A Hardware-Friendly Algorithm for the On-Board Compression of Hyperspectral Images

Conference Paper · September 2019

DOI: 10.1109/WHISPERS.2018.8747229

5 authors, including:

Raul Guerra Hernández
IUMA
31 PUBLICATIONS **164** CITATIONS
SEE PROFILE

Maria Diaz
Universidad de Las Palmas de Gran Canaria
17 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Yubal Barrios
Universidad de Las Palmas de Gran Canaria
9 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Sebastian Lopez
Universidad de Las Palmas de Gran Canaria
132 PUBLICATIONS **910** CITATIONS
SEE PROFILE

A HARDWARE-FRIENDLY ALGORITHM FOR THE ON-BOARD COMPRESSION OF HYPERSPECTRAL IMAGES

Raúl Guerra, María Díaz, Yubal Barrios, Sebastián López, Roberto Sarmiento

Institute for Applied Microelectronics (IUMA). University of Las Palmas de Gran Canaria (ULPGC).

ABSTRACT

Hyperspectral sensors are able to provide information that is useful for many different applications. However, the huge amounts of data collected by these sensors are not exempt of drawbacks, especially in remote sensing environments where the hyperspectral images are collected on-board satellites and need to be transferred to the Earths surface. In this situation, an efficient compression of the hyperspectral images is mandatory in order to save bandwidth and storage space. Additionally, the computational capabilities of hardware devices available on-board satellites is limited, being its efficiency typically affected by the data types required by the compression algorithms. This work evaluates the precision required by the *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) compressor in order to execute it using just integer arithmetic operations without decreasing its compression rate-distortion performance. The goal is to make the HyperLCA compressor more suitable for on-board hardware processors, which are more efficient executing integer operations than floating point operations.

The proposed integer version of the HyperLCA compressor has been tested using different hyperspectral images, comparing the obtained results with those provided by its floating point version. The obtained results verify that the HyperLCA compressor can be programmed using just integer arithmetic operations in order to provide the same compression results than when using floating point precision, what easies its implementation in many hardware devices.

Index Terms— HyperLCA compressor, hyperspectral compression, on-board compression, integer arithmetic.

This research is partially funded by the European Commission through the ECSEL Joint Undertaking (ENABLE-S3 project, no. 692455) and the Ministry of Economy and Competitiveness (MINECO) of the Spanish Government (ENABLE-S3 project, no. PCIN-2015-225; and PLATINO project, no. TEC2017-86722-C4-1-R). This work was completed while María Díaz was beneficiary of a pre-doctoral grant given by the Agencia Canaria de Investigacin, Innovacin y Sociedad de la Informacin (ACIISI) of the Consejería de Economía, Industria, Comercio y Conocimiento of the Gobierno de Canarias, which is part-financed by the European Social Fund (FSE) (POC 2014-2020, Eje 3 Tema Prioritario 74 (85%)).

1. INTRODUCTION

The HyperLCA compressor [1] is a lossy transform-based compressor specifically designed for providing a good compression performance at a reasonable computational burden for hyperspectral remote sensing applications. It divides the image to be compressed into blocks of pixels and independently compress each of them. The compression of each block of pixels within the HyperLCA algorithm consists of three main compression stages, which are a spectral transform, a preprocessing stage and the entropy coding stage.

The preprocessing and entropy coding stages already use integer arithmetic. However, most of the compression performance achieved by the HyperLCA compressor is obtained in the transformation stage, carried out by the HyperLCA transform. This transform performs the decorrelation and compression of the hyperspectral data, being able to achieve high compression rate-distortion ratios at a reasonable low computational burden and a high level of parallelism. Due to these reasons, this work focuses on evaluating the precision of the operations carried out by the HyperLCA transform in order to find a efficient way of executing its operations using integer arithmetic. The goal is to make the HyperLCA compressor more suitable for those hardware devices that are more efficient executing integer operations than floating point operations [2, 3]. Additionally, the use of floating-point operations computed in different devices may produce slightly different compression results, while the CCSDS 122.1-B-1 recommends that the same transform should produce identical results, bit by bit, regardless of the device used [4].

For doing so, since the HyperLCA transform may produce values smaller than 1, we have made use of the fixed point concept. However, we have used it in a custom way using integer arithmetic and bits shifting [5]. One important advantage of the HyperLCA transform is that the values obtained in its operations are always inside specific numeric ranges. This allows to fix in advance the maximum and minimum values that can be used in each operation by fixing the number of bits used for the integer and decimal part the numbers. This work focuses on analyzing the range of values that may be produced by the different operations of the HyperLCA transform in order to determine the required number of bits to be used for representing the different variables involved in the operations of the algorithm. This allows to fix the number of bits that need to be shifted in the different operations and operate using standard integer arithmetic.

2. HYPERLCA TRANSFORM PRECISION EVALUATION.

The HyperLCA transform sequentially selects the most different pixels of the image block being compressed. The set of selected pixels is then used for projecting the image block, obtaining a spectral decorrelation and reduction of the data. The reduction achieved within this process directly depends on the number of pixels selected. Selecting more pixels provides better decompressed images but lower compression ratios, understanding the compression ratio as the relation between the size of the real image and the compressed one (the higher, the better). The number of pixels to be selected by the HyperLCA transform, p_{max}, is directly determined according to the desired compression ratio. Both the extracted pixels as well as information of the image block compressed as a linear combination of these pixels are set as the outputs of the HyperLCA transform. This is due to the fact that both are needed for recovering the original hyperspectral image.

Algorithm 1: HyperLCA transform.

```
   Inputs: $M = [r_1, r_2, ..., r_{N_{p_{\text{block}}}}]$, $p_{max}$, $c$
1  $P = [p_1, p_2, ..., p_{p_{\max}}]$; {Extracted pixels.}
2  $V = [v_1, v_2, ..., v_{p_{\max}}]$; {Projected image block vectors.}
3  $M_c = [x_1, x_2, ..., x_{N_{p_{\text{block}}}}]$ {Centralized version of $M$}
4  for $i = 1$ to $p_{\max}$ do
5      for $j = 1$ to $N_p$ do
6          $b_j = {x_j}^t \cdot x_j$;
7      end
8      $j_{\max} = \arg\max(b_j)$;
9      $p_i = r_{j_{\max}}$;
10     $q = x_{j_{\max}}$;
11     $u = x_{j_{\max}} / ((x_{j_{\max}})^t \cdot x_{j_{\max}})$;
12     $v_i = u^t \cdot M_c$;
13     $M_c = M_c - v_i \cdot q$;
14 end
   Outputs: $P = [c, p_1, p_2, ..., p_{p_{\max}}]$,
            $V = [v_1, v_2, ..., v_{p_{\max}}]$
```

The pseudocode that describes the HyperLCA transform for spectrally decorrelating and reducing one block of pixels of the hyperspectral image is presented in Algorithm 1, where matrix M contains $N_{p_{\text{block}}}$ real hyperspectral pixels $[r_1, r_2, ..., r_{N_{p_{\text{block}}}}]$ of a single image block, placed in columns. Each of these pixels is a vector of N_b components, being N_b the number of bands of the hyperspectral image. The input vector c corresponds with the average pixel of the image, also called centroid, in integer values. Additionally, the set of vectors $P = [p_1, p_2, ..., p_{p_{\max}}]$ and $V = [v_1, v_2, ..., v_{p_{\max}}]$ contain the vectors which are extracted by the HyperLCA transform as the compressed information. Specifically, P will store the p_{max} real hyperspectral pixels selected by the HyperLCA transform as the most different pixels of the data set (*Pixels*), and vectors contained in V will store the information of the image that can be represented by the extracted pixels (*V vectors*). Each of the V vectors has $N_{p_{\text{block}}}$ components, one per pixel. These vectors are used by the inverse transform for recovering the real image block.

The number of bits used for representing each hyperspectral image value depends on the sensor capturing the image. The number of bits used for the set of output pixels selected by the HyperLCA transform, $P = [p_1, p_2, ..., p_{p_{\max}}]$, will correspond with the number of bits used by the sensor for representing each image value, as described in [1]. Similarly, the number of bits used for the output $V = [v_1, v_2, ..., v_{p_{\max}}]$ vectors is an input parameter that need to be specified by the user, as described in [1]. Hence, this work focuses on determining just the number of bits to be used for the integer and fractional part of the different variables involved in the operations of the HyperLCA transform, according to the range of values that these operations may produce. After doing so, the entire HyperLCA transform can be programmed using integer arithmetic and bits shifting according to the number of bits specified for each of the HyperLCA transform variables.

Table 1: Precision used for the HyperLCA transform variables

Variable	bits per value		
	Total	Integer part	Decimal part
c	32	32	0
M_c	32	20	12
b	64	48	16
q	32	20	12
u	32	2	30
v	32	2	30

First of all, the HyperLCA transform centers the image block, M, and stores it in M_c, as described in line 3 of Algorithm 1. This is done by subtracting the centroid pixel to all the pixels of the image block. The values of the centroid pixel, c, are in the same range that the values of the image, and so, it can be represented using the same number of bits, considering no decimal part. Nevertheless, once that c is calculated and coded, its values are turned into 32 bits for simplifying the subsequent operations inside the HyperLCA transform. The number of bits used for representing the values of M_c is also 32. While the HyperLCA transform extracts pixels and the information that they are able to represent, the information present in matrix M_c decreases, and its values are contained in an smaller range. However, some specific values may increase in some very unlikely situations. Due to this reason, in order to guarantee that the range of the values of M_c can be represented, 20 bits are used for the integer part. This means that, for images of 16 bits per image value, there would be a margin of 4 bits. The other 12 bits are used for the decimal part, what provides a resolution of 2^{-12}.

Secondly, the pixels are sequentially extracted by the HyperLCA transform as described in lines 4 to 15 of Algorithm 1. In this process, the brightness of each pixel within the M_c image block is first calculated in lines 5 to 7. The extracted pixels are selected as those pixels from M which corresponds with the highest brightness in matrix M_c, as shown in line 9. Then, the orthogonal projection vectors q and u are accordingly obtained as shown in lines 10 and 11. Vector q is directly selected as one pixel of M_c, and hence, its values can be represented using the same number of bits, 20 for the integer part and 12 for the decimal part. The calculation of the brightness is the most troublesome operation, since it involve N_b products and $N_b - 1$ addition operations. In order to be able to accomplish this operation without overflowing, the brightness of each pixel, b_j, is represented using 64 bits, using 48 of them for the integer part and 16 bits for the decimal part. Since the vector u is obtained by dividing the vector q by the maximum brightness value, $b_{j_{\max}}$, its values are in the range (-1, 1). Due to this reason, the values of this vector are represented using 2 bits for the integer part and 30 for the decimal part, obtaining a resolution of 2^{-30}.

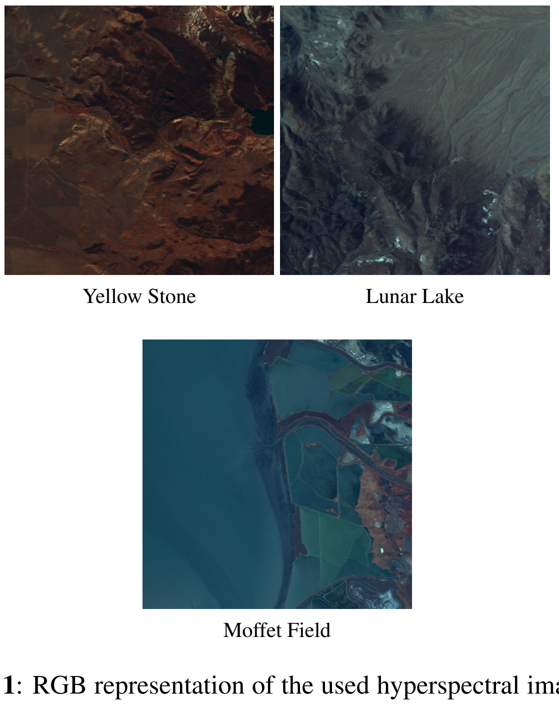

Fig. 1: RGB representation of the used hyperspectral images collected by the AVIRIS sensor.

Finally, the information that can be spanned by the defined orthogonal vectors u and q is stored in the projected image block vector v_p and subtracted from the M_c image block, as described in lines 12 to 13 of Algorithm 1. The process finishes when the $p_{\max}$ pixels p_j have been selected and the information of the image block that they can spanned has been stored in the $p_{\max}$ v_j vectors. The values of the vector v are always in the (-1, 1] range, as it is described in [1]. Due to these reason, the values of this vector are represented using 2 bits for the integer part and 30 for the decimal part. Since this vector is one of the outputs of the HyperLCA transform, and the number of bits that are used for coding it in the subsequent compression stages is specified by the user, the decimal part of this vector is then truncated. Accordingly, the resolution of the output v vectors is $2^{N_{\text{bits}}-2}$, where N_{bits} is the number of bits specified by the user for coding the V vectors.

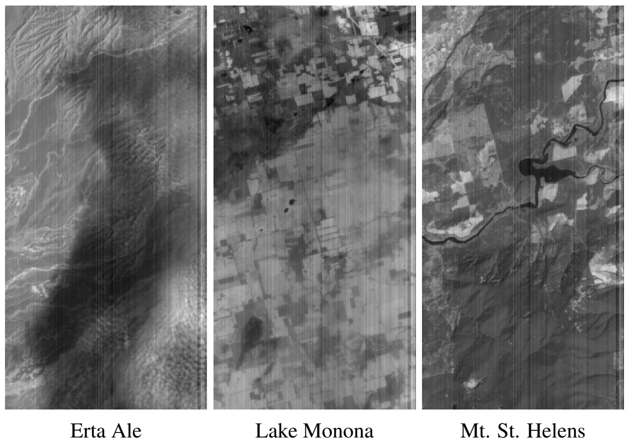

Fig. 2: Grey scale representation of the used hyperspectral images collected by the Hyperion sensor.

Table 1 summarizes the number of bits to be used for the integer and fractional part of the different variables involved in the operations of the HyperLCA transform according to this analysis. The number of bits displayed in this table represents the number of bits used for programing the integer version of the HyperLCA compressor proposed in this work.

3. RESULTS.

A heterogeneous test bench has been selected in order to evaluate the behavior of the proposed integer version of the HyperLCA compressor against its reference single precision floating-point version. On one side, we have selected three different images from the well known Airbone Visible/Infrarred Imaging Spectrometer (AVIRIS) [6]. This sensor captures 224 spectral bands in the wavelength range of 400 to 2500 nm, coding each pixel value using 16 bits [7]. These three images are the Yellow Stone (Sc 0) radiance image and the Lunar Lake image and the Moffet Field reflectance images. These images have been cropped to portions of 512x512 pixels. Figure 1 graphically shows a false color representation of the portions of the AVIRIS hyperspectral images used.

On the other side, three different uncalibrated images from the Hyperion sensor [8] have been used. The Hyperion sensor produces 242 spectral bands between 355.59 and 2577.08 nm, coding the image values using 12 bits. The Hyperion images that have been used in the experiments are the Erta Ale image, the Lake Monona image and the Mt. St. Helens one. These images have been cropped to portions of 512x256 pixels. Figure 2 shows a grey scale representation of these images.

The compression performance of both versions of the HyperLCA compressor has been evaluated in terms of *Signal-to-Noise Ratio* (SNR) and *Maximum Single Error* (MaxSE) for every compression ratio achieved (bpppb), as described in Equations 1 and 2, where *Hy* refers to the original hyperspectral image treated as a 2D matrix with the pixels placed in columns, and *Hy'* refers to the compressed-decompressed one. Table 2 shows the results obtained with the images collected by the AVIRIS sensor. Similarly, Table 3 shows the results obtained with the images collected by the Hyperion sensor. As done in [1], the input parameter N_{bits}, which determines the number of bits used for coding the obtained V vectors, has been set to 12 and 8 bits for the images collected by the AVIRIS and Hyperion sensors, respectively. Additionally, the number of pixels per block has been set to 1024 since this value provides good compression results, as demonstrated in [1].

$$\text{SNR} = 10 * \log_{10}\left(\frac{\sum_{p=1}^{N_p}\sum_{b=1}^{N_b}(\text{Hy}_{b,p} - \text{Hy}'_{b,p})^2}{\sum_{p=1}^{N_p}\sum_{b=1}^{N_b}(\text{Hy}_{b,p})^2}\right) \quad (1)$$

$$\text{MaxSE} = \text{maximum}(|\sum_{p=1}^{N_p}\sum_{b=1}^{N_b}(\text{Hy}_{b,p} - \text{Hy}'_{b,p})|) \quad (2)$$

The results displayed in Tables 2 and 3 demonstrate that the proposed integer version of the HyperLCA compressor provides almost identical compression results than its reference single precision floating-point version, with independence of the sensor and image used, and for the different targeted compression ratios. This represents an important advantage for many compression applications since the integer arithmetic operations are more efficient implementable in many hardware devices than the floating-point operations.

4. CONCLUSIONS.

In this paper the precision involve in the operations executed by the *Lossy Compression Algorithm for Hyperspectral Image Systems* (HyperLCA) has been evaluated in order to develop a integer version of this compressor. The fixed point concept has been used in order to be able to work with values smaller than 1, implementing it in a custom way using integer arithmetic and bits shifting. The goal is to make the HyperLCA compressor more easily implementable in those hardware devices that are not so well optimized for floating-point operations as they are for integer arithmetic.

The proposed integer version of the HyperLCA compressor has been tested using many hyperspectral images with a wide range of characteristics and from different sensors. The compression results have been compared with those produced by the single precision floating-point version of the HyperLCA compressor, in terms of Signal-to-Noise Ratio (SNR) and Maximum Single Error (MaxSE). According to the obtained results it is concluded that the proposed integer version of the HyperLCA compressor provides almost identical compression results than its single precision floating-point version, with independence of the sensor and image used, and for the different targeted compression ratios.

Table 2: Comparison of the compression results obtained with both the reference and proposed versions of the HyperLCA compressor for the images collected by the AVIRIS sensor.

CR (bpppb)	SNR (dB)		MaxSE	
	Reference	Proposed	Reference	Proposed
Yellowstone (Sc 0)				
3.23	53.74	53.69	91	91
1.64	50.84	50.82	190	190
1.08	47.54	47.53	421	421
0.80	44.11	44.11	713	714
0.63	41.13	41.13	1240	1241
0.52	38.95	38.95	1745	1744
0.46	37.83	37.83	2164	2164
0.40	36.51	36.51	2637	2637
Lunar Lake				
3.12	53.93	53.33	18	19
1.56	51.65	51.31	40	41
1.02	50.16	49.93	58	57
0.75	48.85	48.68	86	87
0.59	47.55	47.42	126	126
0.48	46.33	46.22	152	158
0.43	45.58	45.50	172	172
0.38	44.66	44.59	197	197
Moffet Field				
3.05	51.29	50.83	48	48
1.50	42.65	42.60	396	396
0.97	36.76	36.75	695	395
0.71	33.58	33.56	960	960
0.56	31.47	31.46	1255	1255
0.46	30.18	30.18	1404	1404
0.41	29.51	29.50	1462	1462
0.35	28.81	28.81	1615	1615

Table 3: Comparison of the compression results obtained with both the reference and proposed versions of the HyperLCA compressor for the images collected by the Hyperion sensor.

CR (bpppb)	SNR (dB)		MaxSE	
	Reference	Proposed	Reference	Proposed
Erta Ale				
1.70	39.24	39.14	99	100
0.86	36.58	36.53	173	173
0.58	35.07	35.04	253	253
0.41	33.14	33.13	379	378
0.33	31.87	31.86	462	461
0.28	31.00	30.99	632	631
0.23	29.95	29.94	761	759
0.20	29.57	29.56	769	767
Lake Monona				
1.70	39.36	39.30	96	98
0.85	36.38	36.35	212	212
0.56	34.76	34.74	275	274
0.40	32.53	32.51	486	485
0.32	31.21	31.20	641	641
0.27	30.09	30.08	717	716
0.22	29.02	29.01	862	862
0.19	28.48	28.48	861	862
Mt. St. Helens				
1.69	39.12	39.03	86	85
0.85	36.38	36.33	162	162
0.57	34.80	34.76	254	270
0.40	32.51	32.50	473	471
0.33	31.28	31.27	559	558
0.27	30.32	30.32	702	702
0.22	29.42	29.40	743	742
0.20	28.95	28.93	947	947

5. REFERENCES

[1] Raúl Guerra, Yubal Barrios, María Díaz, Lucana Santos, Sebastián López, and Roberto Sarmiento, "A new algorithm for the on-board compression of hyperspectral images," *Remote Sensing*, vol. 10, no. 3, pp. 428, 2018.

[2] Walter B Ligon, Scott McMillan, Greg Monn, Kevin Schoonover, Fred Stivers, and Keith D Underwood, "A re-evaluation of the practicality of floating-point operations on fpgas," in *FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on*. IEEE, 1998, pp. 206–215.

[3] Keith B Cullen, Guenole CM Silvestre, and Neil J Hurley, "Simulation tools for fixed point dsp algorithms and architectures," in *Proc. International Conference on Signal Processing*, 2004.

[4] "Consultative committee for space data systems (ccsds). spectral preprocessing transform for multispectral and hyperspectral image compression. blue book. issue 1. september 2017. [online]. https://public.ccsds.org/pubs/122x1b1.pdf," .

[5] Erick L Oberstar, "Fixed-point representation & fractional math," *Oberstar Consulting*, p. 9, 2007.

[6] Gregg Vane, Robert O Green, Thomas G Chrien, Harry T Enmark, Earl G Hansen, and Wallace M Porter, "The airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sensing of Environment*, vol. 44, no. 2, pp. 127–143, 1993.

[7] "Jet propulsion laboratory, nasa. airbone visible infrared imaging spectrometer website. [online]. http://aviris.jpl.nasa.gov/aviris/index.html," .

[8] "U.s. geological survey and nasa. earth observing 1, hyperion website. [online]. https://eo1.usgs.gov/sensors/hyperion," .