GPU Implementation of Iterative-Constrained Endmember Extraction from Remotely Sensed Hyperspectral Images

Eysteinn Már Sigurdsson, Antonio Plaza, *Fellow, IEEE*, and Jón Atli Benediktsson, *Fellow, IEEE*

***Abstract*—Hyperspectral unmixing is an important technique for remotely sensed hyperspectral data exploitation. Linear spectral unmixing is frequently used to characterize mixed pixels in hyperspectral data. Over the last few years, many techniques have been proposed for identifying pure spectral signatures (endmembers) in hyperspectral images. The iterated constrained endmembers (ICE) algorithm is an iterative method that uses the linear model to extract endmembers and abundances simultaneously from the data set. This approach does not necessarily require the presence of pixels in the hyperspectral image as it can automatically derive the signatures of endmembers even if these signatures are not present in the data. As it is the case with other endmember identification algorithms, ICE suffers from high computational complexity. In this paper, a complete and scalable adaptation of the ICE algorithm is implemented using the parallel nature of commodity graphics processing units (GPUs). This gives significant speed increase over the traditional ICE method and allows for processing of larger data set with an increased number of endmembers.**

***Index Terms*—Graphics processing units (GPUs), hyperspectral imaging, iterative constrained endmembers (ICEs), spectral unmixing.**

I. Introduction

A. Hyperspectral Unmixing

HYPERSPECTRAL unmixing is a very important technique for remotely sensed hyperspectral data exploitation [1]. This is because the signal recorded by a hyperspectral sensor at a given band and from a given pixel, letting alone the effects of the atmosphere, is a mixture of the "light" scattered by the constituent substances located in the respective pixel coverage. With the ultimate goal of recovering the ability to discriminate materials, a significant amount of research work has been devoted to hyperspectral unmixing in recent years (see, e.g., [1], [2], and references therein).

Manuscript received December 04, 2014; revised April 15, 2015; accepted May 25, 2015. Date of publication June 17, 2015; date of current version July 30, 2015. This work was supported by the EU FP7 Theme Space Project North State.

E. M. Sigurdsson is with the Faculty of Electrical and Computer Engineering, University of Iceland, Reykjavik IS-107, Iceland (e-mail: eysteinn@hi.is).

A. Plaza is with the Department of Technology of Computers and Communications, University of Extremadura, Caceres E-10071, Spain (e-mail: aplaza@unex.es).

J. A. Benediktsson is with the Faculty of Electrical and Computer Engineering, University of Iceland, Reykjavik IS-107, Iceland (e-mail: benedikt@hi.is).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2015.2441699

The linear mixing model holds true when the mixing scale is macroscopic and the incident light interacts with just one material, as is the case in checkerboard type scenes [3]. Despite its simplicity, the linear model is an acceptable approximation for the light scattering in many real scenarios. In linear spectral unmixing, each mixed pixel is characterized as a linear combination of pure spectral signatures (called endmembers [4]), weighted by their corresponding abundance fractions in the pixel [5]. Despite the simple interpretation of the linear model, researchers are beginning to move into the nonlinear mixing field to cope with the limitations of the linear model [6]. Unmixing via sparse regression is still another direction recently introduced to circumvent a part of the limitations of the linear model. This line of research formulates hyperspectral unmixing as a semisupervised approach in which the endmember identification is replaced by a sparse regression over a library of ideal spectral signatures, usually overcomplete, and generally obtained in laboratory [7]. This approach avoids the identification of the endmembers directly from the image scene, which is generally a difficult step, as well as the estimation of their number [8]. However, in the cases, in which a spectral library is not available, techniques for endmember identification from the image scene are required. Available algorithms can broadly be classified either as pure pixel or nonpure pixel based.

In the pure pixel-based algorithms, the presence in the data of at least one pure pixel per endmember is assumed, meaning that there is at least one spectral vector on each vertex of the data simplex. This class of algorithms has been the most often used in linear spectral unmixing applications, perhaps because of the light computational burden and clear conceptual meaning. Most of the pure pixel-based algorithms exploit one of the following properties of the endmember signatures. 1) The extremes of the projection of the spectral vectors onto any subspace correspond to endmembers. 2) The volume defined by any set of spectral vectors is maximum when those are endmembers. Representative algorithms of class 1 are *pixel purity index* (PPI) [9], *vertex component analysis* (VCA) [10], simplex growing algorithm (SGA) [11], *successive volume maximization* (SVMAX) [12], and the *recursive algorithm for separable NMF* (RSSNMF) [13]. Representative algorithms of class 2 are N-FINDR [14], *iterative error analysis* (IEA), [15], *sequential maximum angle convex cone* (SMACC), and *alternating volume maximization*(AVMAX) [12], and among many others [1], [2].

For nonpure pixel-based algorithms, the endmembers are generally inferred by fitting a minimum volume simplex to

the data. This rather simple and yet powerful idea, introduced by Craig in his seminal work [16], underlies several geometrical-based unmixing algorithms. This is the case for *iterative-constrained endmembers* (ICE) algorithm [17] and of the *minimum volume-constrained-nonnegative matrix factorization* (MVC-NMF) [18], whose main differences are related with the way they define the volume regularizer term. For variations of these ideas recently introduced, see [1] and [2]. The *sparsity-promoting ICE* (SPICE) [19] is an extension of the ICE algorithm that incorporates sparsity-promoting priors aiming at finding the number of endmembers. A main problem of this kind of algorithms is their high computational complexity, since the optimization problem that they address is nonconvex. The *simplex identification via variable splitting and augmented Lagrangian* (SISAL) [20] and the *minimum volume enclosing simplex* (MVES) [21] address this complexity by introducing efficient solvers. In this work, we develop an optimized version of ICE, which exploits the computational power offered by commodity graphics processing units (GPUs). In the following, we provide an overview of recent developments in the exploitation of GPU technology for spectral unmixing applications, as well as a description of the organization of the remainder of the paper and the notations adopted.

B. GPUs for Hyperspectral Image Processing

In recent years, GPUs have evolved into highly parallel, multithreaded, many-core coprocessors with tremendous computational power, and memory bandwidth [22]. The combined features of general-purpose supercomputing, high parallelism, high memory bandwidth, low cost, compact size, and straightforward programmability are now making GPU-based desktop computers an appealing alternative to a massively parallel systems made up of commodity CPUs. The exploding GPU capability has increasingly attracted scientists and engineers to use it as a cost-effective high-performance computing platform, including scientists in hyperspectral imaging areas. With the advent of tools such as compute device unified architecture (CUDA),[1] intended for general-purpose programming of NVidia cards, and OpenCL,[2] which can be used to program GPU cards from any vendor, the number of applications implemented on GPUs has boosted over recent years.

Several efforts exploiting GPU technology can already be found in the hyperspectral imaging literature [23], with many popular algorithms adapted to the GPU (PPI [24] and N-FINDR [25]). For instance, in the area of spectral unmixing, there have already been many developments. A seminal effort was the GPU-based implementation of the automated morphological endmember extraction (AMEE) algorithm [26] for endmember identification, described in [27]. In that case, speedups of the GPU implementation on the order of $15\times$ compared to a single core CPU version were reported. A full spectral unmixing chain comprising the automatic estimation of the number of endmembers, the identification of the endmember signatures, and quantification of endmember fractional abundances has been reported in [28] with speedups superior to $50\times$. Additional efforts toward real-time implementations of spectral unmixing chains have also been developed [29]. It should be noted that, despite the increasing programmability of low-power GPUs such as those available in smartphones, radiation-tolerance and power consumption issues still prevent the full incorporation of GPUs to spaceborne Earth observation missions [30].

[1][Online]. Available: http://www.nvidia.com/object/cuda home new.html, accessed on April 7, 2015.

[2][Online]. Available: https://www.khronos.org/opencl, accessed on April 7, 2015.

C. Paper Organization

This paper is organized as follows. Section II explains the ICE algorithm. In Section III, the OpenCL architecture is explained and the details of the GPU implementation are presented. The performance is discussed in Section IV both in terms of accuracy and processing time. Conclusions are drawn in Section V.

D. Notation

Matrices are denoted by capital letters (A) where a matrix with character in subscript (A_j) indicates a single column of that matrix. 1_p represents a $p \times 1$ vector of ones. Lower case letters can be a vector or a single number depending on their description. In the pseudocode, MATLAB notation is sometimes used with matrices, such as A[i,j]. The *trace* and *covariance* of a matrix are, respectively, denoted as *tr(A)* and ΣA. Finally, $\|A\|_F$ is the Frobenius norm of the matrix A.

II. Iterated-Constrained Endmembers

In this section, the ICE algorithm is explained. This endmember estimation method is an iterative cyclic descent algorithm that converges to an optimal solution for the endmembers. ICE is based on the convex geometry model, which assumes that every pixel is a linear combination of endmembers. This can be described by a mathematical formulation for each pixel Y_i

$$Y_i = \sum_{k=1}^{m} p_{ik} M_k + \epsilon, \quad i = 1, \ldots, n \tag{1}$$

where n is the number of pixels, m is the number of endmembers, M_k is the kth endmember, p_{ik} defines how much the endmember contributes proportionally to the pixel (abundance), and ϵ is the noise term with Gaussian distribution. The proportions of the endmembers can never be negative and the total sum of proportions has to add up to one. As a result, the mixing proportions need to satisfy the constraints

$$p_{ik} \geq 0, \quad k = 1, \ldots, m, \quad \sum_{k=1}^{m} p_{ik} = 1. \tag{2}$$

This indicates that the data lie inside a simplex in b-dimensional space and the endmembers form the vertices of the simplex. The dimensionality of this simplex has to follow:

$$m \leq b + 1. \tag{3}$$

All pixels that are inside this convex shape can be represented as a linear sum of endmember proportions. The aim is then to

find a simplex with vertices (endmembers) that minimizes the error term in (1). This can be reformulated to minimizing the residual sum of squares given by

$$\begin{aligned}\text{RSS} &= \sum_{i=1}^{n}\left(Y_i - \sum_{k=1}^{m} p_{ik}M_k\right)^T\left(Y_i - \sum_{k=1}^{m} p_{ik}M_k\right)\\ &= \sum_{j=1}^{d}(y_j - Ae_j)^T(y_j - Ae_j) \end{aligned} \tag{4}$$

where A is a $m \times n$ matrix of proportions, y_j is a vector of n observations of the jth band and e_j is the m-vector of endmember values in the jth band. The minimizer of this is any m-simplex that encompasses all data points.

The linear model described in (1) can be rewritten as

$$\underset{b\times n}{Y} = \underset{b\times m}{M}\ \underset{m\times n}{A} + \underset{b\times n}{N} \tag{5}$$

where Y represents the pixel spectra, M represents the endmember matrix, A represents the abundances, and N is the noise term. To get a reasonably accurate endmembers, the enclosing simplex should minimize the volume that contains all pixel observations. To constrain the size of the simplex, an additional regularization term is added to (4) that measures the total variance of the simplex vertices. High variance now increases the cost function and the vertices are pulled toward the mean. From an optimization point of view, this can be formulated as

$$\begin{aligned}&\arg\min_{M,A}\|Y - MA\|_F^2 + \lambda V(M)\\ &\text{subject to: } A \geq 0,\ 1_m^T A = 1_n^T\end{aligned} \tag{6}$$

where V is the simplex's variance term that promotes mixing matrices of minimal variance with λ controlling the term's relative weight.

One way to solve (6) is to use a cyclic minimization method where the abundances and the endmembers are estimated in repeated fashion, until convergence is reached.

Each iteration in the algorithm is based on two main steps. In the first step, the abundance estimation step, the endmember proportions are estimated by minimizing the residual sum of squares (RSSs) over (1). The second step is to estimate the endmembers given the new abundances.

A. Estimate Abundances

Equation (6) is minimized for abundances. The variance term can be skipped, since it does not depend on abundances

$$\begin{aligned}&\arg\min_{A}\|Y - MA\|_F^2\\ &\text{subject to: } A \geq 0,\ 1_m^T A = 1_n^T.\end{aligned} \tag{7}$$

This equation can be reformulated as a quadratic programming problem.

B. Estimate Endmembers

$$\arg\min_{M}\|Y - MA\|_F^2 + \lambda V(M). \tag{8}$$

Since there are no constraints, (8) has an explicit solution

$$M = \left(AA^T + \lambda\left(I - \frac{1_m 1_m^T}{m}\right)\right)^{-1} AY^T \tag{9}$$

where I is the $m \times m$ identity matrix, 1 is a m-vector of ones, $\lambda = \frac{n\mu}{(m-1)(1-\mu)}$, and μ is a regularization parameter. Further details about how the variance term is implemented can be found in [31].

An outline of the ICE method is given in Algorithm 1.

Algorithm 1. Iterated constrained endmembers (ICE)

1: $Y \leftarrow$ Image matrix$[b \times n]$
2: $M \leftarrow$ Initial endmembers matrix$[b \times m]$
3: $A \leftarrow$ Initial abundance matrix$[m \times n]$
4: **repeat**
5: $\quad A \leftarrow \arg\min_A \|Y - MA\|_F^2,\ A \geq 0,\ 1_p^T A = 1$
6: $\quad M \leftarrow \left(AA^T + \lambda\left(I - \frac{11^T}{m}\right)\right)^{-1}(AY^T)$
7: $\quad v \leftarrow tr(\Sigma M)$
8: $\quad r_i \leftarrow \frac{1-\mu}{n}\|Y - MA\|^2 + \mu v$
9: **until** $r_i / r_{i-1} \geq t$

Lines 7 and 8 are part of a convergence test that is optional and does not strictly need to be calculated at every iteration. The ratio of successive RSS values including the variance term is computed and the program terminates when the ratio exceeds some predefined value t and the method is considered to have converged to a solution.

III. GPU Implementation

In this section, the OpenCL framework is first discussed, with particular emphasis on how it can be used to effectively speed up the ICE algorithm on GPUs. Most importantly, a good quadratic programming solution method needs to be found, which is suitable to the parallel nature of the GPU. All the required kernel programs are elaborated on and a complete parallel implementation is given in pseudocode.

A. GPU Architecture

There are two main competing frameworks for developing parallel algorithms on GPUs. OpenCL was initially developed by Apple Inc., but is now maintained by the nonprofit technology consortium Khronos Group as an open standard. It is designed to allow programs to be written in a relatively platform agnostic way and to run on CPUs, GPUs, digital signal processors (DSPs) [32], field programmable gate arrays (FPGAs) [33], and other processors. CUDA, on the other hand, is a platform that was developed by NVidia specifically for the line of GPUs they produce and, therefore, allows for platform-specific optimizations. There is an ongoing debate of CUDA versus

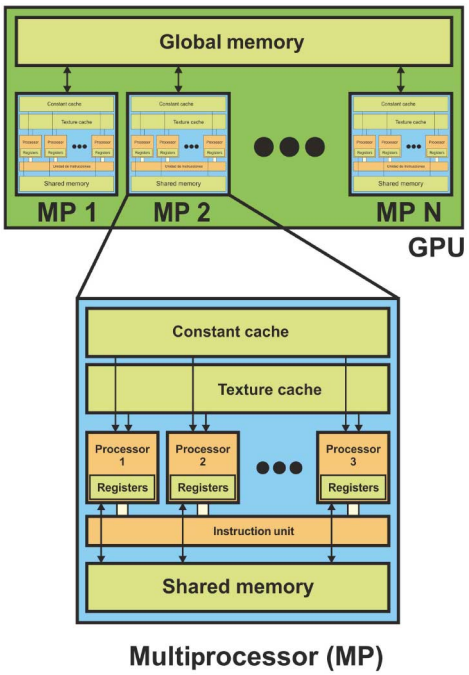

Fig. 1. Schematic overview of a GPU architecture, which can be seen as a set of MPs.

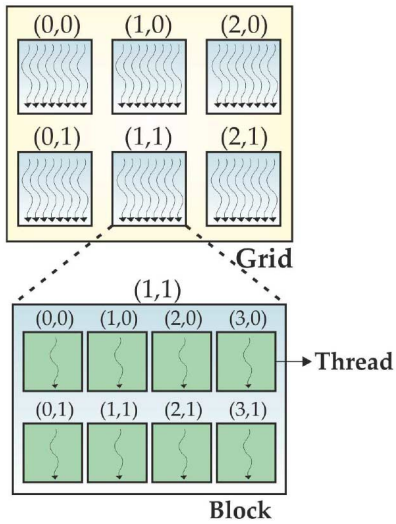

Fig. 2. Batch processing in the GPU: grids of blocks of threads, where each block is composed by a group of threads.

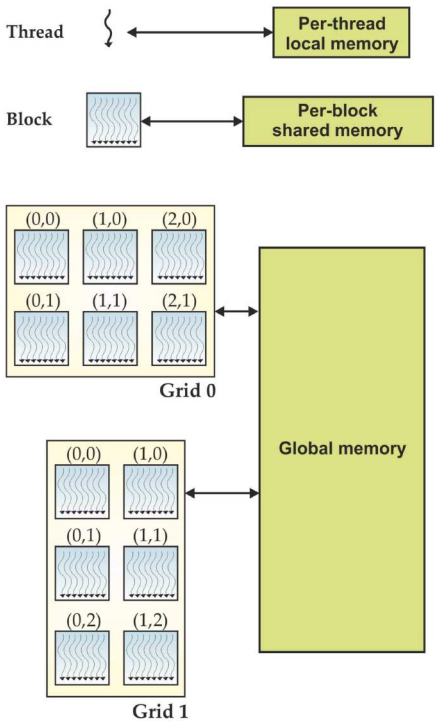

Fig. 3. Different levels of memory in the GPU for the thread, block, and grid concepts.

OpenCL. The portability of OpenCL can come at the price of performance [34], although carefully constructed programs can achieve similar performance as CUDA [35]. This paper values portability above peak performance, and hence OpenCL was chosen for this implementation, but it would be trivial to convert the program to run on the CUDA platform.

GPUs can be abstracted in terms of a stream model, under which all data sets are represented as streams (i.e., ordered data sets). Fig. 1 shows the architecture of a GPU, which can be seen as a set of multiprocessors (MPs). Each MP is characterized by a single instruction multiple data (SIMD) architecture, i.e., in each clock cycle, each processor executes the same instruction, but operating on multiple data streams. Each processor has access to a local shared memory and also to local cache memories in the MP, while the MPs have access to the global GPU (device) memory. Algorithms are constructed by chaining so-called *kernels*, which operate on entire streams and are executed by a MP, taking one or more streams as inputs and producing one or more streams as outputs. Thereby, data-level parallelism is exposed to hardware, and kernels can be concurrently applied without any sort of synchronization. The kernels can perform a kind of batch processing arranged in the form of a grid (workgroup in OpenCL jargon) of blocks (see Fig. 2), where each block is composed by several threads, which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. As a result, there are different levels of memory in the GPU for the thread, block, and grid concepts (see Fig. 3). There is also a maximum number of threads that a block can contain, but the number of threads that can be concurrently executed is much larger (several blocks executed by the same kernel can be managed concurrently, at the expense of reducing the cooperation between threads, since the threads in different blocks of the same grid cannot synchronize with the other threads).

B. Algorithm

By observing the program flow in Algorithm 1, it is possible to identify mainly two potential bottlenecks in the ICE algorithm. The most important one is the minimization problem in line 5, which is a process that can require many iterations to converge. There are also two large matrix multiplications in line 6 that could benefit from parallelization. Assuming no optimization, AA^T requires nm^2 multiplications and

$(n-1)m^2$ additions totaling $(2n-1)m^2$ floating-point operations (FLOPS) and AY^T requires nbm multiplications and $(n-1)bm$ additions totaling $(2n-1)bm$ FLOPS. During multiple iterations of Algorithm 1, the FLOPS multiply, respectively.

For the remainder of this paper, the local workgroup is designed to be two-dimensional with the first dimension (y) fixed to the number of endmembers (m) and second dimension (x) that scales according to defined threads per block. The x dimension is an integer multiple of y that is guaranteed to be lower than a predefined *threads per block* variable. In other words, $dim = [m \times w]$, where $w = \lfloor \text{threads per block}/m \rfloor$. For example, if threads per block is 192 and we search for 19 endmembers, the dimensions of the local workgroup are 19×10 for a total of 190 active threads. This lends itself well to the problem at hand, since most of the calculations are done on matrices with the first dimension equal to the number of endmembers, in practice, which means that local (or global) thread (work-item in OpenCL jargon) identifiers give the position of the element in the matrix.

C. Abundance Estimation

An efficient method to solve the quadratic programming problem in line 5 is essential for achieving good overall performance of the algorithm. Consider the minimization of the quadratic problem in (7). If both abundance nonnegativity (ANC) and abundance sum-to-one constraints (ASCs) are to be satisfied, the problem of abundance estimation becomes complicated. A simple solution is to introduce a soft constraint [36] on the ASC and include it in the ANC formula. To do this, Y and M matrices are augmented with a slack variable δ by adding a row as follows:

$$\begin{aligned} &\arg\min_{A} \|\widetilde{Y} - \widetilde{M}A\|_F^2 \\ &\text{subject to: } A \geq 0 \end{aligned} \tag{10}$$

where $\widetilde{Y} = \begin{bmatrix} Y \\ 1^T\delta \end{bmatrix}$ and $\widetilde{M} = \begin{bmatrix} M \\ 1^T\delta \end{bmatrix}$.

As a consequence of this, for each pixel spectra in Y, the last row in $\widetilde{Y} - \widetilde{M}A$ becomes $\delta - (a_1\delta + \cdots + a_m\delta)$. The minimization of (10) will, therefore, try to maintain the ASC intrinsically with high value of δ enforcing ASC more forcefully. With the aforementioned considerations in mind, the optimization problem (10) can be formulated as a quadratic programming problem as follows:

$$\begin{aligned} &\arg\min_{v} \frac{1}{2} v^T H v + f^T v \\ &\sum_{k=1}^{m} v_k = 1, \quad v_k \geq 0 \; \forall k \end{aligned} \tag{11}$$

where v is a single column vector in A, $H = 2\left(\widetilde{M}^T\widetilde{M}\right)$, and f is a single column vector in $F = \widetilde{M}^T\widetilde{Y}$. The matrix H is symmetric and positive definite, since $z^T H z = z^T \widetilde{M}\widetilde{M}^T z = (\widetilde{M}^T z)^T(\widetilde{M}^T z) > 0$, so its optimization is convex, but due to the nonnegativity constraint an iterative solution is necessary.

The multiplicative margin maximization method [37] provides a suitable solution to nonnegative quadratic programming problems and has been used in hyperspectral unmixing problems before [38]. The matrix H can be split into two nonnegative matrices, $H = H^+ - H^-$ where

$$\begin{aligned} H_{ij}^+ &= \max(H_{ij}, 0) \quad \forall i, j \\ H_{ij}^- &= \max(-H_{ij}, 0) \quad \forall i, j. \end{aligned} \tag{12}$$

The iterative updates are then

$$\widetilde{v}_i = v_i \frac{-f_i + \sqrt{f_i^2 + 4(H^+v)_i(H^-v)_i}}{2(H^+v)_i}. \tag{13}$$

These updates never violate the ANC constraint, since the elements of H^+, H^-, and v are always nonnegative. Therefore, the factor multiplying v_i can only be nonnegative. Using this update, function will decrease the cost function in (11) monotonically to the value of its global minimum. The ASC is not respected by the iterative algorithm, but introduced using soft constraints, as described in (10).

The matrix $F = -2\widetilde{M}^T\widetilde{Y}$ only needs to be evaluated once and then reused each iteration. Computationally, the multiplication of the matrices requires b^2p multiplications and $(b-1)bp$ additions and can benefit from parallelization.

Two separate kernels are developed to solve the quadratic programming problem. The first kernel computes the value for $F = -2\widetilde{M}^T\widetilde{Y}$ and the second kernel uses that information to solve (11). In the following, we describe these two kernels.

1) SolveQP Kernel: A very important property of the iterative updates in (13) is that parallelization comes naturally, since each new element in vector v can be updated independently. This kernel is outlined in Algorithm 2. The loop starting in line 11 needs to read H_g and read/write A_g repeatedly. Therefore, to reduce memory fetches to global memory, it is beneficial to move some of the required content to local (cache) memory. In line 8, A_l is defined as a $[m \times w]$ matrix that holds the required part of the abundance matrix and in line 9, H_l is defined a $[m \times m]$ matrix and holds a copy of the global H_g matrix.

In line 19 of Algorithm 2, there is a test for convergence. This test is optional and does not need to be run every iteration. According to our experiments, running the test for every 50 iterations gives good results. This, however, requires that the old value for the abundance be kept either in private or local memory. In some of our benchmarks, this has been disabled to get a consistent result between platforms.

The local memory footprint per workgroup is kept relatively low for any number of endmembers, since the x-dimension of the workgroup is scaled by m. This results in a local memory consumption of $m^2 + mw$ reals, i.e. with 15 endmembers and 192 threads, the local memory usage is $15^2 + 15\lfloor 192/15 \rfloor = 405$ reals per workgroup.

Algorithm 2. SolveQP Kernel

1: global $H_g \leftarrow$ Initial H matrix $[m \times m]$
2: global $F_g \leftarrow$ Initial F matrix $[m \times n]$
3: global $A_g \leftarrow$ Initial A matrix $[m \times n]$

4: $gy \leftarrow$ get_global_id(0), $gx \leftarrow$ get_global_id(1)
5: $ly \leftarrow$ get_local_id(0), $lx \leftarrow$ get_local_id(1)
6: $lh \leftarrow$ get_local_size(0)
7: $gi \leftarrow gy * lh + gx$
8: local $A_l[ly, lx] \leftarrow A_g[gi]$
9: local $H_l \leftarrow H_g$
10: $f \leftarrow F_g(gi)$
11: **repeat**
12: $v^+ \leftarrow 0, v^- \leftarrow 0$
13: **for** i=0,lh **do**
14: $v^+ \leftarrow v^+ + \max(H_l[ly, i], 0) * A_l[i, lx]$
15: $v^- \leftarrow v^- + \max(-H_l[ly, i], 0) * A_l[i, lx]$
16: **end for**
17: Synchronize Threads
18: $\widetilde{A_l}[ly, lx] \leftarrow A_l[ly, lx] * \frac{\max(-f,0)+v^-}{\max(f,0)+v^+}$
19: **if** $\sum_{k=0}^{m} \left(A_l[k, lx] - \widetilde{A_l}[k, lx]\right)^2 \leq \epsilon$ **then**
20: **break**
21: **end if**
22: **until** maximum iteration
23: $A_g[gi] \leftarrow A_l[ly, lx]$

2) CalculateF Kernel: This kernel simply multiplies the transposed abundance matrix by the pixel matrix and is outlined in Algorithm 3. In addition to multiplying these two matrices, it takes the slack variable δ introduced in (10) as input to satisfy the soft ASC. The kernel is not highly efficient, since it reads from two global variables in a loop, but it is only called once in each iteration of ICE, so there is not much need to optimize it further, since the overall improvement of the processing time will be minimal.

Algorithm 3. CalculateF kernel

1: global $Y_g \leftarrow$ Initial Y matrix $[b \times n]$
2: global $M_g \leftarrow$ Initial M matrix $[b \times m]$
3: global $a_g \leftarrow$ Augmentation value δ
4: global F_g
5: $v \leftarrow a_g^2$
6: $gy \leftarrow$ get_global_id(0), $gx \leftarrow$ get_global_id(1)
7: **for** i=0,b **do**
8: $v \leftarrow v + Y_g[gx * b + i] * M_h[gy * b + i]$
9: **end for**
10: $F_g[gx * m + gy] \leftarrow -2v$

D. Endmember Identification

The endmember identification step of line 6 of Algorithm 1 has three important elements. First, there are the two matrix multiplications: 1) AA^T and 2) AY^T, and then, the matrix inverse operator. In the following, we describe the kernels that perform these operations.

1) Calculate AA^T Kernel: This kernel multiplies the abundance matrix by its transpose and then, adds the lambda term. Two inputs are required, the abundance matrix (A_g) and λ value from (6) with the output stored in memory buffer V_g. Since, the output matrix has dimensionality $m \times m$, the global workgroup dimension for this kernel needs to be $m \times \lceil m/w \rceil * w$. The kernel is outlined in Algorithm 4, the δ in line 6 is the dirac delta function.

Algorithm 4. Calculate AA^T Kernel

1: global $A_g \leftarrow$ Initial abundance matrix$[m \times n]$
2: global λ_g
3: global V_g
4: $gy \leftarrow$ get_global_id(0), $gx \leftarrow$ get_global_id(1)
5: **if** $gx < m$ **then**
6: $v \leftarrow \lambda_g \left(\delta \left(gy - gx\right) - 1/m\right)$
7: **for** p=0,n **do**
8: $i \leftarrow p * m$
9: $v \leftarrow v + A_g[i + gy] + A_g[i + gx]$
10: **end for**
11: $V_g[gx * m + gy] \leftarrow v$
12: **end if**

2) Calculate AY^T Kernel: This kernel multiplies the abundance matrix (M) by the pixel matrix (Y) resulting in an $m \times b$ output matrix (V). The global workgroup dimension for this kernel is $m \times \lceil b/w \rceil * w$. The kernel is outlined in Algorithm 5.

Algorithm 5. Calculate AY^T Kernel

1: global $A_g \leftarrow$ Initial abundance matrix$[m \times n]$
2: global $Y_g \leftarrow$ Initial pixel matrix $[b \times n]$
3: global V_g
4: $gy \leftarrow$ get_global_id(0), $gx \leftarrow$ get_global_id(1)
5: **if** $gx < b$ **then**
6: $v \leftarrow 0$
7: **for** p=0, n **do**
8: $v \leftarrow v + A_g[p * m + gy] + Y_g[p * b + gx]$
9: **end for**
10: $V_g[gx * m + gy] \leftarrow v$
11: **end if**

3) Inverse Operator: The size of the matrix to be inverted is only $m \times m$; therefore, it is preferable to use the highly optimized LAPACK library[3] to invert the matrix on the CPU it instead of doing it on the GPU. The transfer time for such a small matrix is also negligible.

IV. Experimental Results

In this section, the performance of the proposed method is analyzed via two experiments. The first experiment uses a synthetic data set where endmembers from spectral library are used. In the second experiment, a real data set collected over the Cuprite mining district in Nevada is used, and the extracted endmembers are compared against some highly represented elements using the United States Geological Survey (USGS) spectral library. For comparative purposes, two other implementations in addition to the GPU one have been compared: 1) a single thread version of the algorithm in C++ and 2) a high level implementation in MATLAB. The test hardware for the parallel version is Nvidia Tesla M2090 GPU with 512 cuda

[3][Online]. Available: http://www.netlib.org/lapack, accessed on April 7, 2015.

cores, 6 GB of memory, and capable of floating point peak performance of 1330 GFLOPS. The hardware used to run the single thread applications is a Linux machine with 64 GB of main memory and Intel Xeon CPU Model E5-2648L running at 1.80 GHz with a peak performance of 115.2 GFLOPS.

Both the parallel version and the single core versions were compiled using GNU compiler (g++) with the *O3* flag for highest optimization level and *SSE* enabled, which allows significant portion of the QP routine to be vectorized. The vectorizer verbose output was used to verify that important parts of the algorithms source code was vectorized. Denormal floating point numbers are set to zero in code using the macro *_MM_SET_DENORMALS_ZERO_MODE(_MM_DENORMALS_ZERO_ON);* which does not have significant impact on accuracy, but can slightly enhances performance. All matrix and vector operation are performed using optimized version of the BLAS library, and LAPACK is used to solve a system of linear equations. To evaluate the accuracy of the methods, the following metrics are defined.

1) Spectral angle distance (SAD) [2] measures the angle between two vectors based on their dot product and is not affected by signal scaling and sign. SAD is given in degrees and a value of zero means two identical vectors

$$\text{SAD} = cos^{-1}\left(\frac{m^T\hat{m}}{\|m\|\|\hat{m}\|}\right)\frac{180}{\pi} \tag{14}$$

where m and $\hat{m}$ are two column vectors.

2) The mean square error (MSE) is denoted as the Frobenius norm of the difference of two endmember matrices

$$\text{MSE} = \|M - \hat{M}\|_F. \tag{15}$$

A. Synthetic Data Set

In this experiment, a synthetic data set is created consisting of 221 bands and 10 000 pixels constructed using the linear mixture model with nine endmembers. This gives us a complete control over the data set and, therefore, allows for accurate estimation of the endmember identification and abundance estimation accuracy. The spectral signatures used to generate the data are randomly selected from the USGS library. The abundance matrix is randomly generated using a Dirichlet distribution [10], so that the samples are uniformly distributed over the simplex with the exception that no single sample can have abundance fraction higher than 0.80 to guarantee that no pure pixels exist in the simulated data. This can, e.g., describe a low spatial resolution image where each pixel is heavily mixed. Gaussian noise is then added, so that the signal-to-noise ratio (SNR) is 50 dB. Table I shows the selected endmembers and the single maximum abundance of the corresponding spectra in the simulated abundance matrix.

Fast methods such as PPI or VCA that assume the presence of pure pixels in the data can be used to extract sane values for the initialization of the ICE algorithm. In this case, the VCA was used to extract nine vertices that are used as a starting point. To facilitate a comparison between the three different versions of the program, the convergence test in the quadratic programming solver (Algorithm 2, line 19) is skipped, and a 500 iteration limit is set on the loop. This is mainly done to simplify the vectorization of loops in MATLAB to maximize speed.

TABLE I
NINE ENDMEMBERS RANDOMLY SELECTED FROM THE USGS LIBRARY

#	Name	Maximum abundance
1	Olivine KI3291 <60um	0.7319
2	Witherite HS273.3B	0.7214
3	Dolomite HS102.3B	0.7012
4	Rutile HS126.3B	0.6729
5	Jarosite WS368 Pb	0.7015
6	Praseodymium_Oxide GDS35	0.7656
7	Erionite+Merlinoit GDS144	0.6560
8	Coquimbite GDS22	0.6406
9	Topaz Wigwam_Area_6_#16	0.7072

TABLE II
PROCESSING TIME (IN SECONDS) FOR 5000 ITERATIONS

Program	Total time	Mean iteration time	Relative time
OpenCL	93.63	0.0187	1.00
C++	4080.56	0.8161	43.64
Matlab	1710.18	0.3420	18.29
OpenCL*	37.25	0.0075	0.40

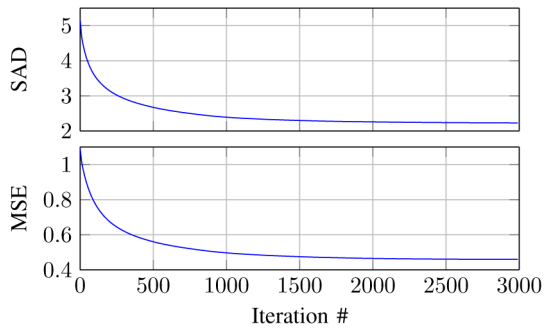

Fig. 4. Results from the ICE algorithm by iteration shows: (a) mean SAD score and (b) mean MSE for the estimated endmembers when compared to their true values.

To guarantee an accurate value for the mean iteration time, Algorithm 1 is iterated 5000 times and Table II shows the processing time. The slack variable for the ASC constraint is $\delta = 1$, and the quadratic solver is set to 500 iterations (except for entry marked with asterisk in the table, where the convergence test is allowed to terminate the loop prematurely) and $\mu = 10^{-5}$.

Fig. 4 shows how the mean SAD and MSE values converge as the iterations increase. From the plots, it is clear that the quality of the estimated endmembers increases fast after the first couple of hundred iterations (and very slowly after 3000 iterations).

To assess the accuracy of the method, a set M of nine endmembers were estimated by iterating the ICE algorithm 3000 times. Then for each extracted endmember, a SAD score was calculated against a set F that contains each of the nine true endmember spectra that the data set was created from. The endmember and spectra with the lowest SAD scores were said to have the same spectral signature and removed from both M and F sets. This is repeated until all endmembers in M have been matched with a spectra from F. Fig. 5 shows the spectral

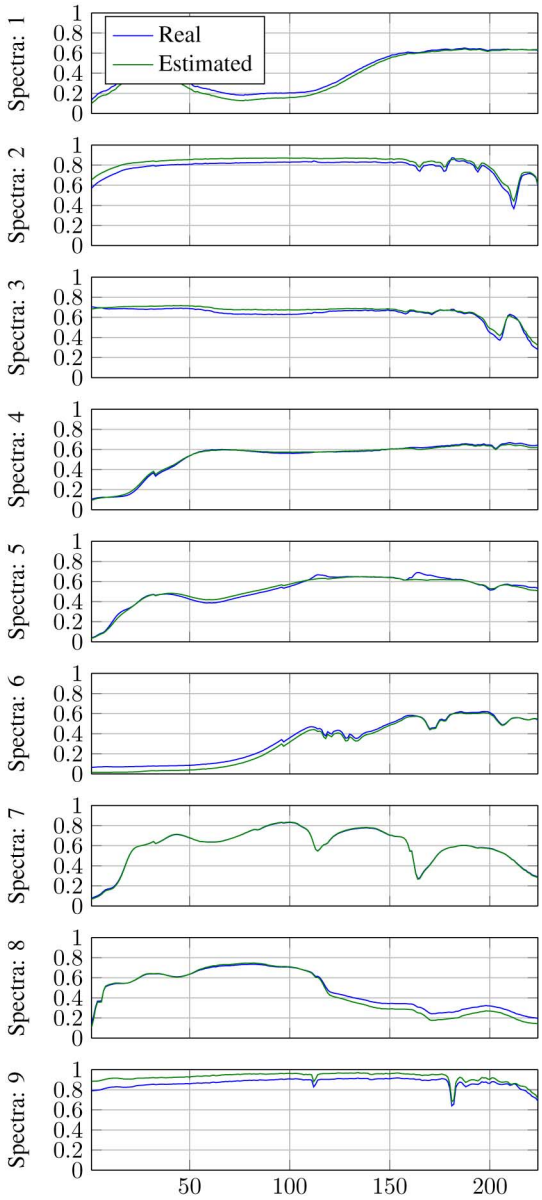

Fig. 5. Estimated endmembers after 3000 iterations of the algorithm. The mean SAD is 0.038910 and the mean MSE is 0.460266.

signatures from Table I with a matching extracted endmember and Table III shows the MSE and SAD values for the corresponding pairs. As a reference, mean SAD and MSE values for initial values for the endmembers that were extracted using VCA are 5.5079 and 1.1339, respectively. These results indicate the accuracy obtained by the proposed method in the task of endmember identification and abundance estimation in the considered scenario. The soft constraint encourages a sum-to-one for the abundances, this is verified by summing the abundances for each pixel, which gives a mean value of 1.0001 and variance of 0.0018.

B. Cuprite Data Set

In our final experiment, the method is evaluated on a real hyperspectral data set. The hyperspectral scene used is the well-known airborne visible infra-red imaging spectrometer (AVIRIS) Cuprite data set, available online in reflectance units.[4] This scene has been widely used to validate the performance of endmember extraction algorithms. 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. The scene comprises 224 spectral bands between 0.4 and 2.5 μm, with nominal spectral resolution of 10 nm. Prior to the analysis, bands 1–2, 105–115, 150–170, and 223–224 were removed due to water absorption and low SNR in those bands, leaving a total of 188 spectral bands. The Cuprite site is well understood mineralogically, and has several exposed minerals of interest, all included in the USGS library considered in experiments, denoted splib06[5] and released in September 2007. In our experiments, we use spectra obtained from this library to substantiate the quality of the endmembers derived by MVSA and compare them with those produced by other algorithms. For illustrative purposes, Fig. 6 shows a mineral map produced in 1995 by USGS, in which the Tricorder 3.3 software product was used to map different minerals present in the Cuprite mining district.[6]

There are many ways to estimate the accuracy of the extracted endmembers. For instance, in [39], five spectra from the USGS library that are highly representative in the Cuprite data set are compared against 19 extracted endmembers. Specifically, the spectral signatures of the minerals are: alunite (GDS84), buddingtonite (GDS85), calcite (WS272), kaolinite (KGa-1), and muscovite (GDS107). Table IV shows the selected spectra and the best SAD in degrees to any of the endmembers extracted by running ICE for 100 iterations with $\mu = 0.0001$ and $\delta = 8929$. The whole AVIRIS Cuprite scene, comprising 1 22 500 pixels, has been processed, where each pixel has a spectral dimension of 188 bands. When extracting 19 endmembers, the total processing time is 24.94 s for the OpenCL version with an average iteration time 0.2494 s. In the following, we provide a more detailed evaluation of performance.

TABLE III
MSE AND SAD FOR ESTIMATED ENDMEMBERS

Spectra #	SAD	MSE
1	3.9342	0.5558
2	1.1012	0.6397
3	1.4643	0.4266
4	1.0720	0.1559
5	2.3186	0.3248
6	4.7394	0.5337
7	0.3444	0.0568
8	4.0628	0.5695
9	1.0277	0.8795
Mean	2.2294	0.4603

[4][Online]. Available: http://aviris.jpl.nasa.gov/html/aviris.freedata.html, accessed on April 7, 2015.

[5][Online]. Available: http://speclab.cr.usgs.gov/spectral.lib06, accessed on April 7, 2015.

[6][Online]. Available: http://speclab.cr.usgs.gov/cuprite95.tgif.2.2um_map.gif, accessed on April 7, 2015.

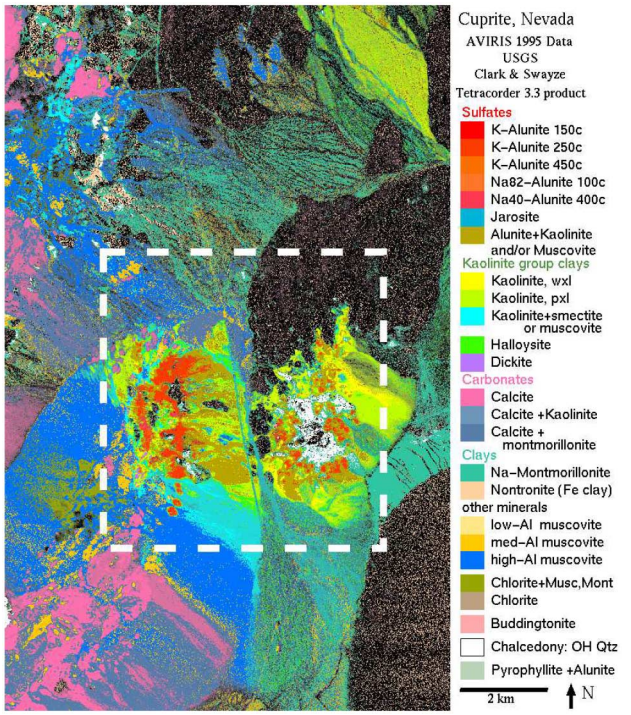

Fig. 6. USGS map showing the location of different minerals in the Cuprite mining district in Nevada.

TABLE IV
BEST SAD BETWEEN SELECTED SPECTRA AND ESTIMATED ENDMEMBERS IN THE AVIRIS CUPRITE SCENE

Spectra	SAD
Alunite GDS84 Na03	3.3647
Buddingtonite GDS85 D-206	7.060
Calcite WS272	5.6681
Kaolinite KGa-1	6.3115
Muscovite GDS107	5.7068
Mean	5.6222

TABLE V
TIME TAKEN BY THE DIFFERENT KERNELS USED BY THE OPENCL IMPLEMENTATION OF ICE

	Total	Abund. est.		Endm. est.		IO
		M^TY	QP	AA^T	AY^T	
Seconds	75.070	6.390	62.100	2.140	3.680	0.760
Relative	1.000	0.085	0.827	0.029	0.049	0.010

C. Evaluation of Performance

To evaluate the computational performance of our OpenCL implementation of ICE, Table V shows a breakout of the time taken by the different kernels used in the implementation. The Cuprite scene is processed the same way as in the previous section except to simplify comparison the QP solver is set to 500 iterations and the convergence test is skipped. This allows us to identify which part of the method is the most time consuming. As shown by the table, the QP kernel takes up roughly 83% of the total processing time. This is directly proportional to how many iterations of the QP solver are performed.

TABLE VI
TIME TAKEN BY THE DIFFERENT KERNELS USED BY THE OPENCL IMPLEMENTATION OF ICE ALLOWING EARLY QP EXIT

	Total	Abund. est.		Endm. est.		IO
		M^TY	QP	AA^T	AY^T	
Seconds	22.730	6.250	9.940	2.080	3.720	0.740
Relative	1.000	0.275	0.437	0.088	0.164	0.033

TABLE VII
COMPARISON OF PERFORMANCE ACHIEVED BY DIFFERENT IMPLEMENTATIONS OF THE ICE ALGORITHM

	Mean time per iteration (s)					
Implementation	Total	M^TY	QP	AA^T	AY^T	I/O
OpenCL	0.753	0.063	0.622	0.021	0.037	0.009
OpenCL*	0.227	0.063	0.099	0.021	0.037	0.006
C++	32.506	0.200	32.087	0.025	0.194	–
MATLAB	14.148	0.272	13.817	0.019	0.040	–
Speedup over C++	43.169	3.175	51.587	1.190	5.243	–
Speedup over MATLAB	18.789	4.317	22.214	0.905	1.081	–

An obvious modification is to allow the convergence test on the abundance matrix. This reduces the iterations required for the QP solver considerably as shown in Table VI. However, in MATLAB, this is not so simple to implement, since the loops are vectorized for performance reasons.

Allowing the convergence test to exit the QP loop early will give slightly different abundance matrices and in turn endmembers. To evaluate the error, normalized mean square error of the two resulting endmember matrices (M and $\hat{M}$) is defined as

$$\text{nMSE} = \frac{\|M - \hat{M}\|_F}{\|M\|_F}. \tag{16}$$

For the previous example, the mean value of the nMSE for the 19 endmembers is found to be 0.0034. This is ultimately a balance between accuracy and speed. In this case, the cost of mean nMSE of 0.0034 for 6.2475 speed increase of the QP kernel.

Finally, Table VII provides a comparison between the GPU times (measured for the different kernels) and for the single core versions of the algorithm (in MATLAB and C++). The table also includes details about the transfer times, which are labeled as I/O (for input/output). The table also provides the speedup achieved by the GPU implementation from the C++ and the MATLAB version. From this table, we can conclude that the OpenCL implementation outperforms the other C++ and MATLAB versions by one order of magnitude. The QP solver has the biggest impact on the overall performance and, therefore, benefits the most from optimization. The other kernels are rather basic, using global memory fetches that could be improved, but still compare to and even outperform highly optimized CPU libraries, such as BLAS for C++ and MATLAB. But as previously stated, the cumulative effect on overall performance is small, so the benefit of the optimization is smaller than for the QP kernel. This is evident in line 2 of Table VII,

where the convergence test of the QP kernel has been enabled and the iteration time is considerably reduced even further.

V. Conclusion and Future Research

In this work, we have developed a GPU implementation of the ICE algorithm for endmember identification in remotely sensed hyperspectral data. This algorithm is an instance of the family of algorithms that do not assume the presence of pure pixels in the data, and has been successfully used in many scenarios. Our implementation utilizes the parallel nature of GPUs to accelerate the algorithm, and has been developed in OpenCL, which allows its portability to GPU cards developed by any vendor. The evaluation of the parallel implementation has been conducted using both synthetic and real data sets, obtaining very promising results. When compared to a single core version using MATLAB and C++, the GPU implementation is shown to be orders of magnitude faster. The method is also easily scalable to increase the number of desired endmembers or pixels to process in the original hyperspectral image. Although the results obtained are very promising, it is felt that the performance of the parallel algorithm can be further increased in future developments by resorting to multi-GPU platforms. A more detailed comparison of other parallel endmember identification algorithms implemented on GPUs with OpenCL is desirable.

References

[1] J. Bioucas-Dias *et al.*, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 354–379, Apr. 2012.

[2] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 44–57, Jan. 2002.

[3] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. 110–122, 2009.

[4] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 650–663, Mar. 2004.

[5] A. Plaza, Q. Du, J. Bioucas-Dias, X. Jia, and F. Kruse, "Foreword to the special issue on spectral unmixing of remotely sensed data," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4103–4110, Nov. 2011.

[6] R. Heylen, M. Parente, and P. Gader, "A review of nonlinear hyperspectral unmixing methods," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 1844–1868, Jun. 2014.

[7] M.-D. Iordache, J. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 6, pp. 2014 –2039, Jun. 2011.

[8] J. Bioucas-Dias and J. Nascimento, "Hyperspectral subspace identification," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 8, pp. 2435–2445, Aug. 2008.

[9] J. Boardman, "Automating spectral unmixing of AVIRIS data using convex geometry concepts," in *Proc. Ann. JPL Airborne Geosci. Workshop*, vol. 1, 1993, pp. 11–14.

[10] J. Nascimento and J. Bioucas-Dias, "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 4, pp. 898–910, Apr. 2005.

[11] C.-I. Chang, C.-C. Wu, W. Liu, and Y.-C. Ouyang, "A new growing method for simplex-based endmember extraction algorithm," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 10, pp. 2804–2819, Oct. 2006.

[12] T.-H. Chan, W.-K. Ma, A. Ambikapathi, and C.-Y. Chi, "A simplex volume maximization framework for hyperspectral endmember extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4177–4193, Nov. 2011.

[13] N. Gillis and S. Vavasis, "Fast and robust recursive algorithms for separable nonnegative matrix factorization," arXiv preprint arXiv:1208.1237, 2012.

[14] M. E. Winter, "N-FINDR: An algorithm for fast autonomous spectral endmember determination in hyperspectral data," in *Proc. SPIE Image Spectrom. V*, vol. 3753, 1999, pp. 266–277.

[15] R. A. Neville, K. Staenz, T. Szeredi, J. Lefebvre, and P. Hauff, "Automatic endmember extraction from hyperspectral data for mineral exploration," in *Proc. Can. Symp. Remote Sens.*, 1999, pp. 21–24.

[16] M. Craig, "Minimum-volume transforms for remotely sensed data," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, pp. 542–552, May 1994.

[17] M. Berman, H. Kiiveri, R. Lagerstrom, A. Ernst, R. Dunne, and J. F. Huntington, "ICE: A statistical approach to identifying endmembers in hyperspectral images," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 10, pp. 2085–2095, Oct. 2004.

[18] L. Miao and H. Qi, "Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 3, pp. 765–777, Mar. 2007.

[19] A. Zare and P. Gader, "Sparsity promoting iterated constrained endmember detection for hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, no. 3, pp. 446–450, Jul. 2007.

[20] J. Bioucas-Dias, "A variable splitting augmented Lagrangian approach to linear spectral unmixing," in *Proc. IEEE GRSS Workshop Hyperspectral Image Signal Process.: Evol. Remote Sens. (WHISPERS)*, 2009 pp. 1–4.

[21] T. Chan, C. Chi, Y. Huang, and W. Ma, "Convex analysis based minimum-volume enclosing simplex algorithm for hyperspectral unmixing," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4418–4432, Nov. 2009.

[22] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, pp. 56–69, Mar.–Apr. 2010.

[23] A. Plaza, J. Plaza, A. Paz, and S. Sánchez, "Parallel hyperspectral image and signal processing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 119–126, May 2011.

[24] S. Sanchez and A. Plaza, "GPU implementation of the pixel purity index algorithm for hyperspectral image analysis," in *Proc. IEEE Int. Conf. Cluster Comput. Workshops Posters (CLUSTER WORKSHOPS'10)*, 2010, pp. 1–7.

[25] S. Sánchez, G. Martin, and A. Plaza, "Parallel implementation of the N-FINDR endmember extraction algorithm on commodity graphics processing units," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS'10)*, 2010, pp. 955–958.

[26] A. Plaza, P. Martinez, R. Perez, 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.

[27] J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado, "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geosci. Remote Sens. Lett.*, vol. 43, no. 3, pp. 441–445, Jul. 2007.

[28] S. Sánchez, A. Paz, G. Martin, and A. Plaza, "Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units," *Concurrency Comput.: Pract. Exper.*, vol. 23, no. 13, pp. 1538–1557, Sep. 2011.

[29] S. Bernabe, S. Sanchez, A. Plaza, S. Lopez, J. Benediktsson, and R. Sarmiento, "Hyperspectral unmixing on GPUs and multi-core processors: A comparison," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 6, no. 3, pp. 1386–1398, Jun. 2013.

[30] S. Sanchez, G. Leon, A. Plaza, and E. Quintana-Orti, "Assessing the performance-energy balance of graphics processors for spectral unmixing," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2305–2316, Jun. 2014.

[31] M. Berman, H. Kiiveri, R. Lagerstrom, A. Ernst, R. Dunne, and J. F. Huntington, "Ice: A statistical approach to identifying endmembers in hyperspectral images: Learning from Earth's shapes and colors," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 10, pp. 2085–2095, Oct. 2004.

[32] M. Castillo, J. Fernandez, F. Igual, A. Plaza, E. Quintana-Orti, and A. Remon, "Hyperspectral unmixing on multicore DSPs: Trading off performance for energy," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2297–2304, Jun. 2014.

[33] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza, "The promise of reconfigurable computing for hyperspectral imaging onboard systems: A review and trends," *Proc. IEEE*, vol. 101, no. 3, pp. 698–722, Mar. 2013.

[34] J. Fang, A. L. Varbanescu, and H. Sips, "A comprehensive performance comparison of CUDA and OpenCL," in *Proc. Int. Conf. Parallel Process. (ICPP'11)*, 2011, pp. 216–225.

[35] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, and J. Dongarra, "From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming," *Parallel Comput.*, vol. 38, no. 8, pp. 391–407, 2012.

[36] D. C. Heinz and C.-I. Chang, "Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 3, pp. 529–545, Mar. 2001.
[37] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1041–1048.
[38] J. Sigurdsson, M. O. Ulfarsson, and J. R. Sveinsson, "Hyperspectral unmixing with $l\_\{q\}$ regularization," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 11, pp. 6793–6806, Nov. 2014.
[39] G. Martin and A. Plaza, "Region-based spatial preprocessing for endmember extraction and spectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 745–749, Jul. 2011.

Eysteinn Már Sigurdsson received the B.S. degree in 2006 and the M.S. degree in 2008, both in electrical and computer engineering from the University of Iceland, Reykjavik, Iceland. He is currently pursuing the Ph.D. degree in electrical and computer engineering from the Faculty of Electrical and Computer Engineering, University of Iceland.

His research interests include hyperspectral image processing and high-performance computing.

Antonio Plaza (M'05–SM'07–F'15) was born in Caceres, Spain, in 1975. He received the Computer Engineer degree in 1997, the Master degree in computer engineering in 1999, and the Ph.D. degree in computer in engineering in 2002, all from the University of Extremadura, Cáceres, Spain.

He is an Associate Professor (with accreditation for Full Professor) with the Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp). He has been the advisor of 12 Ph.D. dissertations and more than 30 M.Sc. dissertations. He was the Coordinator of the Hyperspectral Imaging Network, a European project with total funding of 2.8 million Euro. He has authored more than 400 publications, including 142 journal papers (92 in IEEE journals), 20 book chapters, and over 240 peer-reviewed conference proceeding papers (94 in IEEE conferences). He has edited a book on *High-Performance Computing in Remote Sensing* (CRC Press/Taylor & Francis) and guest edited eight special issues on hyperspectral remote sensing for different journals. He has served as a Proposal Evaluator for the European Commission, the National Science Foundation, the European Space Agency, the Belgium Science Policy, the Israel Science Foundation, and the Spanish Ministry of Science and Innovation. He has reviewed more than 500 articles for over 50 different journals. His research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) in 2011–2012, and is currently serving as President of the Spanish Chapter of IEEE GRSS (since November 2012). He is currently serving as the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING journal. He is also an Associate Editor for IEEE ACCESS, and was a member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER (2011–2012) and the IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE (2013). He was also a member of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He was a recipient of the Recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, in 2009, the Recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, in 2010, a journal for which he served as Associate Editor in 2007–2012, the 2013 Best Paper Award of the JSTARS journal, the most highly cited paper (2005–2010) in the *Journal of Parallel and Distributed Computing*, the Best Paper Awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology, and the best Ph.D. Dissertation Award at the University of Extremadura.

Jón Atli Benediktsson (S'84–M'90–SM'99–F'04) received the Cand.Sci. degree in electrical engineering from the University of Iceland, Reykjavik, Iceland, in 1984, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1987 and 1990, respectively.

He is currently Rector and Professor of Electrical and Computer Engineering at the University of Iceland. He is the Co-Founder of the biomedical start-up company Oxymap, Reykjavik, Iceland. His research interests include remote sensing, biomedical analysis of signals, pattern recognition, image processing, and signal processing, and he has published extensively in those fields.

Dr. Benediktsson was the President of the IEEE Geoscience and Remote Sensing Society (GRSS), 2011–2012, and has been on the GRSS Administrative Committee, since 2000. He was Editor-in-Chief of the IEEE Transactions on Geoscience and Remote Sensing (TGRS) from 2003 to 2008 and has served as Associate Editor of TGRS since 1999, the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS since 2003 and IEEE ACCESS since 2013. He was a member of the 2014 IEEE Fellow Committee. He is a member of the Association of Chartered Engineers in Iceland (VFI), Societas Scinetiarum Islandica, and Tau Beta Pi. He is on the Editorial Board of the PROCEEDINGS OF THE IEEE, the International Editorial Board of the *International Journal of Image and Data Fusion* and was the Chairman of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS) from 2007 to 2010. He is a Fellow of the Society of Photographic Instrumentation Engineers (SPIE). He was the recipient of the Stevan J. Kristof Award from Purdue University in 1991 as an Outstanding Graduate Student in remote sensing, the Icelandic Research Council's Outstanding Young Researcher Award in 1997, the IEEE Third Millennium Medal in 2000, the Yearly Research Award from the Engineering Research Institute of the University of Iceland in 2006, the Outstanding Service Award from the IEEE Geoscience and Remote Sensing Society in 2007, and the IEEE/VFI Electrical Engineer of the Year Award in 2013. He was the corecipient of the University of Iceland's Technology Innovation Award in 2004, the 2012 IEEE Transactions on Geoscience and Remote Sensing Paper Award, the IEEE GRSS Highest Impact Paper Award in 2013 and the 2012-2013 Best Paper Award from the International Journal of Image and Data Fusion.