Low Power Compressive Sensing for Hyperspectral Imagery

José M. P. Nascimento
Instituto de Telecomunicações, and
Instituto Superior de Engenharia de Lisboa, IPL
Lisbon, Portugal
zen@isel.pt

Mário Véstias
Instituto Superior de Engenharia de Lisboa, IPL, and
INESC-ID, Lisbon, Portugal
Lisbon, Portugal
mvestias@deetc.isel.pt

***Abstract*—Hyperspectral imaging instruments allow remote Earth exploration by measuring hundreds of spectral bands at very narrow channels of a given spatial area. The resulting hyperspectral data cube typically comprises several gigabytes. Such extremely large volumes of data introduces problems in its transmission to Earth due to limited communication bandwidth. As a result, the applicability of data compression techniques to hyperspectral images have received increasing attention.**

This paper, presents a study of the power and time consumption of a parallel implementation for a spectral compressive acquisition method on a Jetson TX2 platform. The conducted experiments have been performed to demonstrate the applicability of these methods for onboard processing. The results show that by using this low energy consumption GPU and integer data type is it possible to obtain real-time performance with a very limited power requirement while maintaining the methods accuracy.

***Index Terms*—Hyperspectral imagery, Compressive Sensing, Embedded Systems**

I. Introduction

Hyperspectral remote sensing extracts information from objects or scenes lying on the Earth surface, based on their radiance acquired by airborne or spaceborne sensors. This information is collected in hundreds of images representing the radiance collected in each spectral band. These bands offers very significant potential in the identification of materials and their properties [1]. This high spectral resolution has enabled the use of hyperspectral imagery in the fields of urban and regional planning, water resource management, environmental monitoring, oil spill and other types of chemical contamination, and target detection for military and security purposes [1].

Due to the significantly improved spectral resolution provided by the latest generation of hyperspectral sensors, hyperspectral images are extremely large, introducing the need for compression methods that can operate on-board [2], [3]. Additionally, for certain applications that demand a real-time response this compression is of paramount importance since it reduces the volume of data to be transmitted to the Earth, optimizing the limited bandwidth available for downlink. Conventional compression schemes are not suited for this purpose since it first acquire the full data set and then implement some compressing technique [4].

Recently, hyperspectral compressive sensing (CS) has received considerable interest, both in terms of hardware and signal processing algorithms [5]–[8]. These algorithms are able to reconstruct the original hyperspectral image from a low number of random projections in the spectral domain. This is only possible if the spectral vectors live in a low dimensional subspace, which is a very good approximation in most hyperspectral images (HSIs) of the real world, namely when the observed spectral vectors are well approximated by linear mixing model (LMM) [1], [9]. In this way, CS algorithms are able to reconstruct the original hyperspectral image with a number of measurements per pixel in the order of the size of this subspace, which typically is much lower than the number of bands of the sensor [10].

The possibility of real-time, onboard data compression is a highly desirable feature to overcome the problem of transmitting a sheer volume of high-dimensional data to Earth control stations via downlink connections [11]. Usually compression algorithms comes with higher computational complexity. Moreover, these algorithms are usually implemented in standard PCs which cannot be easily employed for onboard processing due to its weight, heat dissipation and energy consumption issues. To alleviate the computational burden it is desirable to implement such methods in parallel. Recently, graphics computing units (GPUs) has become a topic of considerable interest due to their extremely high floating-point processing performance, huge memory bandwidth and their comparatively low cost. There are a few methods that have been implemented on this platforms [12], [13]. In particular, GPUs may be suitable in the future for real-time onboard processing due to their portability, although the current energy consumption of these devices still makes them not totally appealing for spacebased Earth observation missions from satellites (the high energy consumption affects mission payload) [11].

Over the last years, the advances in semiconductor industry and the huge interest on developing mobile devices have allowed companies such as Nvidia to develop low power GPUs like the Jetson TX2, which is a low energy consumption GPUs, that nevertheless, can achieve high throughput in image

This work was supported in part by the Instituto de Telecomunicações and in part by the Portuguese Science and Technology Foundation under Project UID/EEA/50008/2019.

processing applications at the same time.

This paper explores the possibility to use these low power GPUs to perform the Hyperspectral CS process as an alternative to traditional compression and dimensionality reduction algorithms performed on common GPU boards. In section II, a CS method called Hyperspectral Coded Aperture method (HYCA) is summarized, which have been chosen in this work as CS algorithm to demonstrate the performance of such methods on low power hardware. In section III, a brief description of the GPU architecture and the main features of the Jetson TX2 hardware used on the experiments is provided. In section IV, a set of experiments are conducted to demonstrate the effectiveness of this hardware, performing the random projections in real-time. Finally, section V presents some conclusions and future lines of work.

II. Compressive Sensing Method

In this section a CS method named Hyperspectral Coded Aperture (HYCA) [5] is briefly described. This approach compresses the data on the acquisition process, then the compressed signal is sent to Earth and stored in compressed form. Later the original signal can be recovered by taking advantage of the fact that the hyperspectral data can be explained using a reduced set of spectral endmembers due to the mixing phenomenon [10] and also exploits the high spatial correlation of the fractional abundances associated to the spectral endmembers. This method for its characteristics is well suited to be developed in a parallel fashion [12].

Let $\mathbf{x} \in \mathbb{R}^{n_b \times n_p}$ represent, in vector format, a hyperspectral image with n_b spectral bands and $n_p := n_r \times n_c$ pixels where n_r and n_c denote, respectively, the number of rows and columns of the hyperspectral image in the spatial domain. The ordering of $\mathbf{x}$ correspond to all image pixels for each spectral band. In order to perform the compression of the original signal $\mathbf{x}$, and as in [5], for each pixel $i \in \{1, \ldots, n_p\}$, a set of q inner products between $\mathbf{x}_i$ and samples of i.i.d. Gaussian random vectors is performed. The total number of measurements is therefore qn_p yielding an undersampling factor of q/n_b. This measurement operation can be represented as a matrix multiplication $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a block diagonal matrix containing the matrices $\mathbf{A}_i \in \mathbb{R}^{q \times n_b}$ acting on the pixel $\mathbf{x}_i$, for $i \in \{1, \ldots, n_p\}$. Since the image can be very large, measurement strategy splits the dataset into different windows of size $m = ws \times ws$ and then repeat the matrices $\mathbf{A}_i$ used in each window, thus requiring to store in memory just m different $\mathbf{A}_i$ matrices.

Let us now define the linear operator $\mathbf{x} = (\mathbf{I} \otimes \mathbf{E})\mathbf{z}$, where the matrix $\mathbf{E}$ represents the basis of the subspace where the data lives [10], $\mathbf{I}$ is the identity matrix, and the vector $\mathbf{z}$ contains the coefficients. In this work, the $\mathbf{E}$ matrix contains the p endmembers of the data set by columns obtained in a very fast way through the vertex component analysis (VCA) algorithm [9], thus $\mathbf{z}$ contains the fractional abundances associated to each pixel.

Let us now assume that $\mathbf{K} = \mathbf{H}(\mathbf{I} \otimes \mathbf{E})$. If matrices $\mathbf{E}$ and $\mathbf{H}$ are available, one can formulate the estimation of $\mathbf{z}$ with from q-dimensional vector of measurements. Since the fractional abundances in hyperspectral images exhibit a high spatial correlation, we exploit this feature for estimating $\mathbf{z}$ using the following optimization problem:

$$\min_{\mathbf{z} \geq \mathbf{0}} (1/2)\|\mathbf{y} - \mathbf{K}\mathbf{z}\|^2 + \lambda_{TV}\mathrm{TV}(\mathbf{z}). \quad (1)$$

Therefore, the minimization (1) aims at finding a solution which is a compromise between the fidelity to the measured data, enforced by the quadratic term $(1/2)\|\mathbf{y} - \mathbf{K}\mathbf{z}\|^2$, and the properties enforced by the TV regularizer, that is piecewise smooth image of abundances. The relative weight between the two characteristics of the solution is set the regularization paremeter $\lambda_{TV} > 0$.

To solve the convex optimization problem in Eq. (1), a set of new variables per term of the objective function were used and the ADMM methodology [14] has been adopted to decompose very hard problems into a cyclic sequence of simpler problems (see work [5] for more details).

III. GPU arquitecture

In recent years, graphics processing units (GPUs) have evolved into highly parallel and programmable systems [12]. Specifically, several hyperspectral imaging algorithms have shown to be able to benefit from this hardware taking advantage of the extremely high floating-point processing performance, compact size, huge memory bandwidth, and relatively low cost of these units, which make them appealing for onboard data processing, specially for CS applications, due to the fact that the random projection process typically involves Matrix-Vector products which are a perfect match for the GPU architecture [12]. However, one of the main problems for the use of this hardware onboard is the high power and energy consumption that they require. Remote sensing missions frequently perform the bulk of data processing and storage onboard of airborne devices and satellites, which may impose severe constraints on the power and energy consumption (e.g., due to battery life time or electricity being produced by the attached solar panels). The combination of the high dimensionality of hyperspectral images, very demanding processing methods and energy restrictions justifies the exploration of high-performance yet low-power technologies together with energy-aware novel computational algorithms that can produce a response in real time or near real time while minimizing the energy usage.

In this work, the parallel development of the coder side of HYCA method on a low consumption GPU such as the Jetson TX2 is explored. The TX2 employs an SOC (system-on-chip) design that incorporates a quad-core 2.0-GHz 64-bit ARMv8 A57 processor, a dual-core 2.0-GHz superscalar ARMv8 Denver processor, and an integrated Pascal GPU. There are two 2-MB L2 caches, one shared by the four A57 cores and one shared by the two Denver cores. The GPU has two streaming multiprocessors (SMs), each providing 128 1.3-GHz cores that share a 512-KB L2 cache. The six CPU cores and integrated GPU share 8 GB of 1.866-GHz DRAM memory

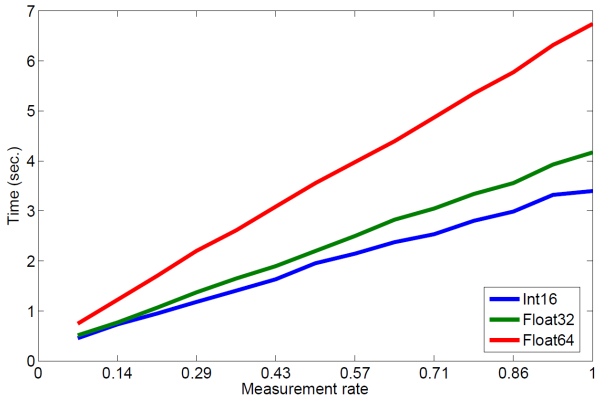

Fig. 1. Processing times in seconds for the Jetson TX2 hardware for different measurement ratios using different data types: float64, float32, and int16.

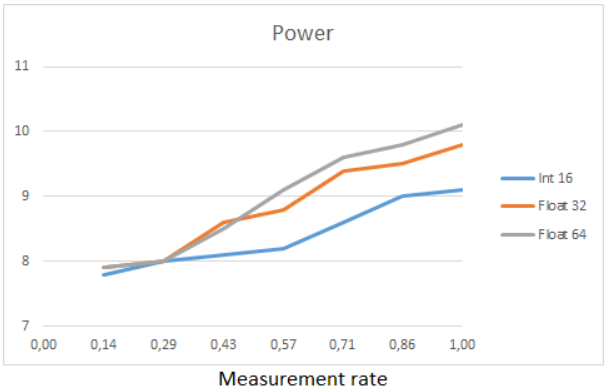

Fig. 2. Average power in Watts (W) for different measurement ratios using different data types: float64, float32, and int16.

[15]. The Jetson TX2 typically draws between 7.5 and 15 watts with a voltage input of 5.5V-19.6V DC and requires minimal cooling and additional space.

CUDA architecture has been used for implementing the parallel HYCA (P-HYCA) measurement process. The CUDA programming model is designed to develop applications which scales the parallelism in a transparently way, independently from the number of processors or multiprocessors of the hardware. In order to do that, CUDA defines the so called kernels, which are functions to be processed in parallel. For each kernel, the user structures the parallelism into a grid of blocks, each one processed by a GPU multiprocessor. Each block is divided into threads, which are the smallest processing units in the architecture. The different threads may synchronize with the other threads of the same block and communicate with them through the shared memory of the multiprocessor. However the different blocks of the grid are executed in parallel with any sort of synchronization. Specifically in the implementation of P-HYCA each thread perform the measurement process of one pixel, thus, each thread performs the operation $\mathbf{y}_i = \mathbf{A}_j\mathbf{x}_i$ with $i \in 1 \dots n_p$ and $j \in 1 \dots m$. The number of threads per block is the maximum allowed by the architecture, which in the case of the Jetson TX2 is 1024. Due to the fact that many threads in the block uses the same $\mathbf{A}_j$ matrices, these matrices are loaded into shared memory before processing the dot products, in this way we reduce the memory accessed required to global memory.

IV. Experimental Results

In this section a series of experiments are conducted in order to evaluate the performance and power requirements of the Jetson TX2 on performing the measurement process of the CS method. Herein, we focus on the coder-side of the method since it is the one to be performed onboard, while the decoder side may be performed on the Earth station where plenty of hardware may be available. Power is measured with a power meter connected between the energy socket and the board.

The simulated data set used in this experiments was generated from spectral signatures randomly selected from the United States Geological Survey (USGS)[1]. The dataset size is set to 614 samples times 512 lines and 224 bands, which is the same size as an AVIRIS sensor image. In order to evaluate the performance of the hardware with different data types, three different versions of the same image were generated with different data types: 64 bits double precision floating point (float64), 32 bits single precision floating point (float32) and 16 bits short integer (int16), respectively. The int16 version of the image was generated multiplying the original reflectance values by 5000 and rounding to the nearest integer. The matrices $\mathbf{A}_j$ were generated following random Gaussian i.i.d., three different versions of the matrices with different data types float32, float16, and int16 were generated. The resulting measurements generated $\mathbf{y}_i$ were stored in float32, float16 and int32 data types, respectively. In the case of the integer data type it was necessary to store the resulting measurements using int32 data type in order to avoid overflows in the dot products.

On all the experimental tests of P-HYCA implementation with different data types, the accuracy is maintained with negligible deviations, when compared with the image reconstructed with the same method in a desktop computer with a GPU GTX 980. Fig. 1 show the execution times in seconds corresponding to the compression process performed in the Jetson TX2 for the three different data types considered. The figure shows that, as expected, the compression process performs much better when int16 and float32 data types are used than when using double precision floating point operations. As a result we may conclude that by using int16 operations it is possible to achieve very similar reconstruction accuracy while at the same time reducing considerably the computational cost. Furthermore this figure also shows that for a measurement ratio of one third the Jetson TX2 performs compression process in about 1 second in the case of the integer and single precision floating point and about 2 seconds in the case of the double precision floating point. Fig. 2 shows the average power for different measurement rate q/n_b and for the three types of data used in experiments. One can notice that

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

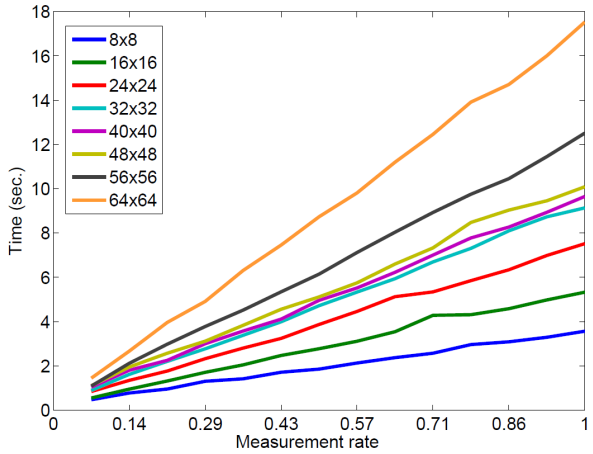

Fig. 3. Processing times in seconds for the Jetson TX2 hardware for different measurement ratios using different window sizes from 8×8 to 64×64.

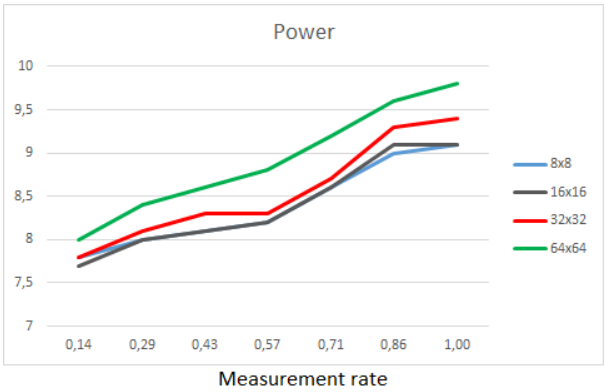

Fig. 4. Average power in Watts (W) for the Jetson TX2 hardware for different measurement ratios using different window sizes from 8×8 to 64×64.

int16 data type is the one that consume less power, however for a measurement ratio of one third the power is very similar among the considered data types.

Finally the performance of the compression process with regards different window sizes ranging from 8×8 to 64×64 was evaluated. For this experiment the int16 data type was used. In the cases in which the window size is grater than 8×8, the matrices $\mathbf{A}_j$ do not fit in the shared memory, therefore in those cases the shared memory is not used. Fig. 3 shows the execution times for different measurement ratios for different window sizes in different colors. As expected, the execution time scales with the window size. This is because the higher the number of matrices $\mathbf{A}_j$ the higher the amount of data to be loaded from the global memory. Furthermore, with higher window size is less likely that two pixels processed in the same block use the same $\mathbf{A}_j$ matrix, resulting in more cache memory faults and increasing the execution time due to the increase of memory accesses. Regarding the average power, one can see, on Fig. 4, that it is higher for larger window size, as expected.

V. Conclusions and Future Lines

In this paper the use of compressive sensing techniques for onboard compression using a low energy consumption GPU Jetson TX2 have been proposed. Conducted experiments using different data types reveal that the use of integer data types does not affect the accuracy in the reconstruction of the compressed data, while at the same time, reduces significantly the processing time onboard and the power. The presented times also reveals that real-time processing in the task of compressing hyperspectral images can be achieved. In future further research may be done using real hyperspectral data and comparing the results of the proposed methodology with other traditional compression schemes.

References

[1] J. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 99, no. 1-16, 2012.

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

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

[4] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 2, pp. 201–205, 2007.

[5] G. Martin, J. M. Bioucas-Dias, and A. Plaza, "Hyca: A new technique for hyperspectral compressive sensing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 5, pp. 2819–2831, 2014.

[6] J. E. Fowler and Q. Du, "Reconstructions from compressive random projections of hyperspectral imagery," in *Optical Remote Sensing*. Springer, 2011, pp. 31–48.

[7] G. Martín and J. M. Bioucas-Dias, "Hyperspectral blind reconstruction from random spectral projections," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 6, pp. 2390–2399, 2016.

[8] P. V. M. Golbabaee, S. Arberet, "Compressive source separation: Theory and methods for hyperspectral imaging," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5096–5110, 2013.

[9] J. M. P. Nascimento and J. M. Bioucas-Dias, "Vertex Component Analysis: A Fast Algorithm to Unmix Hyperspectral Data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 4, pp. 898–910, 2005.

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

[11] 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," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 698–722, March 2013.

[12] S. Bernabe, G. Martin, J. Nascimento, J. Bioucas-Dias, A. Plaza, and V. Silva, "Parallel hyperspectral coded aperture for compressive sensing on gpus," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,*, vol. PP, no. 99, pp. 1–14, 2015.

[13] J. Sevilla, G. Martín, J. Nascimento, and J. Bioucas-Dias, "Hyperspectral image reconstruction from random projections on gpu," in *Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International*. IEEE, 2016, pp. 280–283.

[14] J. Eckstein and D. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Mathematical Programming*, vol. 5, pp. 293–318, 1992.

[15] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "Gpu scheduling on the nvidia tx2: Hidden details revealed," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, Dec 2017, pp. 104–115.