GPUs versus FPGAs for Onboard Payload Compression of Remotely Sensed Hyperspectral Data

Antonio Plaza, Sergio Sánchez, Abel Paz and Javier Plaza

Hyperspectral Computing Laboratory
Department of Technology of Computers and Communications
Escuela Politécnica de Cáceres, University of Extremadura
Avda. de la Universidad s/n, E-10003 Cáceres, Spain
Email: {aplaza, sersanmar, apazgal, jplaza}@unex.es

ABSTRACT

In this paper, we compare field programmable gate arrays (FPGAs) versus graphical processing units (GPUs) in the framework of (lossy) remotely sensed hyperspectral data compression by developing parallel implementations of a spectral unmixing-based compression strategy on both platforms. For the FPGA implementations, we resort to Xilinx hardware devices certified for on-board operation, while for the GPU implementation we make use of hardware devices available from NVidia, such as the Tesla series. In our comparison, we thoroughly assess the advantages and disadvantages of each considered architecture by designing and evaluating new parallel compression algorithms for both types of platforms. These algorithms are quantitatively evaluated in terms of lossy compression performance using hyperspectral data collected by the NASA's Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) system over the World Trade Center (WTC) in New York, five days after the terrorist attacks that collapsed the two main towers in the WTC complex. Our experimental results indicate that low-weight and low-power integrated components are very appealing to reduce mission payload and obtain analysis/compression results in real-time, thus bridging the gap towards on-board lossy compression of remotely sensed hyperspectral data.

1. INTRODUCTION

Remotely sensed hyperspectral sensors provide image data containing rich information in both the spatial and the spectral domain. The extreme dimensionality of the data introduces the need for compression algorithms which can operate on-board the sensor in time-critical applications that demand a response in real-time, thus reducing the volume of data before it is transmitted to Earth and optimizing the bandwidth of the downlink connection. Specifically, hyperspectral data compression [1] has received considerable interest in recent years due to the enormous data volumes collected by latest-generation imaging spectrometers [2] such as NASA Jet Propulsion Laboratory's Airborne Visible Imaging Infra-Red Imaging Spectrometer (AVIRIS) [3], which is now able to collect hundreds of contiguous spectral bands with high between-band spectral correlation (see Fig. 1). Two approaches have been explored in the literature to address the relevant issue of hyperspectral data compression prior to the analysis: lossless and lossy, in accordance with redundancy removal. Which compression framework should be used depends heavily upon the application domain. Since we are interested in exploitation-based applications, our target hyperspectral data analysis techniques are generally determined by features of objects in the image data rather than the image itself As a result, lossless compression may not offer significant advantages over lossy compression in the sense of feature extraction.

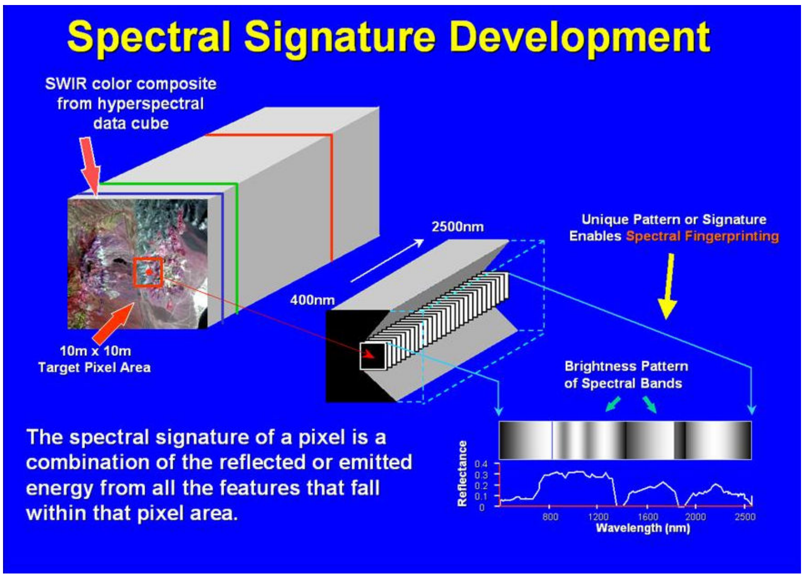


Figure 1. The concept of hyperspectral imaging illustrated using the AVIRIS spectrometer.

The success of a lossy compression technique is generally measured by whether or not its effectiveness meets a preset desired goal which in turn determines which criterion should be used for compression. As an example, principal component analysis (PCA) [4] can be seen as a compression technique that represents data in a few principal components determined by data variances. Its underlying assumption is based on the fact that the data are well represented and structured in terms of variance, where most of data points are clustered and can be packed in a low dimensional space. However, it has been shown that signal-to-noise ratio (SNR) is generally a better measure than data variance to measure image quality in multispectral imagery. Similarly, the mean squared error (MSE) has been also widely used as a criterion for optimality in communications and signal processing such as quantization. However, it is also known that it may not be appropriate to be used as a measure of image interpretation. This is particularly true for hyperspectral imagery which can uncover many unknown signal sources, some of which may be very important in data analysis such as anomalies, small targets which generally contribute very little to SNR or MSE. In the PCA these targets may only be retained in minor components instead of principal components. So, preserving only the first few principal components may lose these targets. In SNR or MSE, such targets may very likely be suppressed by lossy compression if no extra care is taken since missing these targets may only cause inappreciable loss of signal energy or small error.

By realizing the importance of hyperspectral data compression, many efforts have been devoted to design and development of compression algorithms for hyperspectral imagery [1]. Some of these techniques are based on a direct extension of 2D image compression to 3D image compression where many 2D image compression algorithms that have proven to be efficient and effective in 2D images are extended to 3D algorithms. Two techniques of particular interest will be used in our investigation: the JPEG2000 multicomponent [5] and the 3D-SPIHT [6, 7]. Despite a hyperspectral image can be considered as an image cube and therefore processed by the techniques above, our experimental results in previous work [8] show that a direct application of 3D image compression to hyperspectral data may not be able to accurately preserve the very rich spectral information that is present in hyperspectral data cubes. In particular, using MSE or SNR as a compression criterion may result in significant loss of spectral information for data analysis.

Our main focus in this work is to design efficient strategies for reducing significantly the large volume of information contained in the original hyperspectral data cube while, at the same time, being able to retain information that is crucial to deal with small targets (possibly smaller than the pixel size) and anomalies [9]. These two types of pixels are essential in many hyperspectral analysis applications, including military target detection and tracking, environmental modeling and assessment at sub-pixel scales. A subpixel target is a mixed pixel with size smaller than the available pixel size (spatial resolution). So, it is embedded in a single pixel and its existence can only be verified by using the wealth of spectral information provided by hyperspectral sensors. An anomaly is a pixel with a spectral signature which is completely different from those of its surrounding pixels. In both cases, spectral information can greatly help to effectively characterize the substances within the mixed pixel via spectral unmixing techniques [10]. When hyperspectral image compression is performed, it is critical and crucial to take into account these two issues, which have been generally overlooked in the development of lossy compression techniques in the literature.

In this work, we address the aforementioned issue by designing spectral unmixing-based lossy hyperspectral data compression techniques and efficiently implementing them on specialized hardware devices intended for onboard exploitation, such as field programmable gate arrays (FPGAs) [11] and graphical processing units (GPUs) [12]. These platforms offer significant advantages in terms of mission payload, mainly due to their low weight and compact size, and to their capacity to provide high performance computing at low costs. For instance, FPGAs offer the appealing possibility of being able to adaptively select the data processing algorithm to be applied (out of a pool of available algorithms) from a control station on Earth, immediately after the data is collected by the sensor. This feature is possible thanks to the inherent reconfigurability of FPGA devices, which are generally more expensive than GPU devices. In the near future, significant developments are expected in the active research area devoted to radiation-hardening of GPU and FPGA devices, which may allow their full incorporation to satellite-based Earth and planetary observation platforms in space.

The remainder of the paper is organized as follows. Section 2 describes the proposed linear spectral mixture-based lossy compression technique. Sections 3 and 4 respectively develop parallel versions of the proposed algorithms for efficient implementation in FPGAs and GPUs. Section 5 provides experimental results comparing the performance of the proposed techniques with regards to existing ones in terms of both information preservation after compression and parallel performance. Finally, section 6 concludes with some remarks and hints at future research lines.

2. SPECTRAL UNMIXING-BASED LOSSY COMPRESSION OF HYPERSPECTRAL DATA

This section describes an application-oriented, lossy compression algorithm for hyperspectral image data which relies on the utilization of the linear mixture model [13]. The idea of the proposed lossy data compression algorithm is to represent a hyperspectral image cube by a set of fractional abundance images (see Fig. 2). Let us assume that a remotely sensed hyperspectral scene with N bands is denoted by $\mathbf{F}$, in which a pixel at discrete spatial coordinates is represented by a vector $\mathbf{f} = [f_1, f_2,\ldots, f_N]$, where f_k denotes the spectral response at the k-th sensor wavelength, with $k = 1, 2, \ldots, N$. Under the linear mixture model assumption, each pixel vector in the original scene can be modeled using the following expression:

$$\mathbf{f} = \sum_{i=1}^{E} a_i \cdot \mathbf{e}_i + \mathbf{t}, \quad (1)$$

where $\mathbf{e}_i$ designates the i-th pure spectral component (endmember) residing in the pixel, a_i is a scalar value designating the fractional abundance of the endmember $\mathbf{e}_i$ at the pixel $\mathbf{f}$, E is the total number of endmembers, and $\mathbf{t}$ is a noise vector. The solution of the linear spectral mixture problem described in (1) and adopted in our work relies on the correct determination of a set of endmembers and their correspondent abundance fractions at each pixel $\mathbf{f}$ (possibly after a dimensionality reduction transformation to speed up the procedure for searching for E endmembers in the data). Two physical constrains are generally imposed into the model described in (1), these are the abundance non-negativity constraint (ANC) and the abundance sum-to-one constraint (ASC). By using the linear mixture model above, we can represent each pixel vector $\mathbf{f}$ by its associated, linearly derived abundance vector $\mathbf{f} = [f_1, f_2,\ldots, f_E]$ with E dimensions, which can be seen as a fingerprint of $\mathbf{f}$ with regards to E endmembers which are extracted from the original input scene by a fully automatic algorithm. With this idea in mind, in order to compress the original hyperspectral image (in lossy fashion) we would just need to retain the extracted E endmembers and their associated E fractional abundance maps (see Fig. 2). Since the number of endmembers in a hyperspectral image, E, is generally much higher than the number of spectral bands, N, significant compression rations can be achieved by this simple lossy compression strategy.

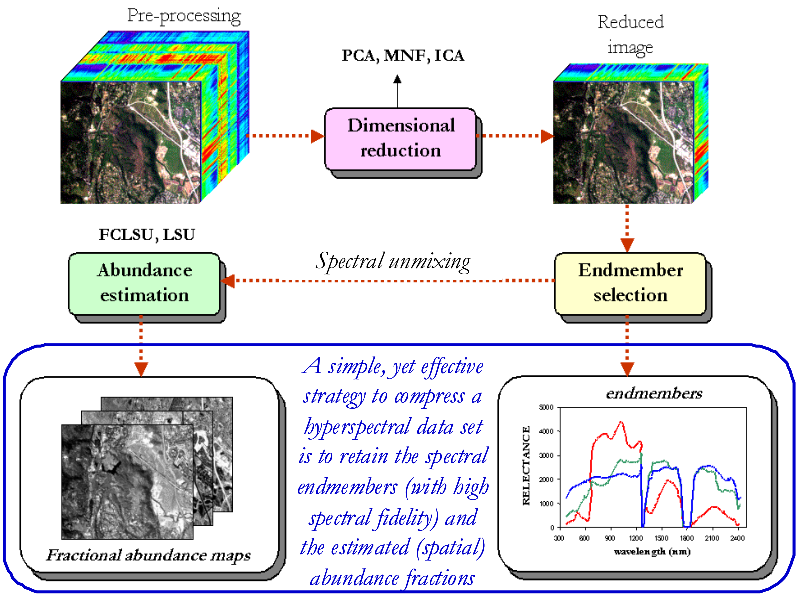


Figure 2. The proposed lossy hyperspectral data compression strategy using spectral unmixing concepts.

In order to accomplish the endmember extraction part of the proposed lossy compression strategy, one of the most successful algorithms for automatic endmember extraction in the literature has been the PPI algorithm [14]. The algorithm proceeds by generating a large number of random, N-dimensional unit vectors called "skewers" through the dataset. Every data point is projected onto each skewer, and the data points that correspond to extrema in the direction of a skewer are identified and placed on a list. As more skewers are generated, the list grows, and the number of times a given pixel is placed on this list is also tallied. The pixels with the highest tallies are considered the final endmembers. The concept of the PPI algorithm is explained in Fig. 3. On the other hand, for the abundance estimation part of the lossy compression algorithm we use the fully constrained linear spectral unmixing (FCLSU) algorithm in [15], which includes the desired ASC and ANC constraints. The full compression strategy is illustrated in pseudo-code in Fig. 4.

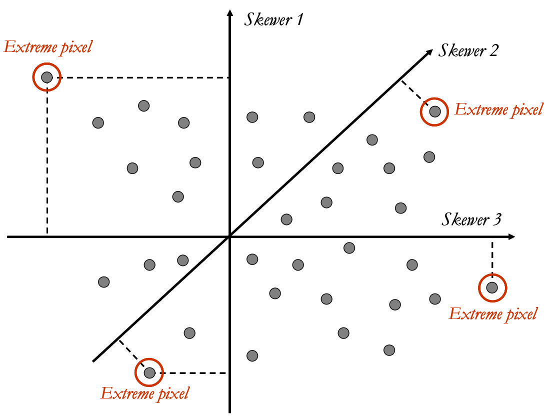


Figure 3. Toy example illustrating the behaviour of the pixel purity index (PPI) algorithm in a 2-dimensional space.

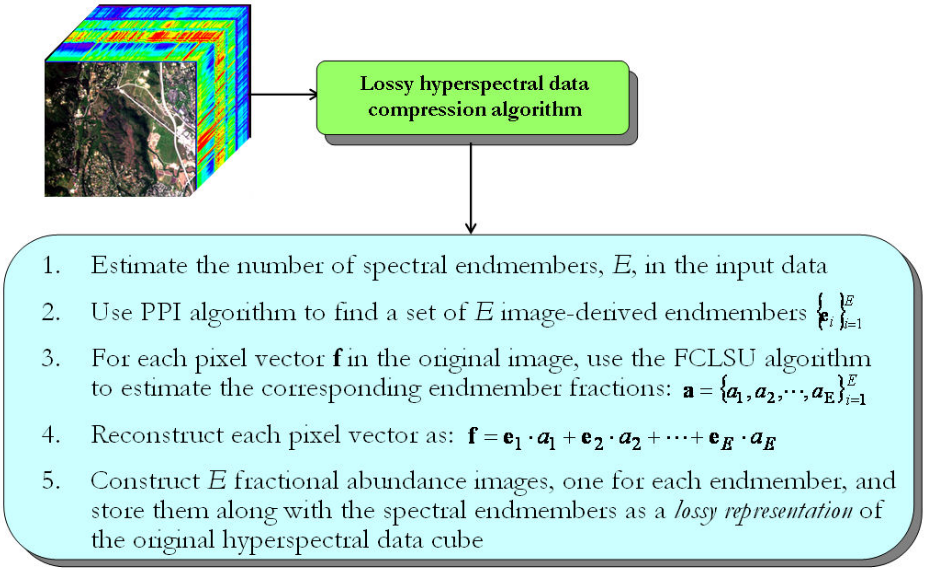


Figure 4. Pseudo-code of the proposed lossy hyperspectral data compression strategy.

3. FPGA IMPLEMENTATION

Our strategy for implementation of the hyperspectral unmixing chain in reconfigurable hardware is aimed at enhancing replicability and reusability of slices in FPGA devices through the utilization of systolic array design [16]. Fig. 5 describes our systolic architecture. Here, local results remain static at each processing element, while a total of T pixel vectors with N dimensions are input to the systolic array from top to bottom. Similarly, K skewers with N dimensions are fed to the systolic array from left to right. In Fig. 5, asterisks represent delays. The processing nodes labeled as *dot* in Fig. 5 perform the individual products for the skewer projections. On the other hand, the nodes labeled as *max* and *min* respectively compute the maxima and minima projections after the dot product calculations have been completed. In fact, the *max* and *min* nodes avoid broadcasting the pixel while simplifying the collection of the results. Based on the systolic array in Fig. 5 (which also allows implementation of the fully constrained spectral unmixing stage) we have implemented the full hyperspectral unmixing chain using the very high speed integrated circuit hardware description language (VHDL) for the specification of the systolic array. Further, we have used the Xilinx ISE environment and the Embedded Development Kit (EDK) environment to specify the complete system. The full system has been ported to a low-cost reconfigurable board of XUPV2P type with a single Virtex-II PRO xc2vp30 FPGA component.

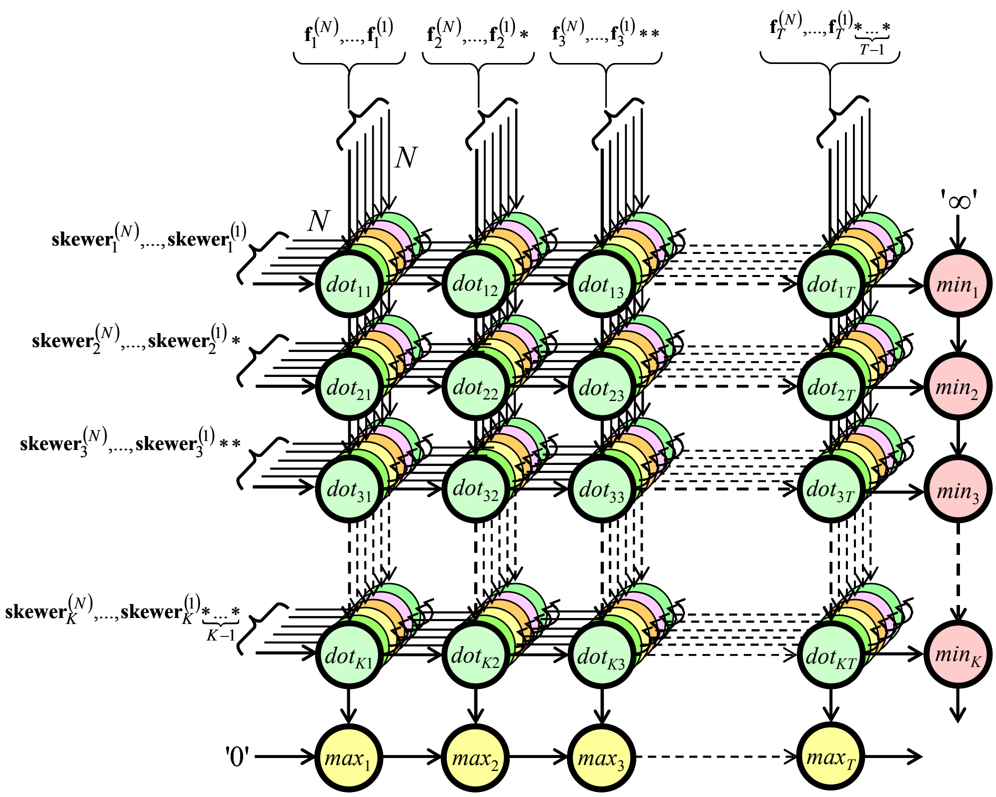


Figure 5. Systolic array architecture used in our FPGA hardware implementation.

Fig. 6 shows the architecture of the hardware used to implement the hyperspectral unmixing chain. For data input we use the FPGA memory slot (of DDR2 SDRAM type) which holds up to 2 Gigabytes, and a direct memory access (DMA) module (controlled by a PowerPC) with a write queue to store the pixel data. A read queue and a transmitter are also used to send the endmembers to the FPGA via a RS232 port. A control unit, the systolic array, and a module for random generation of skewers are also implemented. Additional details are given in [17].

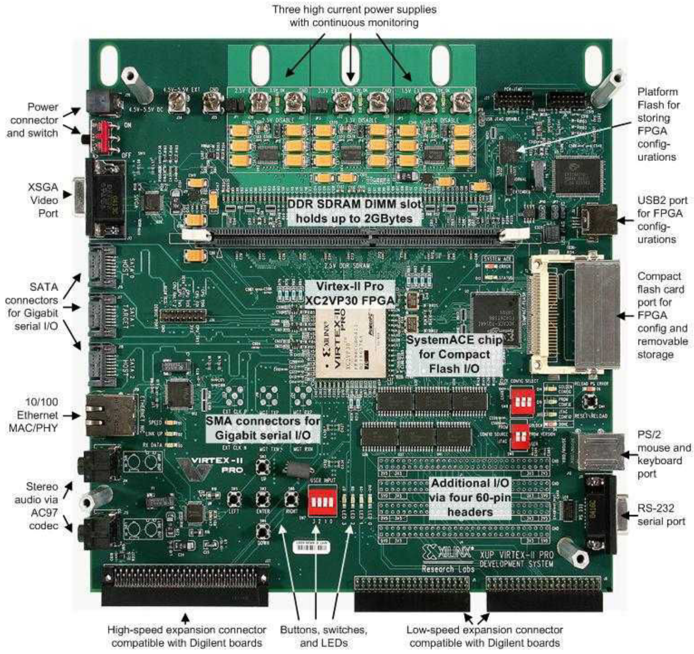


Figure 6. FPGA board used in our experiments.

4. GPU IMPLEMENTATION

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. In order to implement the considered hyperspectral unmixing chain in a GPU, the first issue that needs to be addressed is how to map a hyperspectral image onto the memory of the GPU. Since the size of hyperspectral images may exceed the capacity of the GPU memory, in that case we split the image into multiple spatial-domain partitions [16] made up of entire pixel vectors. Fig. 7 shows a flowchart describing the kernels that comprise our GPU-based implementation, which has been developed using the compute unified device architecture (CUDA).

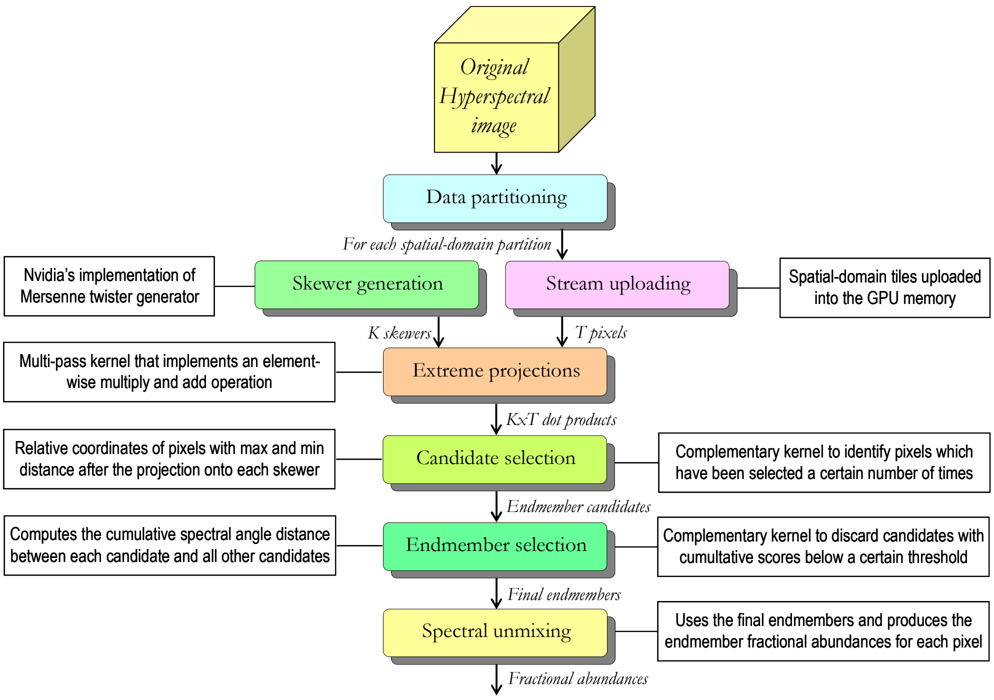


Figure 7. Flowchart summarizing the kernels involved in our GPU implementation.

5. EXPERIMENTAL RESULTS

This section evaluates the effectiveness of the proposed hyperspectral data compression technique. Subsection 5.1 describes the hyperspectral data sets used for evaluation purposes. Subsection 5.2 describes the parallel computing architectures (FPGA and GPU). A survey on algorithm performance in a real application domain is then provided in subsection 5.3, along with a discussion on the advantages and disadvantages of the proposed lossy compression technique in terms of hyperspectral data fidelity after compression. Finally, subsection 5.4 provides an evaluation of parallel performance of the proposed high performance implementations.

5.1. Hyperspectral data set

The image scene used for experiments in this work was collected by the AVIRIS instrument, which was flown by NASA's Jet Propulsion Laboratory over the World Trade Center (WTC) area in New York City on September 16, 2001, just five days after the terrorist attacks that collapsed the two main towers and other buildings in the WTC complex. The data set selected for experiments was geometrically and atmospherically corrected prior to data processing, and consists of 614x512 pixels, 224 spectral bands and a total size of 140 MB. The spatial resolution is 1.7 meters per pixel. The leftmost part of Fig. 8 shows a spectral band of the data set selected for experiments, in which a detail of the WTC area is shown in a rectangle. The rightmost part of Fig. 8 shows a thermal map generated by U.S. Geological Survey and centered at the region where the buildings collapsed. The map displays the locations of the thermal hot spots with temperatures ranging from 700F (marked as “F”) to 1300F (marked as “G”). This thermal map will be used in this work as ground-truth to substantiate the capacity of the proposed compression algorithm to retain the sub-pixel spectral information required to detect the hot spot fires in the WTC complex.

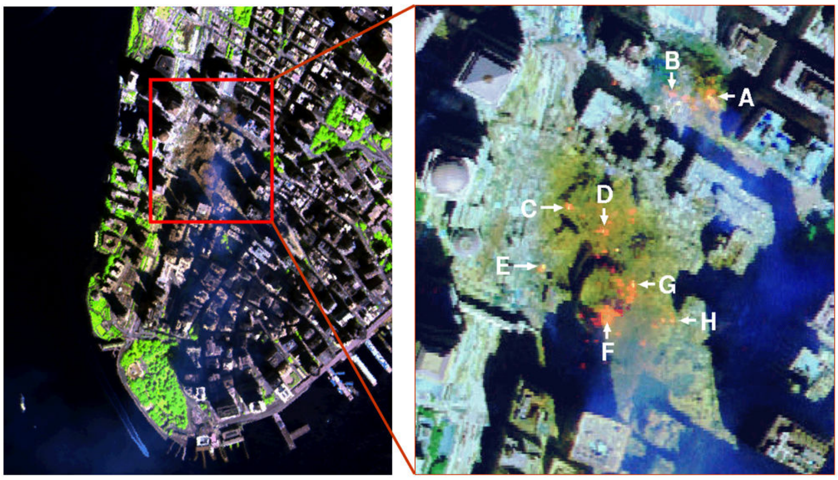


Figure 8. AVIRIS image collected over World Trade Center (left). Location of thermal hot (right).

5.2. Parallel computing architectures

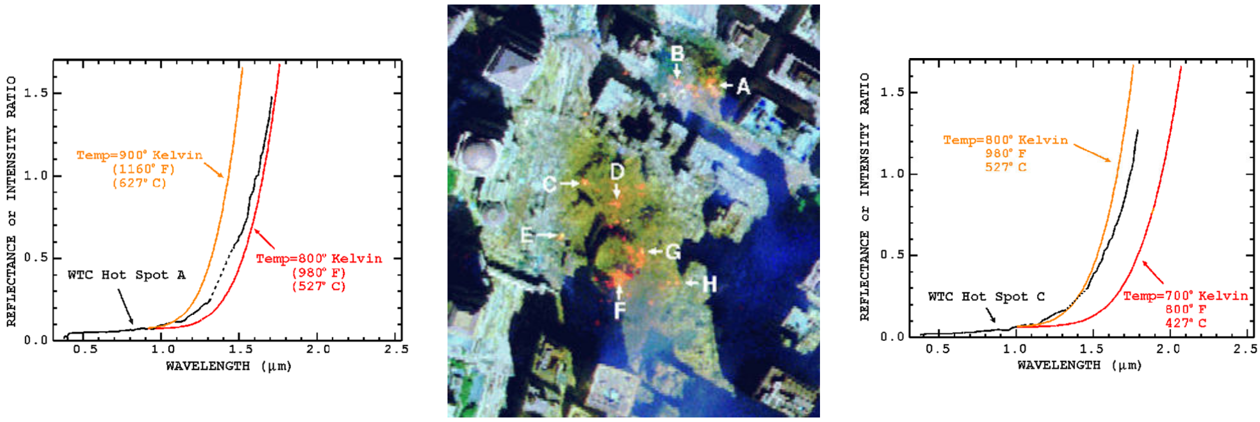
Two specialized hardware architectures have been considered in this work. The first one is a low-cost reconfigurable board with a single Virtex-II PRO xc2vp30 FPGA component, a DDR SDRAM DIMM memory slot with 2 Gigabytes of main memory, a RS232 port, and some additional components not used by our implementation (see Fig. 6). This board is very similar to other Xilinx designs which have been certified for aerospace operation. We also use an NVidia Tesla C1060 GPU, which features 240 processor cores operating at 1.296 Ghz, with single precision floating point performance of 933 Gigaflops, double precision floating point performance of 78 Gflops, total dedicated memory of 4 GB, 800 MHz memory (with 512-bit GDDR3 interface) and memory bandwidth of 102 GB/sec. 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.

5.3. Experimental assessment of data fidelity after compression

Prior to a full examination and discussion of compression results, it is important to outline parameter values used for the different stages of the proposed hyperspectral lossy compression technique. Specifically, the number of endmembers to be extracted in the WTC data was set to E=20 after estimating the intrinsic dimensionality of the data using the *virtual dimensionality* concept [18]. In addition, the number of skewers was set to $K=10^4$ (although values of $K=10^3$ and $K=10^5$ were also tested, we experimentally observed that the use of $K=10^3$ resulted in the loss of important endmembers, while the endmembers obtained using $K=10^5$ were essentially the same as those found using $K=10^4$). These parameter values are in agreement with those used before in the literature [16].

In order to explore the degree of spectral fidelity of the compressed images obtained after applying the proposed compression method, Fig. 9 reports the spectral angle values among the pixels labeled as hot spots and the pixels at the same spatial locations in the images obtained after applying the proposed lossy hyperspectral data compression technique (in the table, the lowest the scores, the highest the spectral similarity). In experiments, compression ratios of

20:1, 40:1 and 80:1 (given by different tested values of input parameter *E*) were used. For illustrative purposes, we have also included the results provided by two standard methods in our comparison, i.e., the wavelet-based JPEG2000 multicomponent [6] and the 3D-SPIHT [7]. The JPEG2000 implementation used for our experiments was the one available in kakadu software. Both techniques are 3D compression algorithms that treat the hyperspectral data as a 3D volume, where the spectral information is the third dimension. As expected, as the compression ratio was increased, the quality of extracted endmembers was decreased. Results in Fig. 9 show that, for the same compression ratio, a 3D lossy compression algorithm may result in significant loss of spectral information which can be preserved better, in turn, by an application-oriented algorithm such as the proposed lossy compression approach. It should be noted that all the algorithms reported on Table 1 required several minutes of computation to compress the AVIRIS WTC data set in a PC with AMD Athlon 2.6 GHz processor and 512 MB of RAM (specifically, the proposed lossy compression algorithm required almost one hour).



Hot spot	Lat. (North)	Long. (West)	Temp. (Kelvin)	JPEG2000			3D-SPIHT			Proposed		
				20:1	40:1	80:1	20:1	40:1	80:1	20:1	40:1	80:1
'A'	40°42'47.18"	74°00'41.43"	1000	0.114	0.125	0.136	0.107	0.112	0.124	0.065	0.082	0.092
'B'	40°42'47.14"	74°00'43.53"	830	0.124	0.131	0.140	0.109	0.113	0.129	0.058	0.077	0.094
'C'	40°42'42.89"	74°00'48.88"	900	0.098	0.117	0.129	0.106	0.111	0.126	0.068	0.079	0.093
'D'	40°42'41.99"	74°00'46.94"	790	0.135	0.144	0.151	0.115	0.123	0.138	0.073	0.082	0.099
'E'	40°42'40.58"	74°00'50.15"	710	0.123	0.139	0.150	0.110	0.119	0.141	0.071	0.080	0.090
'F'	40°42'38.74"	74°00'46.70"	700	0.116	0.128	0.146	0.099	0.110	0.123	0.055	0.069	0.083
'G'	40°42'39.94"	74°00'45.37"	1020	0.119	0.133	0.149	0.103	0.120	0.128	0.062	0.071	0.088
'H'	40°42'38.60"	74°00'43.51"	820	0.130	0.144	0.152	0.119	0.131	0.145	0.069	0.078	0.091

Figure 9. Spectral angle-based similarity scores between the original hyperspectral image pixels labeled as hot spots in the AVIRIS scene over the World Trade Center and the corresponding pixels in the hyperspectral images obtained after compression using different techniques.

5.4. Parallel performance

Our implementation of the lossy hyperspectral data compression algorithm on the considered FPGA board was able to achieve a total processing time for the considered AVIRIS scene of 31.23 seconds utilizing approximately 76% of the available hardware resources in the considered board. This FPGA has a total of 13696 slices (out of which 10423 were used by our implementation). An interesting feature of the considered design is that it can be scaled without significantly increasing the delay of the critical path (the clock cycle remained constant at 187 MHz). It should also be noted that in our implementation he have paid special attention to the impact of communications. In previous designs [16], the module for random generation of skewers was situated in an external processor. Hence, frequent communications between the host (CPU) and the device (FPGA) were needed. However, in our implementation the hardware random generation module is implemented internally in the FPGA board. This approach significantly reduced the communications, leading to increased parallel performance.

On the other hand, the execution time for the lossy hyperspectral compression algorithm implemented on the NVidia C1060 Tesla GPU was 17.59 seconds. In our experiments on the Intel Core i7 920 CPU, the hyperspectral unmixing chain took 1078.03 seconds to process the considered AVIRIS scene (using just one of the available cores). This means that the speedup achieved by our GPU implementation with regards to the serial implementation in one core was approximately 61.28. After profiling the implementation using the CUDA Visual Profiler tool, we checked that the *skewer projections* kernel consumed more than 95% of the total GPU time, while the one used for random generation of skewers comprratively occupied a much smaller fraction. Finally, the data movement operations were not significant, which indicates that most of the GPU processing time is invested in the most time-consuming operation, i.e. the calculation of skewer projections and identification of maxima and minima projection values leading to endmember identification with *E*=20, which is indeed a crucial step for defining the compression ratio in our proposed lossy compression algorithm.

6. CONCLUSIONS AND FUTURE RESEARCH LINES

We have inter-compared FPGAs versus GPUs in the role of compressing hyperspectral data (in lossy fashion) using spectral unmixing concepts. The proposed implementations are intended for on-board analysis (before the hyperspectral data is transmitted to Earth). A major goal is to overcome an existing limitation in many remote sensing and observatory systems: the bottleneck introduced by the bandwidth of the downlink connection from the observatory platform. In this regard, our experimental results demonstrate that the proposed hardware implementations make appropriate use of computing resources in FPGA and GPU architectures, and further provide adequate compromises in terms of on-board payload data compression. However, it should be noted that the achieved response times are not in real time. Specifically, the cross-track scan line of AVIRIS, a pushbroom instrument, is 8.3 msec to collect 512 pixel vectors, and this introduces the need to process a standard data set of 614x512 pixels in less than 5 sec to fully achieve real-time performance. However, the measured times can still be improved and are actually believed to be acceptable in most remote sensing applications. As a result, the reconfigurability of FPGA systems on the one hand, and the low cost of GPU systems on the other, open many innovative perspectives, ranging from the appealing possibility of being able to adaptively select one out of a pool of available data compression algorithms (which could be applied on the fly aboard the airborne/satellite platform, or even from a control station on Earth), to the possibility of providing near real-time response. Flexibility is also important parameter to be discussed in aerospace applications, since the available space is very restricted. In this regard, both platforms offer a low-weight solution compliant with mission payload requirements, although power consumption is higher in the GPU solution than in the FPGA solution. In both cases the solutions can be easily reused to deal with other different processing problems (e.g. lossless compression). In the GPU case, this is as simple as executing a different code, whereas in the FPGA case this can be achieved by reconfiguring the platform with an adequate algorithm. Although the experimental results presented in this work are quite encouraging, further work is still needed to optimize the proposed implementations and fully achieve real-time performance. Radiation-tolerance and power consumption issues for these hardware accelerators should be explored in more detail in future developments (particularly for GPUs, since many FPGAs have already been certified for aerospace operation).

REFERENCES

[1] G. Motta, F. Rizzo, and J. A. Storer, *Hyperspectral data compression*, Springer-Verlag, New York, 2006.

[2] 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.

[3] R. O. Green, “Imaging spectroscopy and the airborne visible-infrared imaging spectrometer (AVIRIS),” *Remote Sensing of Environment*, vol. 65, pp. 227–248, 1998.

[4] J. A. Richards, “Analysis of remotely sensed data: the formative decades and the future,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 422–432, 2005.

[5] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standard and practice*, Kluwer, Boston, 2002.

[6] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.

[7] B.-J. Kim, Z. Xiong, andW. A. Pearlman, “Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT),” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 1374–1387, 2000.

[8] A. Plaza, “Clusters versus FPGAs for spectral mixture analysis-based lossy hyperspectral data compression,” *Proceedings of SPIE, Satellite Data Compression, Communication and Processing Conference V*, vol. 7084, 2008.

[9] A. Paz and A. Plaza, “Clusters versus GPUs for parallel automatic target detection in remotely sensed hyperspectral images,” *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Article ID 915639, 18 pages, 2010.

[10] Q. Du and C.-I. Chang, “Linear mixture analysis-based compression for hyperspectral image analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, pp. 875–891, 2004.

[11] A. Plaza, “Towards real-time compression of hyperspectral images using Virtex-II FPGAs,” *Lecture Notes in Computer Science*, vol. 4641, pp. 248-257, 2007.

[12] J. Setoain, M. Prieto, C. Tenllado, A. Plaza and F. Tirado, “Parallel morphological endmember extraction using commodity graphics hardware,” *IEEE Geoscience and Remote Sensing Letters*, vol. 43, no. 3, pp. 441-445, 2007.

[13] 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.

[14] J. W. Boardman, “Automating spectral unmixing of aviris data using convex geometry concepts,” *Summaries of Airborne Earth Science Workshop*, R. O. Green, ed., JPL Publication 93-26, pp. 111–114, 1993.

[15] D. Heinz and C.-I. Chang, “Fully constrained least squares linear mixture analysis for material quantification in hyperspectral imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, pp. 529–545, 2001.

[16] A. Plaza and C.-I. Chang, *High performance computing in remote sensing*, CRC Press, Boca Raton, 2006.

[17] C. Gonzalez, J. Resano, D. Mozos, A. Plaza and D. Valencia, “FPGA implementation of the pixel purity index algorithm for remotely sensed hyperspectral image analysis,” *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Article ID 969806, 13 pages, 2010.

[18] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*, Kluwer Academic and Plenum Publishers, New York, 2003.