. \tag{4}$$

4. *Replacement*. For each pixel vector $\mathbf{y}$ in the input hyperspectral data, recalculate the volume by testing the pixel in all p endmember positions, i.e., first calculate $V\left(\mathbf{y}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right)$, then calculate $V\left(\mathbf{e}_1^{(k)}, \mathbf{y}, \ldots, \mathbf{e}_p^{(k)}\right)$, and so on until $V\left(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{y}\right)$. If none of the p recalculated volumes is greater than $V\left(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right)$, then

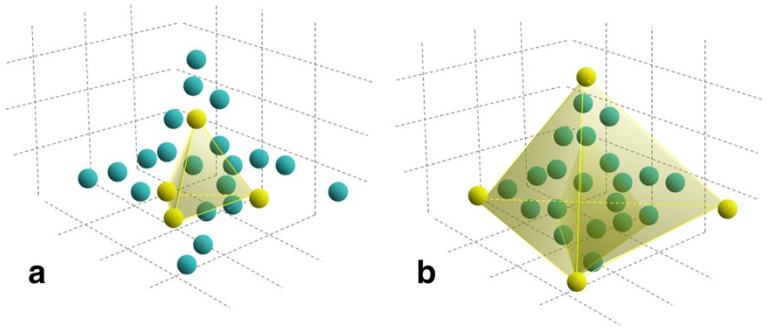


Figure 3 Graphical interpretation of the N-FINDR algorithm in a 3-dimensional space. (a) *N*-FINDR initialized randomly ($p = 4$) and **(b)** final volume estimation by *N*-FINDR.

no endmember is replaced. Otherwise, the combination with maximum volume is retained. Let us assume that the endmember absent in the combination resulting in the maximum volume is denoted by $\mathbf{e}_i^{(k+1)}$. In this case, a new set of endmembers is produced by letting $\mathbf{e}_i^{(k+1)} = \mathbf{y}$ and $\mathbf{e}_l^{(k+1)} = \mathbf{e}_l^{(k)}$ for $l \neq i$. The replacement step is repeated for all the pixel vectors in the input data until all the pixels have been exhausted.

Computationally, this algorithm requires two major operations: feature reduction and volume calculation + replacement. Exploiting that $n \ll m$ and that only a few columns of $\tilde{\mathbf{Y}}\mathbf{V}$ are required, determines that the PCT (first operation) can be computed in only $2mn^2$ flops, which are basically due to the calculation of the SVD of $\tilde{\mathbf{Y}}$. The determination of the volumes are much more expensive. In particular, a straight-forward implementation of the computations of the determinants in steps (3)–(4), via e.g. the LU factorization (with partial pivoting), renders a total cost of $2mp^4/3$ flops, which results from having to compute mp factorizations of $p \times p$ matrices, with a cost of $2p^3/3$ flops per LU factorization. In [63], we describe a refined alternative that, by exploiting simple properties of the LU factorization, reduces this cost to $mp^3 + 2p^4/3$ flops.

As a final comment, it has been observed that different random initializations of *N*-FINDR may produce different final solutions. Thus, our *N*-FINDR algorithm was implemented in iterative fashion, so that each sequential run was initialized with the previous algorithm solution, until the algorithm converges to a simplex volume that cannot be further maximized. Our experiments show that, in practice, this approach allows the algorithm to converge in a few iterations only.

2.4 Methods for abundance estimation

This section introduces two different methods for estimating the abundance fractions: unconstrained least squares (ULS) [65] and non-negative constrained least squares (NCLS) [11].

2.4.1 *Unconstrained least squares*

Once a set of $\mathbf{E} = \{\mathbf{e}_j\}_{j=1}^p$ endmembers has been estimated using an endmember extraction algorithm, an unconstrained p-dimensional estimate of the endmember abundances in a given pixel in $\mathbf{y}$ can be simply obtained (in least squares sense) from the following expression [65]:

$$\hat{\mathbf{a}}^{\mathrm{UC}} = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T\mathbf{y}. \tag{5}$$

In the computation of (5), we can leverage that the term $M = (\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T$ remains fixed for all the pixels of the image. Thus, by explicitly obtaining M first, the cost of computing $\hat{\mathbf{a}}^{\mathrm{UC}}$ for all the scene pixels is basically reduced to $2mnp$ flops, since $n, p \ll m$ and, therefore, the number of arithmetic operations that are necessary to form M is negligible compared to that.

The main advantages of the unconstrained abundance estimation approach in Eq. (5) are the simplicity of its implementation and its fast execution. However, under this unconstrained model, the derivation of negative abundances is possible if the model endmembers are not pure or if they are affected by variability caused by spatial or temporal variations [9]. To address this issue, two physical constrains can be introduced into the model described in Eq. (1), these are the abundance non-negativity constraint (ANC), i.e., $a_j \geq 0$, and the abundance sum-to-one constraint (ASC), i.e., $\sum_{j=1}^p a_j = 1$ [12]. Imposing the ASC results in the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{a}\in\Delta}\left\{(\mathbf{y} - \mathbf{a}\cdot\mathbf{E})^T(\mathbf{y} - \mathbf{a}\cdot\mathbf{E})\right\}, \\ &\text{subject to: } \Delta = \left\{\mathbf{a}\,\middle|\,\sum_{j=1}^p a_j = 1\right\}. \end{aligned} \tag{6}$$

Similarly, imposing the ANC results in the optimization problem:

$$\begin{aligned} &\min_{\mathbf{a}\in\Delta}\left\{\left(\mathbf{y}-\mathbf{a}\cdot\mathbf{E}\right)^T\left(\mathbf{y}-\mathbf{a}\cdot\mathbf{E}\right)\right\},\\ &\text{subject to: } \Delta=\{\mathbf{a}\mid a_j\geq 0 \text{ for all } j\}. \end{aligned} \tag{7}$$

As indicated in [12], a fully constrained (i.e. ASC-constrained and ANC-constrained) estimate can be obtained in least-squares sense by solving the optimization problems in Eq. (6) and Eq. (7) simultaneously. While partially constrained solutions imposing only the ANC have found success in the literature [11], the ASC is however prone to criticisms because, in a real image, there is a strong signature variability [66] that, at the very least, introduces positive scaling factors varying from pixel to pixel in the signatures present in the mixtures. As a result, the signatures are defined up to a scale factor, and thus, the ASC should be replaced with a generalized ASC of the form $\sum_{j=1}^{p}\xi_j\cdot a_j=1$, in which the weights ξ_j denote the pixel-dependent scale factors [67]. What we conclude is that the non-negativity of the endmembers automatically imposes a generalized ASC. For this reason, in the following section we describe a solution that does not explicitly impose the ASC but only the ANC.

2.4.2 Non-negative constrained least squares

A NCLS algorithm can be used to obtain a solution to the ANC-constrained problem described in Equation (7) in iterative fashion [11]. A successful approach for this purpose in different applications has been the image space reconstruction algorithm (ISRA) [68], a multiplicative algorithm for solving NCLS problems. The algorithm is based on the following iterative expression:

$$\hat{\mathbf{a}}^{k+1}=\hat{\mathbf{a}}^{k}\left(\frac{\mathbf{E}^T\cdot\mathbf{y}}{\mathbf{E}^T\mathbf{E}\cdot\hat{\mathbf{a}}^{k}}\right), \tag{8}$$

where the endmember abundances at pixel $\mathbf{y}$ are iteratively estimated, so that the abundances at the k+1-th iteration, $\hat{\mathbf{a}}^{k+1}$, depend on the abundances estimated at the k-th iteration, $\hat{\mathbf{a}}^{k}$. The procedure starts with an unconstrained abundance estimation $\hat{\mathbf{a}}^{\mathrm{UC}}$ which is progressively refined in a given number of iterations. For illustrative purposes, Algorithm 3 shows the ISRA pseudocode for unmixing one hyperspectral pixel vector $\mathbf{y}$ using a set of $\mathbf{E}$ endmembers. For simplicity, in the pseudocode $\mathbf{y}$ is treated as an n-dimensional vector, and $\mathbf{E}$ is treated as a $n\times p$-dimensional matrix. The estimated abundance vector $\hat{\mathbf{a}}$ is a p-dimensional vector, and variable *iters* denotes the number of iterations per pixel in the abundance estimation process (in this work, we set iters = 200 as we have found good results empirically using this parameter setting). The pseudocode is subdivided into the numerator and denominator calculations in Equation (8). When these terms are obtained, they are divided and multiplied by the previous abundance vector. It is important to emphasize that the calculations of the fractional abundances for each pixel are independent, and therefore they can be calculated simultaneously without data dependencies, thus increasing the possibility of parallelization.

Algorithm 3 Pseudocode of ISRA algorithm for unmixing one hyperspectral pixel vector y using a set E of *p* endmembers

```
// For a certain number of iterations
for (k = 0; k < iters; k++) {
  // For all endmembers
  for (j = 0; j < p; j++)  {
    // For all bands
    for (i = 0; i < n; i++) {
      // Calculate the numerator of Equation (8)
      numerator = numerator + E[i][j]*y[i];
      // Calculate denominator of Equation (8) using â
from previous iteration (in first iteration, â = â^UC)
      for (s = 0; s < p; s++)  {
        dot += E[i][s]*â[s];
      } end for
      denominator += dot * E[i][j];
      dot = 0;
    } end for
    // Calculate the new â
    â[j] = â[j]* numerator/denominator);
    numerator = 0;
    denominator = 0;
  } end for
} for
```

The pseudocode for the ISRA algorithm reveals that this procedure is composed of very simple arithmetic operations, but also that the innermost loop, for variable s, dominates its arithmetic cost. In particular, as two arithmetic operations are performed at each iteration of this loop, this yields a total cost for the algorithm of $2np^2\cdot$ iters flops.

3 Multi-core implementations

The six numerical methods introduced in the previous section for the different stages of hyperspectral unmixing can be decomposed into a collection of basic and advanced dense linear algebra operations. Among the basic ones, we can find, e.g, vector scalings, inner (dot) products, matrix-vector products, solution of triangular systems, matrix–matrix products, etc. The advanced ones comprise the solution of linear systems of equations, matrix inversion, eigenvalue problems, and singular

values problems, among others. Fortunately, these operations are quite common to many other scientific and engineering applications, and nowadays there exist linear algebra libraries offering highly tuned and numerically reliable implementations of most of these operations for a variety of computer architectures, including multi-core processors.

In particular, Basic Linear Algebra Subprograms (BLAS) [69-71] defines the specification (the interface and functionality) of a collection of routines for basic linear algebra operations as those listed above. There is a legacy implementation of BLAS publicly available at http://www.netlib.org, but the aspect that makes BLAS really useful is the existence of implementations developed by most hardware vendors and highly tuned for their specific products. These developments include Intel MKL, AMD ACML, and IBM ESSL for their multi-core designs, but also more generic efforts like Goto-BLAS2 and ATLAS. This approach has revealed so successful that NVIDIA, manufacturer of fancier hardware architectures such as GPUs, also offers their customers its own specialized implementation, CUBLAS. For the type of architectures considered in our work, i.e. multi-core processors, an appealing property of these libraries is that they can exploit the existence of hardware concurrency, in the form of several cores, by carefully using optimized multi-threaded codes. For example, the implementation of the matrix–matrix product kernel from MKL (routine_gemm), executed on a single core of an Intel Xeon core, attains more than 90% of the peak performance of the architecture when operating on matrices of moderate to large size. If several cores are used, and the problem dimension is scaled proportionally, the routine still achieves a similar performance rate.

The contents of BLAS are structured into three separate levels—BLAS-1, BLAS-2 and BLAS-3—according to the number of flops and memory operations (memops) carried out by the kernels. Thus, routines from BLAS-1 perform a linear number of flops on a linear number of data items and, therefore, memops; an example of a BLAS-1 routine is the inner product of two vectors. For BLAS-2, both flops and memops are quadratic on the amount of data items; the classical example for this level is the matrix-vector product. Finally, for BLAS-3 the flops are cubic while the memops are quadratic; e.g., the matrix–matrix product. The type of routine (level) has important implications on performance as current architectures feature a wide difference between the floating-point performance (flops/sec.) of the processor and the memory bandwidth (memops/sec.), and this gap continues growing. Concretely, only the routines from BLAS-3 exhibit enough data reuse so as to exploit the hierarchical structure of the memory subsystem of current computers, with several layers of cache, and thus hide the large latencies that requires the access to data that lie on the main memory. Developers leverage this property by designing so-called blocked algorithms for their implementations of BLAS-3 kernels that retrieve data from the main memory to the processor by blocks (square or rectangular submatrices), and operate with them as much as possible before returning the results back to memory. This is clearly not possible for BLAS-1 and BLAS-2 as the routines in these levels exhibit a flops/memops ratio that is $O(1)$. An additional advantage of BLAS-3 over the two other levels is that, in general, the use of multiple cores in a concurrent execution, is only justified if the arithmetic cost of the operation is cubic. In consequence, when implementing the spectral unmixing methods, it will be very important to identify numerical operations that can be casted in terms of the most convenient routine from BLAS, preferably BLAS-3.

Linear Algebra PACKage (LAPACK) [72] provides advanced methods for dense linear algebra operations as those mentioned above. There is a legacy implementation

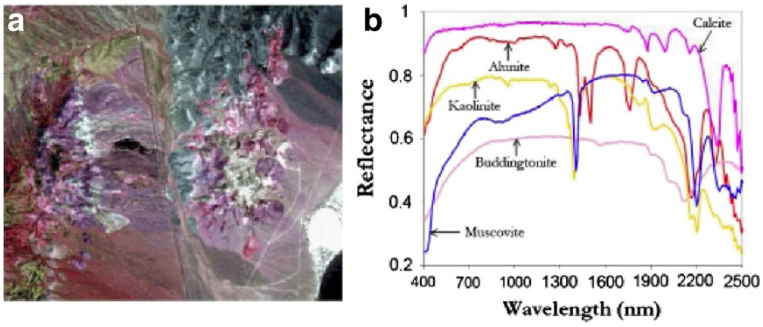


Figure 4 AVIRIS hyperspectral over the Cuprite mining district in Nevada. (a) False color composition and **(b)** U.S. Geological Survey mineral spectral signatures used for validation purposes.

Table 2 VD estimates of p for various false alarm probabilities (P_F)

P_F	Number of endmembers p
10^{-1}	37
10^{-2}	28
10^{-3}	25
10^{-4}	20
10^{-5}	19
10^{-6}	19
10^{-7}	17
10^{-8}	16

HySime provided an estimate of $p = 19$ for the Cuprite scene.

at http://wwww.netlib.org, but some hardware vendors also include tuned versions of certain routines in their mathematical libraries (e.g., Intel MKL and AMD ACML). The routines in LAPACK make a heavy use of kernels from BLAS, thus inheriting the performance (and parallelism) intrinsic to the latter. LAPACK provides specialized implementations that can leverage special matrix properties like symmetry, band structure, positive definiteness, etc., when solving linear systems or linear least squares problems as well as calculating the eigenvalues/singular values of a matrix. To improve numerical accuracy and increase performance, it is very important to select the appropriate routine from LAPACK as, in many cases, this library offers different solvers to tackle one particular problem.

Our task of developing high performance, possibly parallel, codes for the spectral unmixing methods started by (i) carefully selecting the appropriate data structures to hold the data (image and intermediate results); and (ii) developing an initial sequential implementation of the method, while simultaneously identifying basic and advanced linear algebra operations that could be performed by invoking the appropriate routines from BLAS and LAPACK. For those parts of the method that could be performed using kernels from these libraries, the implementation/optimization task was over.

However, during the implementation of the methods, we detected certain parts of the methods that had to be manually encoded, usually in the form of (nested) loops. For many of these code fragments, special care was taken to apply basic optimizations such as avoiding expensive operations, eliminating common subexpressions, avoiding branches, selecting the appropriate order in nested loops, using the appropriate compiler optimizations, etc. After this stage, the execution of the code was profiled to identify possible performance bottlenecks. For those fragments of code, in particular loops, that exhibited a significant cost (execution time), we analyzed the possibility of reducing their impact by leveraging loop concurrency via OpenMP [73]. This is a standard parallelization tool that is available in most current compilers (e.g., Intel `icc`, GNU `gcc`) and allows an easy and, in most cases, efficient parallelization of C/Fortran codes on multi-core processors.

Each one of these implementation and optimization stages was carefully monitored from the point of view of correctness, experimental accuracy, and performance. The result of this process was the spectral unmixing routines that we evaluate in the next section.

4 Experimental results

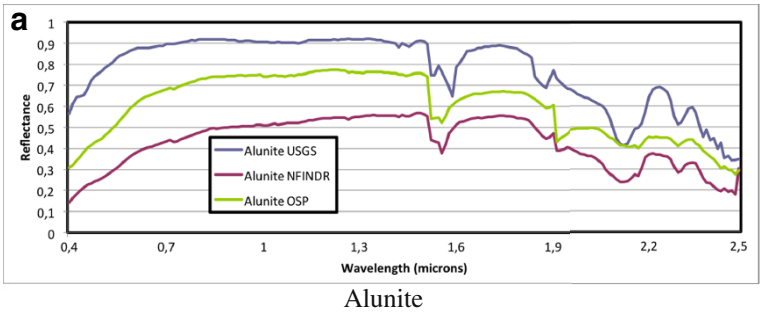
This section is organized as follows. Section 4.1 describes the hyperspectral data set used in experiments. In Section 4.2 we describe the multi-core processing platforms. Finally, Section 4.3 performs a detailed assessment of the performance versus energy consumption of the considered multi-core architectures when executing the different unmixing chains that can be formed with the processing modules described in Section 2.

4.1 Hyperspectral data

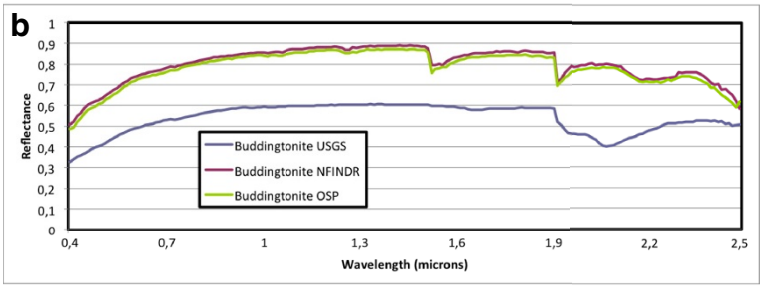
The hyperspectral data set used in experiments was collected by the AVIRIS sensor over the Cuprite mining district in Nevada in the summer of 1997 (see Figure 4). It is available online (in reflectance units) after atmospheric correction [74]. The portion used in experiments corresponds to a 350 × 350-pixel subset of the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data, which comprise 188 spectral bands in the range from 400 to 2,500 nm and a total size of around 50 MB. Water absorption bands as well as bands with low signal-to-noise ratio (SNR) were removed prior to the analysis. 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 USGS library [75] have been used in the literature for evaluation purposes [16].

Table 3 Spectral angle values (in degrees) between the target pixels extracted by the OSP-GS algorithm and the reference USGS mineral signatures for the AVIRIS Cuprite scene

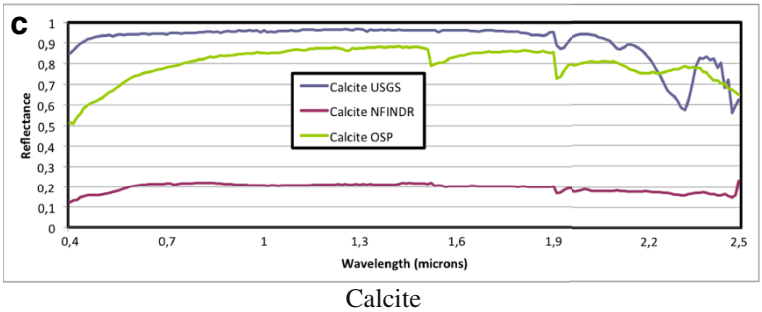
Algorithm	Alunite (°)	Buddingtonite (°)	Calcite (°)	Kaolinite (°)	Muscovite (°)	Average (°)
OSP-GS	5.48	4.08	5.87	11.14	5.68	6.45
N-FINDR	4.81	4.29	7.60	9.92	5.05	6.33



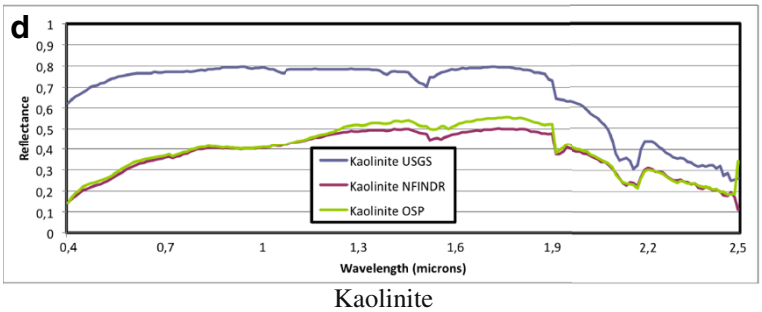
Alunite



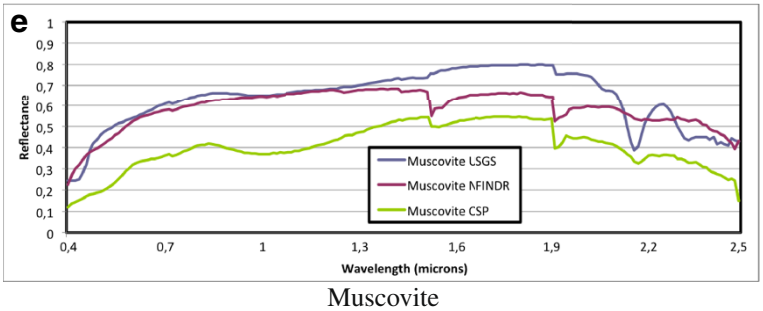
Muscovite



Calcite



Kaolinite



Muscovite

Figure 5 Estimated versus ground-truth endmembers for minerals: (a) alunite, (b) buddingtonite, (c) calcite, (d) kaolinite and (e) muscovite.

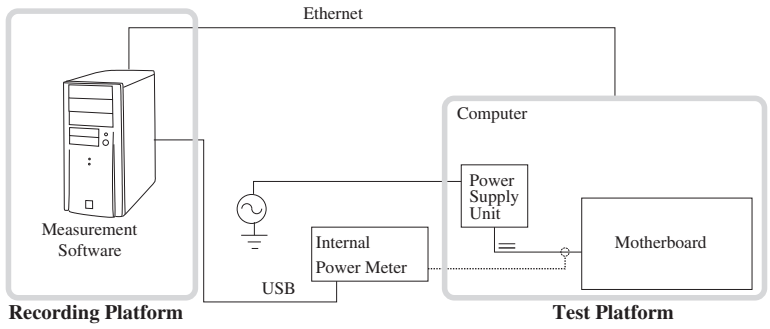


Figure 6 Experimental setup for power evaluation.

For illustrative purposes, Table 2 provides the values of p (number of endmembers) estimated by the VD method for the considered hyperspectral scene, using different values of the false alarm probability (P_F). The number of endmembers estimated by HySime was $p = 19$ for the Cuprite scene. As shown by Table 2, a consensus between VD and HySime was observed for $P_F = 10^{-5}$ and $P_F = 10^{-6}$. On the other hand, Table 3 shows the spectral angles (in degrees) between the most similar endmembers extracted by the OSP-GS and the reference USGS spectral signatures available for this scene. The range of values for the spectral angle is [0°, 90°], with values close to 0° indicating higher spectral similarity. As shown by Table 3, the endmembers extracted by both the OSP-GS and N-FINDR algorithms are very similar, spectrally, to the USGS reference signatures, despite the potential variations (due to posible interferers still remaining after the atmospheric correction process) between the ground signatures and the airbone data. For illustrative purposes, Figure 5 plots the estimated endmembers against the ground-truth spectra for the considered endmember extraction algorithms.

4.2 Multi-core platforms

In 2004, the evolution of processor architecture shifted from a progressive increment of clock frequency to a growth in the number of cores. Thus, although current processors still feature a moderate number of cores (between 4 and 16), the trend indicates that next generations will include a larger number of cores. On the other hand, as-of-today it is possible to build a commodity shared-memory multiprocessor with four sockets that can accommodate 16-core processors each, for a total of 64 cores in a single desktop platform.

Following this trend towards high levels of hardware concurrency at the core-level, all the experiments were conducted on a platform equipped with 4 AMD Opteron 6172 processors, with 12 cores per processor, and a total of 48 cores in the platform. The software employed in the experiments included Intel MKL v10.3 implementation of the BLAS and LAPACK libraries, and Intel `icc` v12.1.3 compiler. The codes were compiled with the optimization flag `-O3`, and single-precision arithmetic was employed in all experiments. The explosion of hardware concurrency

Table 4 Processing time (seconds), energy consumption (Watts-hour), and maximum power dissipation (Watts) of different full spectral unmixing chains applied to the AVIRIS Cuprite scene

Number of endmembers	Endmember identification	Abundance estimation	Time	Energy	Maximum power
VD (20)	OSP-GS (34)	ULS (20)	0.413	0.0468	492
VD (20)	OSP-GS (34)	ISRA (48)	2.957	0.4297	543
VD (20)	N-FINDR (20)	ULS (20)	2.332	0.2187	489
VD (20)	N-FINDR (20)	ISRA (48)	4.899	0.6093	563
HySime (48)	OSP-GS (34)	ULS (20)	15.508	2.1093	540
HySime (48)	OSP-GS (34)	ISRA (48)	16.762	2.2812	597
HySime (48)	N-FINDR (20)	ULS (20)	16.971	2.2343	540
HySime (48)	N-FINDR (20)	ISRA (48)	18.077	2.4140	571

The values inside the parentheses are the number of cores used in the implementation of each method.

Table 5 Processing time (seconds) by each different processing module for the AVIRIS Cuprite scene in a multi-core system

Number of endmembers		Endmember identification		Abundance estimation	
VD	HySime	OSP-GS	N-FINDR	ULS	ISRA
0.15	15.2	0.25	2.17	0.02	2.55

in multi-core processors requires the development of efficient parallel software that attains a significant fraction of the platform peak performance. However, in some cases the target method does not exhibit enough concurrency to efficiently exploit all the computational units in the platform. Therefore, when executing this kind of methods, the use of all the cores results in an increment of the execution time (due to the overhead introduced by the synchronization, communication and management of the cores) and the corresponding waste of energy. In this context, it is preferable to limit the number of cores employed, idling the rest of them so that the OS can move these cores into an energy-saving state (*C*-state) that yields a significant reduction of dynamic power.

In our study of the spectral unmixing methods, each code was evaluated separately to determine the optimal number of cores from the point of view of execution time. Given that the energy consumption equals the product of power dissipation times execution time, in general, reducing the execution time is an important step towards increasing energy efficiency.

In our experiments, the power consumption was measured using an internal DC powermeter. This device obtains 25 samples per second and is attached to the 12 V lines connecting the power suply with the motherboard (chipset plus processors) of the platform (see Figure 6). With this configuration, the results are not affected by inefficiencies of the power supply unit, or the "noise" due to the operation of other hardware components like fans, disks, etc. Also, samples from the powermeter are collected in a separate system, to prevent the measurements from impairing the accuracy of the tests. The error of the device is less than 5%. The powermeter is controlled via our library `pmlib`, which requires minimum changes to the code.

4.3 Assessment of performance versus energy consumption

In this section, we analyze the processing times, energy consumption and maximum power obtained by different combinations of the modules reported in Section 2 for estimating the number of endmembers (VD and HySime), identifying the endmember signatures (OSP-GS and *N*-FINDR), and estimating the abundances (ULS and ISRA). In all cases, the number of endmembers to be extracted was set to $p = 19$. Table 4 shows the processing times, energy consuption and maximum power for different full unmixing chains formed using combinations of the aforementioned modules. In Table 4, we also indicate (in the parentheses) the number of cores from the AMD Opteron 6172 system that were used in the multi-core implementation of each method. It is important to emphasize that, in order to satisfy the real-time processing constraint for the AVIRIS Cuprite scene, we should be able to process it in less than 1.98 s which is the time needed by the instrument to collect the data. As shown in Table 4, only the chain VD+OSP-GS+LSU achieved real-time performance, with two other chains (VD + OSP−GS + ISRA and VD + *N*-FINDR + ISRA) providing near real-time performance in the target multi-core architecture. It is remarkable that the inclusion of HySime for the identification of the number of endmembers significantly increments the processing times and also increases the energy consumption.

In order to investigate the individual contributions of the methods in different parts of the unmixing chain to the total processing time, Table 5 reports the processing times measured for all the individual methods when processing the AVIRIS Cuprite scene. It can be observed that, by far, HySime is the most computationally expensive method while VD provides an alternative for this part of the chain which is about 100× faster in comparison. Table 5 also reveals that the OSP-GS method used for the endmember identification is about 9× faster than N-FINDR in the multi-core. Finally, ISRA is more than 100× slower than ULS as a consequence of the fact that it imposes the non negativity constraint in the abundance estimation. The results in Table 5 suggest that the combination VD+OSP-GS+ULS provides a good basis for fast spectral unmixing in the considered multi-core platform (in fact, this combination is the only one that results in real-time performance in our experiments).

For comparative purposes, Table 6 shows the processing times measured for the same methods reported in

Table 6 Processing time (seconds) by each different processing module for the AVIRIS Cuprite scene in a GPU system

Number of endmembers		Endmember identification		Abundance estimation	
VD	HySime	OSP-GS	N-FINDR	ULS	ISRA
0.405	1.655	0.024	0.069	0.038	0.858

Table 5 implemented in the NVidia^{TM} GeForce GTX 580 GPU [76], which features 512 processor cores operating at 1.544 GHz, with a total dedicated memory of 1,536 MB, at 2.004 MHz (with 384-bit GDDR5 interface) and memory bandwidth of 192.4 GB/s. The GPU is connected to an Intel core i7 920 CPU at 2.67 GHz with eight cores, which uses a motherboard Asus P6T7 WS SuperComputer. As shown in Table 6, the processing times in the GPU are comparatively similar to those reported for the multi-core for the OSP-GS and ULS algorithms. On the other hand, the times for HySime, *N*-FINDR and ISRA are sensibly lower in the GPU, with only the VD being slightly faster in the multi-core than in the GPU. However, the energy consumption is much higher in the GPU, which still makes the multi-core platform a more interesting platform from an operational point of view.

5 Conclusions

In this article, we have addressed hyperspectral imaging via spectral unmixing on multi-core processors, exposing a detailed evaluation of the performance and energy requirements of efficient parallel codes for all the stages in the spectral unmixing chain on multi-core processors. Specifically, we have implemented modules for (i) the estimation of the number of endmembers, (ii) the identification of a collection of these, and (iii) the estimation of the fractional abundances, using kernels from highly tuned linear algebra libraries and OpenMP directives on a platform equipped with 48 AMD cores.

Our study offers two major conclusions:

- Three of the unmixing chains attain real-time performance (VD + OSP − GS + ULS) or close to it (VD + OSP − GS + ISRA and VD + *N*-FINDR + ULS) as the underlying modules VD, OSP-GS, *N*-FINDR, LSU and ISRA exhibit a high degree algorithmic concurrency that can be leveraged to yield efficient parallel implementations on current multi-core processors. On the other hand, HySime presents a performance bottleneck that turns those chains that utilize this module inappropriate for real-time image processing.
- With the expected increase in the number of cores in future architectures, shared-memory platforms equipped with a few multi-core processors are a competitive approach to efficiently tackle computationally expensive hyperspectral imaging applications on cheap commodity hardware. Compared with FPGAs, conventional multi-core processors offer the plain advantage of being much easier to program, considerably improving the software development cycle. Furthermore, general-purpose cores offer an appealing performance-energy ratio and they clearly outperform GPUs in their tolerance to incorporate radiation-avoidance mechanisms.

As future work, we plan to analyze in more detail the energy consumption of other types of architectures, including FPGAs or GPUs, which are currently considered as candidate specialized hardware platforms for onboard hyperspectral image processing.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

Enrique S Quintana-Ortí and Alfredo Remón were supported by the CICYT project TIN2011-23283 of the *Ministerio de Economía y Competitividad* and FEDER. Funding from the Spanish Ministry of Science and Innovation (CEOS-SPAIN project, reference AYA2011-29334-C02-02) is also gratefully acknowledged.

Author details

[1]Department of Engineering and Computer Sciences, University Jaime I, Castellon, Spain. [2]Hyperspectral Computing Laboratory (HyperComp), Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain.

Received: 15 November 2012 Accepted: 6 February 2013
Published: 2 April 2013

References

1. AFH Goetz, G Vane, JE Solomon, BN Rock, Imaging spectrometry for earth remote sensing. Science. **228**, 1147–1153 (1985)
2. RO Green, ML Eastwood, CM Sarture, TG Chrien, M Aronsson, BJ Chippendale, JA Faust, BE Pavri, CJ Chovit, M Solis, *et al*, Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS). Remote Sens. Envir. **65**(3), 227–248 (1998)
3. A Plaza, J Plaza, A Paz, S Sanchez, Parallel hyperspectral image and signal processing. IEEE Signal Process. Mag. **28**(3), 119–126 (2011)
4. A Plaza, JA Benediktsson, J Boardman, J Brazile, L Bruzzone, G Camps-Valls, J Chanussot, M Fauvel, P Gamba, J Gualtieri, M Marconcini, TC Tilton, G Trianni, Recent advances in techniques for hyperspectral image processing. Remote Sens. Envir. **113**, 110–122 (2009)
5. A Plaza, Q Du, JM Bioucas-Dias, X Jia, F Kruse, Foreword to the special issue on spectral unmixing of remotely sensed data. IEEE Trans. Geosci. Remote Sens. **49**(11), 4103–4110 (2011)
6. PE Johnson, MO Smith, S Taylor-George, JB Adams, A semiempirical method for analysis of the reflectance spectra for binary mineral mixtures. J. Geophys. Res. **88**, 3557–3561 (1983)
7. JB Adams, MO Smith, PE Johnson, Spectral mixture modeling: a new analysis of rock and soil types at the Viking Lander 1 site. J. Geophys. Res. **91**, 8098–8112 (1986)
8. N Keshava, JF Mustard, Spectral unmixing. IEEE Signal Process. Mag. **19**, 44–57 (2002)
9. A Plaza, P Martinez, R Perez, J Plaza, A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. IEEE Trans. Geosci. Remote Sens. **42**(3), 650–663 (2004)
10. Q Du, N Raksuntorn, NH Younan, RL King, End-member extraction for hyperspectral image analysis. Appl. Opt. **47**, 77–84 (2008)
11. CI Chang, D Heinz, Constrained subpixel target detection for remotely sensed imagery. IEEE Trans Geosci. Remote Sens. **38**, 1144–1159 (2000)
12. D Heinz, CI Chang, Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery. IEEE Trans. Geosci. Remote Sens. **39**, 529–545 (2001)
13. CC Borel, SAW Gerstl, Nonlinear spectral mixing model for vegetative and soil surfaces. Remote Sens. Envir. **47**(3), 403–416 (1994)
14. W Liu, EY Wu, Comparison of non-linear mixture models. Remote Sens. Envir. **18**, 1976–2003 (2004)
15. N Raksuntorn, Q Du, Nonlinear spectral mixture analysis for hyperspectral imagery in an unknown environment. IEEE Geosci. Remote Sens. Lett. **7**(4), 836–840 (2010)

16. A Plaza, G Martin, J Plaza, M Zortea, S Sanchez, in *Optical Remote Sensing*, ed. by S Prasad, LM Bruce, and J Chanussot. Recent developments in spectral unmixing and endmember extraction (Springer, Berlin, Germany, 2011), pp. 235–267
17. JJ Settle, NA Drake, Linear mixing and the estimation of ground cover proportions. Int. J. Remote Sens. **14**, 1159–1177 (1993)
18. JC Harsanyi, CI Chang, Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection. IEEE Trans. Geosci. Remote Sens. **32**(4), 779–785 (1994)
19. JW Boardman, FA Kruse, RO Green, in *Proc. JPL Airborne Earth Science Workshop*. Mapping target signatures via partial unmixing of AVIRIS data (Pasadena, USA, 1995), pp. 23–26
20. JH Bowles, PJ Palmadesso, JA Antoniades, MM Baumback, LJ Rickard, Use of filter vectors in hyperspectral data analysis. Proc. SPIE Infrared Spaceborne Remote Sens. III. **2553**, 148–157 (1995)
21. RA Neville, K Staenz, T Szeredi, J Lefebvre, P Hauff, in *Proc. 21st Canadian Symp. Remote Sens*. Automatic endmember extraction from hyperspectral data for mineral exploration (Ottawa, Canada, 1999), pp. 21–24
22. A Ifarraguerri, CI Chang, Multispectral and hyperspectral image analysis with convex cones. IEEE Trans. Geosci. Remote Sens. **37**(2), 756–770 (1999)
23. ME Winter, N-FINDR: An algorithm for fast autonomous spectral endmember determination in hyperspectral data. Proc. SPIE. **3753**, 266–277 (1999)
24. Q Du, H Ren, CI Chang, A comparative study for orthogonal subspace projection and constrained energy minimization. IEEE Trans. Geosci. Remote Sens. **41**(6), 1525–1529 (2003)
25. M Berman, H Kiiveri, R Lagerstrom, A Ernst, R Dunne, JF Huntington, ICE: a statistical approach to identifying endmembers in hyperspectral images. IEEE Trans. Geosci. Remote Sens. **42**(10), 2085–2095 (2004)
26. JMP Nascimento, JM Bioucas-Dias, Vertex component analysis: a fast algorithm to unmix hyperspectral data. IEEE Trans. Geosci. Remote Sens. **43**(4), 898–910 (2005)
27. A Plaza, CI Chang, Impact of initialization on design of endmember extraction algorithms. IEEE Trans. Geosci. Remote Sens. **44**(11), 3397–3407 (2006)
28. CI Chang, A Plaza, A fast iterative algorithm for implementation of pixel purity index. IEEE Geoscience. Remote Sens. Lett. **3**, 63–67 (2006)
29. DM Rogge, B Rivard, J Zhang, J Feng, Iterative spectral unmixing for optimizing per-pixel endmember sets. IEEE Trans. Geosci. Remote Sens. **44**(12), 3725–3736 (2006)
30. J Wang, CI Chang, Applications of independent component analysis in endmember extraction and abundance quantification for hyperspectral imagery. IEEE Trans. Geosci. Remote Sens. **44**(9), 2601–2616 (2006)
31. CI Chang, CC Wu, W Liu, YC Ouyang, A new growing method for simplex-based endmember extraction algorithm. IEEE Trans. Geosci. Remote Sens. **44**(10), 2804–2819 (2006)
32. A Zare, P Gader, Sparsity promoting iterated constrained endmember detection for hyperspectral imagery. IEEE Geosci. Remote Sens. Lett. **4**(3), 446–450 (2007)
33. A Zare, P Gader, Hyperspectral band selection and endmember detection using sparsity promoting priors. IEEE. Remote Sens. Lett. **5**(2), 256–260 (2008)
34. M Zortea, A Plaza, A quantitative and comparative analysis of different implementations of *N*-FINDR: a fast endmember extraction algorithm. IEEE Geosci. Remote Sens. Lett. **6**, 787–791 (2009)
35. X Tao, B Wang, L Zhang, Orthogonal bases approach for the decomposition of mixed pixels in hyperspectral imagery. IEEE Geosci. Remote Sens. Lette. **6**, 219–223 (2009)
36. A Zare, P Gader, PCE: piecewise convex endmember detection. IEEE Trans. Geosci. Remote Sens. **48**(6), 2620–2632 (2010)
37. CI Chang, CC Wu, CS Lo, ML Chang, Real-time simplex growing algorithms for hyperspectral endmember extraction. IEEE Trans. Geosci. Remote Sens. **48**(4), 1834–1850 (2010)
38. F Schmidt, A Schmidt, E Treandguier, M Guiheneuf, S Moussaoui, N Dobigeon, Implementation strategies for hyperspectral unmixing using Bayesian source separation. IEEE Trans. Geosci. Remote Sens. **48**(11), 4003–4013 (2010)
39. O Duran, M Petrou, Robust endmember extraction in the presence of anomalies. IEEE Trans. Geosci. Remote Sens. **49**(6), 1986–1996 (2011)
40. B Zhang, X Sun, L Gao, L Yang, Endmember extraction of hyperspectral remote sensing images based on the ant colony optimization (ACO) algorithm. IEEE Trans. Geosci. Remote Sens. **49**(7), 2635–2646 (2011)
41. M Shoshany, F Kizel, N Netanyahu, N Goldshlager, T Jarmer, G Even-Tzur, An iterative search in end-member fraction space for spectral unmixing. IEEE Geosci. Remote Sens. Lett. **8**(4), 706–709 (2011)
42. CI Chang, CC Wu, HM Chen, Random pixel purity index. IEEE Geosci. Remote Sens. Lett. **7**(2), 324–328 (2010)
43. S Dowler, M Andrews, On the convergence of *N*-FINDR and related algorithms: to iterate or not to iterate. IEEE Geosci. Remote Sens. Lett. **8**, 4–8 (2011)
44. A Plaza, D Valencia, J Plaza, P Martinez, Commodity cluster-based parallel processing of hyperspectral Imagery. J. Parall. Distr. Comput. **66**(3), 345–358 (2006)
45. A Plaza, CI Chang, *High Performance Computing in Remote Sensing*. (Taylor & Francis, Boca Raton, FL, 2007)
46. A Plaza, Q Du, Y-L Chang, RL King, High performance computing for hyperspectral remote sensing. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 528–544 (2011)
47. CA Lee, SD Gasster, A Plaza, CI Chang, B Huang, Recent developments in high performance computing for remote sensing: a review. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 508–527 (2011)
48. A Plaza, CI Chang, Clusters versus FPGA for parallel processing of hyperspectral imagery. Int. J. High Perfor. Comput. Appl. **22**(4), 366–385 (2008)
49. C Gonzalez, J Resano, D Mozos, A Plaza, D Valencia, FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis. EURASIP J. Adv. Signal Process. **969806**, 1–13 (2010)
50. C Gonzalez, D Mozos, J Resano, A Plaza, FPGA implementation of the *N*-FINDR algorithm for remotely sensed hyperspectral image analysis. IEEE Trans. Geosci. Remote Sens. **50**(2), 374–388 (2012)
51. J Setoain, M Prieto, C Tenllado, F Tirado, GPU for parallel on-board hyperspectral image processing. Int. J. High Perfor. Comput. Appl. **22**(4), 424–437 (2008)
52. M Hsueh, CI Chang, Field programmable gate arrays (FPGA) for pixel purity index using blocks of skewers for endmember extraction in hyperspectral imagery. Int. J. High Perfor. Comput. Appl. **22**, 408–423 (2008)
53. Y Tarabalka, TV Haavardsholm, I Kasen, T Skauli, Real-time anomaly detection in hyperspectral images using multivariate normal mixture models and GPU processing. J. Real-Time Image Process. **4**, 1–14 (2009)
54. H Yang, Q Du, G Chen, Unsupervised hyperspectral band selection using graphics processing units. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 660–668 (2011)
55. JA Goodman, D Kaeli, D Schaa, Accelerating an imaging spectroscopy algorithm for submerged marine environments using graphics processing units. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 669–676 (2011)
56. J Mielikainen, B Huang, A Huang, GPU-accelerated multi-profile radiative transfer model for the infrared atmospheric sounding interferometer. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 691–700 (2011)
57. E Christophe, J Michel, J Inglada, Remote sensing processing: from multicore to GPU. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **4**(3), 643–652 (2011)
58. S Sanchez, A Paz, G Martin, A Plaza, Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units. Concur. Comput. Pract. Exp. **23**(13), 1538–1557 (2011)
59. CI Chang, Q Du, Estimation of number of spectrally distinct signal sources in hyperspectral imagery. IEEE Trans. Geosci. Remote Sens. **42**(3), 608–619 (2004)
60. JM Bioucas-Dias, JMP Nascimento, Hyperspectral subspace identification. IEEE Trans. Geosci. Remote Sens. **46**(8), 2435–2445 (2008)
61. S Lopez, P Horstrand, GM Callico, JF Lopez, R Sarmiento, A low-computational-complexity algorithm for hyperspectral endmember extraction: modified vertex component analysis. IEEE Geosci. Remote Sens. Lett. **9**(3), 502–506 (2012)
62. S Bernabé, S López, A Plaza, R Sarmiento, GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis. IEEE Geosci. Remote Sens. Lett. **10**(2), 221–225 (2013). doi:10.1109/LGRS.2012.2198790
63. A Remón, S Sanchez, A Paz, ES Quintana-Ortí, A Plaza, Real-time endmember extraction on multi-core processors. IEEE Geosci. Remote Sens. Lett. **8**(5), 924–928 (2011)

64. JA Richards, X Jia, *Remote Sensing Digital Image Analysis: An Introduction*. (Springer-Verlag, New York, 2006)
65. CI Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. (Kluwer Academic/Plenum Publishers, New York, 2003)
66. CA Bateson, GP Asner, CA Wessman, Endmember bundles: a new approach to incorporating endmember variability into spectral mixture analysis. IEEE Trans. Geosci. Remote Sens. **38**(2), 1083–1094 (2000)
67. MD Iordache, J Bioucas-Dias, A Plaza, Sparse unmixing of hyperspectral data. IEEE Trans. Geosci. Remote Sens. **49**(6), 2014–2039 (2011)
68. ME Daube-Witherspoon, G Muehllehner, An iterative image space reconstruction algorithm suitable for volume ECT. IEEE Trans. Med. Imag. **5**, 61–66 (1986)
69. CL Lawson, RJ Hanson, DR Kincaid, FT Krogh, Basic linear algebra subprograms for Fortran usage. ACM Trans. Math. Soft. **5**(3), 308–323 (1979)
70. JJ Dongarra, J Du Croz, S Hammarling, RJ Hanson, An extended set of FORTRAN basic linear algebra subprograms. ACM Trans. Math. Soft. **14**, 1–17 (1988)
71. JJ Dongarra, J Du Croz, S Hammarling, I Duff, A set of level 3 basic linear algebra subprograms. ACM Trans. Math. Soft. **16**, 1–17 (1990)
72. E Anderson, Z Bai, C Bischof, LS Blackford, J Demmel, JJ Dongarra, JD Croz, S Hammarling, A Greenbaum, A McKenney, D Sorensen, *LAPACK Users' guide*, 3rd edn. (SIAM, Philadelphia, 1999)
73. The OpenMP API specification for parallel programming. [http://www.openmp.org/]
74. NASA, Aviris (Airbone Visible infrared Imaging Spectrometer) - Data. aviris.jpl.nasa.gov/data/free_data.html
75. U.S. Geological Survey (U.S. Dep. of the Interior), USGS Spectroscopy Lab - Spectral Library. speclab.cr.usgs.gov/spectral-lib.html
76. NVIDIA Corporation, NVIDIA GTX580 specifications. [www.nvidia.com/object/product-geforce-gtx-580-us.html]

doi:10.1186/1687-6180-2013-68
Cite this article as: Remón *et al.*: **Performance versus energy consumption of hyperspectral unmixing algorithms on multi-core platforms.** *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:68.