PARALLEL HYPERSPECTRAL IMAGE COMPRESSION USING ITERATIVE ERROR ANALYSIS ON GRAPHICS PROCESSING UNITS

Sergio Sánchez and Antonio Plaza

Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, E-10071 Caceres, Spain.

ABSTRACT

In this paper, we develop a new parallel implementation of the iterative error analysis (IEA) algorithm for lossy hyperspectral image compression on graphics processing units (GPUs), an inexpensive parallel computing platform that has recently become very popular in hyperspectral imaging applications. The proposed GPU implementation is tested on several different architectures from NVidia, the main GPU vendor worldwide, and is shown to exhibit real-time performance in the analysis of AVIRIS data sets. The GPU implementation of the IEA represents a step forward towards real-time onboard (lossy) compression of hyperspectral data where the quality of the compression can be also adjusted in real-time.

***Index Terms*—** Hyperspectral imaging, iterative error analysis, lossy hyperspectral compression, endmember extraction, GPUs.

1. INTRODUCTION

Hyperspectral imaging allows an imaging spectrometer to collect hundreds of bands (at different wavelength channels) for the same area on the surface of the Earth [1]. For instance, the NASA Jet Propulsion Laboratory's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) covers the wavelength region from 0.4 to 2.5 microns using 224 spectral channels, at nominal spectral resolution of 10 nanometers [2]. The resulting multidimensional data cube typically comprises several gigabytes per flight. Due to the extremely large volumes of data collected by imaging spectrometers, hyperspectral data compression has received considerable interest in recent years [3, 4]. These data are usually collected by a satellite or an airbone instrument and sent to a ground station on Earth for subsequent processing. Usually the bandwith connection between the satellite/airborne platform and the ground station is reduced, which limits the amount of data that can be transmitted. As a result, there is a clear need for (either lossless or lossy) hyperspectral data compression techniques that can be applied onboard the imaging instrument [5].

A successful algorithm to perform spectral unmixing-based [6] lossy compression of remotely sensed hyperspectral data is the iterative error analysis (IEA) algorithm [7], which applies an iterative process which allows controlling the amount of information loss and compression ratio depending on the number of iterations performed by the algorithm. This algorithm can be computationally expensive for hyperspectral images with high dimensionality [8]. In this paper, we develop a new parallel implementation of the IEA algorithm for hyperspectral image compression on graphics processing units (GPUs), an inexpensive parallel computing platform that has recently become very popular in hyperspectral imaging applications [9–13]. The proposed GPU implementation is tested on several different architectures from NVidia[1], the main GPU vendor worldwide, and is shown to exhibit real-time performance in the analysis of AVIRIS data sets. The GPU implementation of the IEA represents a significant advance towards real-time onboard (lossy) compression of hyperspectral data where the quality of the compression can be also adjusted in real-time. The remainder of the paper is organized as follows. Section 2 describes the IEA algorithm, which is based on spectral unmixing concepts, and further develops a lossy compression algorithm for hyperspectral data based on this algorithm. Section 3 outlines its parallel implementation on GPUs. Section 4 evaluates the proposed GPU implementation using real hyperspectral data collected by AVIRIS. The section concludes with some remarks and hints at plausible future research.

2. ITERATIVE ERROR ANALYSIS (IEA) FOR LOSSY COMPRESSION OF HYPERSPECTRAL DATA

The IEA algorithm is based on the concept of spectral unmixing of hyperspectral data. In order to define the mixture problem in mathematical terms, let us assume that a remotely sensed hyperspectral scene with n bands is denoted by $\mathbf{X}$, in which the pixel at the discrete, spatial coordinates (i, j) of the scene is represented by a feature vector given by $\mathbf{X}(i,j) = [x_1(i,j), x_2(i,j), \cdots, x_n(i,j)] \in \Re^n$, and $\Re$ denotes the set of real numbers corresponding to the pixel's spectral response $x_k(i,j)$ at sensor channels $k = 1, \ldots, n$. Under a linear mixture model assumption [14], each pixel vector in the original scene can be modeled using the following expression:

$$\mathbf{X}(i,j) = \sum_{k=1}^{p} \Phi_k(i,j) \cdot \mathbf{E}_k + \mathbf{n}(i,j), \quad (1)$$

where $\mathbf{E}_k$ denotes the spectral response of the k-th endmember, $\Phi_z(i,j)$ is a scalar value designating the abundance of the k-th endmember at pixel $\mathbf{X}(i,j)$, p is the total number of endmembers, and $\mathbf{n}(i,j)$ is a noise vector. The solution of the linear spectral mixture problem described in Eq. (1) relies on the correct determination of a set of p endmembers denoted by $\{\mathbf{E}_k\}_{k=1}^{p}$. For this purpose, the IEA [7] performs a series of spectral unmixing operations, each time selecting as endmembers the pixels that minimize the error in the reconstruction of the original image after the unmixing. An advantage of this approach over other available algorithms is that the IEA not only produces a set of endmembers but also their abundances in each

This work has been supported by the European Community's Marie Curie Research Training Networks Programme under contract MRTN-CT-2006-035927, Hyperspectral Imaging Network (HYPER-I-NET). Funding from the Spanish Ministry of Science and Innovation (CEOS-SPAIN project, reference AYA2011-29334-C02-02) is also gratefully acknowledged.

[1] http://www.nvidia.com

pixel of the scene. As a result, it can be used to compress (in lossy fashion) a hyperspectral image $\mathbf{X}$ using the enmember set $\{\mathbf{E}_k\}_{k=1}^{p}$ and the associated fractional abundances at a pixel level. The more endmembers extracted and associated abundance maps, the higher the quality and size of the compressed image. Our implementation of the IEA algorithm can be summarized as follows:

1. *Initialization*. The sample n-dimensional mean vector $\bar{\mathbf{X}}$ of the original hyperspectral image $\mathbf{X}$ is first calculated as:

$$\bar{\mathbf{X}} = \frac{1}{r \times c} \sum_{i=1}^{r} \sum_{j=1}^{c} \mathbf{X}(i,j), \tag{2}$$

where r denotes the number of rows and c denotes the number of columns in $\mathbf{X}$.

2. *Initial endmember calculation*. Let the endmember set $\mathbf{E}$ be initially an empty set, i.e. $\mathbf{E} = \emptyset$. The first endmember pixel $\mathbf{E}_1$ is calculated as follows. First, a reconstructed version $\hat{\mathbf{X}}$ of the original hyperspectral image $\mathbf{X}$ is obtained by performing a spectral unmixing of $\mathbf{X}$ using $\bar{\mathbf{X}}$ as the only spectral endmember. In our implementation of IEA, we apply a simple unconstrained spectral unmixing at each pixel $\mathbf{X}(i,j)$ as follows: $(\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^T\mathbf{X}(i,j)$. The outcome of this operation is an abundance value $\Phi_0(i,j)$ for each pixel in $\mathbf{X}$. The reconstruction is now simply obtained by applying the following expression to all hyperspectral image pixels:

$$\hat{\mathbf{X}}(i,j) = \Phi_0(i,j) \cdot \hat{\mathbf{X}}. \tag{3}$$

Now we calculate the root mean square error (RMSE) between the original and the reconstructed hyperspectral scenes using the following expression:

$$\text{RMSE}(\mathbf{X}, \hat{\mathbf{X}}) = \left(\frac{1}{r \times c} \right) \sum_{i=1}^{r} \sum_{j=1}^{c} \left(\frac{1}{n} \sum_{k=1}^{n} [x_k(i,j) - \hat{x}_k(i,j)]^2 \right)^{\frac{1}{2}}, \tag{4}$$

and select the first endmember as the pixel with maximum reconstruction error: $\mathbf{E}_1 = \text{argmax}_{(i,j) \in Z^2(X)} \text{RMSE}(\mathbf{X}, \hat{\mathbf{X}})$. The resulting pixel vector is stored in the endmember set: $\mathbf{E} = \{\mathbf{E}_1\}$.

3. *Iterative process*. Calculate a new endmember for iterations $2 \leq k \leq p$ by computing an unconstrained spectral unmixing at each pixel $\mathbf{X}(i,j)$ using the current set of endmembers $\mathbf{E}$ as follows: $(\hat{\mathbf{E}}^T\hat{\mathbf{E}})^{-1}\hat{\mathbf{E}}^T\mathbf{X}(i,j)$. The outcome of this operation is a set of abundance values $\{\Phi_k(i,j)\}_{k=1}^{q}$ for each pixel, where q is the number of derived endmembers until that moment, and $q \leq p$. The reconstruction is now obtained by applying the following expression to all image pixels:

$$\hat{\mathbf{X}}(i,j) = \sum_{k=1}^{q} \Phi_k(i,j) \cdot \mathbf{E}_k. \tag{5}$$

Now we can select the k-th endmember $\mathbf{E}_k$ as the pixel with maximum associated reconstruction error as follows: $\mathbf{E}_k = \text{argmax}_{(i,j) \in Z^2(\mathbf{X})} \text{RMSE}(\mathbf{X}, \hat{\mathbf{X}})$. The resulting pixel (at the current iteration) is now stored: $\mathbf{E} = \{\mathbf{E}_1, \cdots, \mathbf{E}_k\}$.

4. *Compression*. The procedure is terminated when $k = p$. In this case, a final set of endmembers $\mathbf{E} = \{\mathbf{E}_1, \cdots, \mathbf{E}_p\}$ and their corresponding abundances $\{\Phi_k(i,j)\}_{k=1}^{p}$ in each pixel $\mathbf{X}(i,j)$ are produced as the outcome of the algorithm and can be used to represent the original image in terms of the extracted endmembers and their associated abundances. The decompression of the data can be done by simply applying Eq. (1) to obtain the original image $\mathbf{X}$.

3. GPU IMPLEMENTATION OF IEA

GPUs can be abstracted in terms of a *stream model*, under which all data sets are represented as streams (i.e., ordered data sets). Algorithms are constructed by chaining so-called *kernels*, which operate on entire streams, 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 of blocks, where each block is composed by a group of threads which share data efficiently through the shared local memory and synchronize their execution for coordinating accesses to memory. The execution model in the GPU can be seen as a set of multiprocesors; in each clock cycle, each processor of the multiprocessor 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 multiprocessor, while the multiprocessors have access to the global GPU (device) memory. With the above issues in mind, we emphasize that the most time consuming step of our IEA algorithm is the calculation of spectral unmixings in iterative fashion as more endmembers become available, as well as the calculation of the reconstructed version $\hat{\mathbf{X}}$ of the original hyperspectral image $\mathbf{X}$ with more endmembers at each iteration. Fortunately the IEA exhibits very few data dependencies within each iteration and each pixel can be processed in parallel. Once the hyperspectral image $\mathbf{X}$ is mapped onto the GPU memory, a structure (`image`) in which the number of blocks equals the number of rows (`num_rows`) in the hyperspectral image and the number of threads equals the number of columns (`num_columns`) is created, thus ensuring that as many pixels as possible are processed in parallel in the considered iteration. The amount of pixels processed in parallel depends of the memory and register resources available in the GPU. These parameters have been carefully optimized in our GPU implementation.

4. EXPERIMENTAL RESULTS

The hyperspectral image scene used in experiments is the well-known Cuprite scene, collected by the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) [2] in the summer of 1997 and available online in reflectance units after atmospheric correction[2]. The portion used in experiments corresponds to a 350×350-pixels subset of the sector labeled as f970619t01p02_r02_sc03.a.rfl in the online data, which comprises 188 spectral bands in the range from 400 to 2500 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[3] will be used in this work for evaluation purposes.

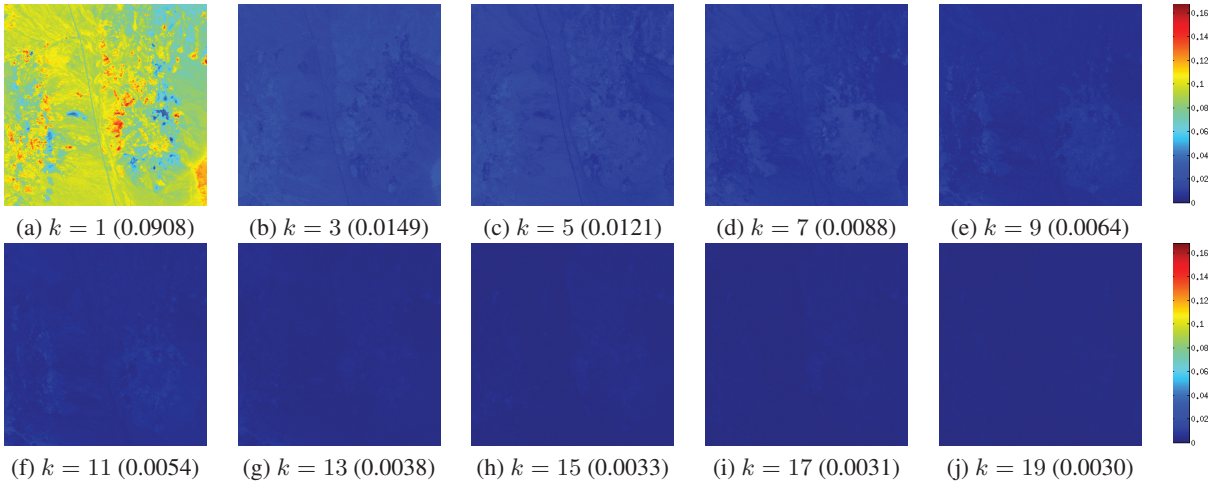
The number of endmembers to be detected was set to $p = 19$ after calculating the virtual dimensionality (VD) [15] of the AVIRIS

[2] http://aviris.jpl.nasa.gov

[3] http://speclab.cr.usgs.gov

Table 1. Spectral angle values (in degrees) between the endmembes extracted by the IEA algorithm and the reference USGS mineral signatures for the AVIRIS Cuprite scene.

Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite	Average
4.81°	4.33°	9.52°	10.76°	5.29°	6.94°



(a) $k = 1$ (0.0908) (b) $k = 3$ (0.0149) (c) $k = 5$ (0.0121) (d) $k = 7$ (0.0088) (e) $k = 9$ (0.0064)

(f) $k = 11$ (0.0054) (g) $k = 13$ (0.0038) (h) $k = 15$ (0.0033) (i) $k = 17$ (0.0031) (j) $k = 19$ (0.0030)

Fig. 1. RMSE values (in the parentheses) between the original and the reconstructed image for different number of iterations, k.

Cuprite image. Table 1 shows the spectral angles (in degrees) between the most similar endmembers extracted by the IEA and the reference USGS spectral signatures. The range of values for the spectral angle is $[0°, 90°]$. As shown by Table 1, the endmembers extracted by the IEA algorithm are very similar, spectrally, to the USGS reference signatures. On the other hand, Figure 1 shows the RMSE values between the original and the reconstructed image [calculated using Eq. (4)] for different number of iterations, k. In each case the compression achieved results from reducing the original n-dimensional hyperspectral image to a compressed version made up of k endmembers and their associated abundance maps, i.e. very high compression ratios are achieved in all cases. As shown by Figure 1, the quality of the reconstruction is already very good for a small number of iterations and for a very high compression ratio. This indicates that, although the proposed compression framework is lossy, most of the relevant information in the original hyperspectral image is retained, particularly in the case in which $k = 19$, for which the $p = 19$ endmembers extracted exhibit good similarity scores with regards to the USGS reference signatures as indicated by the spectral angle values in Table 1.

The GPU platform used to evaluate our GPU implementation is the NVidia™ GeForce GTX 580 GPU[4], which features 512 processor cores operating at 1.544 GHz, with single precision floating point performance of 1,354 Gflops, double precision floating point performance of 198 Gflops, total dedicated memory of 1,536 MB, 2,004 MHz memory (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 8 cores, which uses a motherboard Asus P6T7 WS SuperComputer. Before analyzing the parallel performance of the proposed GPU implementations, we emphasize that our parallel versions provide exactly the same results as the corresponding serial versions, executed in one of the cores of the i7 920 CPU and implemented using the `gcc` (gnu compiler default) of the C programming language (with optimization flag `--O3` to exploit data locality and avoid redundant computations). As a result, the only difference between the serial and parallel versions is the time they need to complete their calculations. The C function GETTIMEOFDAY() was used for timing the CPU implementations, and the CUDA timer was used for the GPU implementations.

Table 2 summarizes the obtained results by the CPU and GPU implementations. The reported GPU times correspond to ten executions in the considered platform for a case study in which $p = 19$ endmembers are extracted, thus reducing the dimensionality of the original hyperspectral data in a ratio given approximately by $188/19 = 9.89$. As shown by Table 2, the measured times were always very similar, with differences –if any– on the order of only a few milliseconds. Table 2 also shows that relevant speedups (above $100x$) were obtained for the IEA algorithm, with very low processing times for the considered case with $p = 19$. Specifically, the cross-track line scan time in AVIRIS, a push-broom instrument [2], is quite fast (8.3 milliseconds to collect 512 full pixel vectors). This introduces the need to process the AVIRIS Cuprite scene in less than 1.98 seconds to fully achieve real-time performance. As noted by Table 2, our processing time in the considered GPU is just 0.68 seconds, well below the real-time processing limit. It should be noted that the GPU implementations have been carefully optimized taking into account the specific parameters of each considered architecture, including the global memory available, the local shared memory in each multiprocessor, and also the local cache memories. Also, we emphasize that the times of the data transfers between CPU and GPU (including the times for loading the image and writing the final results) are included in the GPU times reported on Table 2. Although the obtained results are very encouraging from the viewpoint of obtaining real-time lossy compression results on specialized platforms, we are now experimenting with parallel lossless compression algorithms able to preserve all information in the original hyperspectral data.

[4]http://www.nvidia.com/object/product-geforce-gtx-580-us.html

Table 2. Processing times (seconds) and speedups achieved for ten runs of the GPU implementation of IEA on an NVidia™ GPU GTX 580.

	Time CPU	Time GPU
	69.162401	0.688336
	73.607465	0.686505
	68.839262	0.687528
	67.594444	0.68413
	68.472062	0.683443
	68.112978	0.690085
	67.930827	0.686082
	74.386117	0.687239
	67.617511	0.685526
	67.674788	0.684905
Average time	69.339785	0.686377
Speedup	–	101.0227536

5. REFERENCES

[1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for Earth remote sensing," *Science*, vol. 228, pp. 1147–1153, 1985.

[2] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sensing of Environment*, vol. 65, no. 3, pp. 227–248, 1998.

[3] G. Motta, F. Rizzo, and J. A. Storer, *Hyperspectral data compression*. Berlin: Springer, 2006.

[4] B. Huang, *Satellite data compression*. Berlin: Springer, 2011.

[5] Q. Du and J. E. Fowler, "Low-complexity principal component analysis for hyperspectral image compression," *International Journal of High Performance Computing Applications*, vol. 22, pp. 273–286, 2009.

[6] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 44–57, 2002.

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

[8] A. Plaza, P. Martinez, R. Perez, and J. Plaza, "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 3, pp. 650–663, 2004.

[9] A. Plaza, Q. Du, Y.-L-Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 528–544, 2011.

[10] E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 643–652, 2011.

[11] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, 2011.

[12] A. Plaza, J. Plaza, A. Paz, and S. Sanchez, "Parallel hyperspectral image and signal processing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 119–126, 2011.

[13] S. Sanchez, A. Paz, G. Martin, and A. Plaza, "Parallel unmixing of remotely sensed hyperspectral images on commodity graphics processing units," *Concurrency and Computation: Practice & Experience*, vol. 23, no. 13, pp. 1538–1557, 2011.

[14] J. B. Adams, M. O. Smith, and P. E. Johnson, "Spectral mixture modeling: a new analysis of rock and soil types at the Viking Lander 1 site," *Journal of Geophysical Research*, vol. 91, pp. 8098–8112, 1986.

[15] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Kluwer Academic/Plenum Publishers: New York, 2003.