# HIERARCHICAL PIXEL AVERAGING: A NEW IMAGE COMPRESSION APPROACH

**Riccardo Iudica [(1)], Jordi Portell [(1,2,3)], Enrique García-Berro [(1,4)]**

*[(1)] Institute for Space Studies of Catalonia (IEEC)*
*c/Gran Capità 2-4, 08034 Barcelona, Spain*
*Email: riccardo.iudica@outlook.com*

*[(2)] DAPCOM Data Services S.L.*
*ESA BIC Barcelona (Parc UPC – PMT, RDIT). C/ Esteve Terrades 1, 08860 Castelldefels (Barcelona), Spain*
*Email: jordi.portell@dapcom.es*

*[(3)] Departament d'Astronomia i Meteorologia (ICC-UB), Universitat de Barcelona*
*c/Martí i Franquès 1, 08028 Barcelona, Spain*

*[(4)] Departament de Física Aplicada, Universitat Politècnica de Catalunya (UPC)*
*C/ Esteve Terrades 5, 08860 Castelldefels (Barcelona), Spain*
*Email: enrique.garcia-berro@upc.edu*

## ABSTRACT

FAPEC is the adaptive version of PEC, a data compression algorithm based on an entropy coder. By default, FAPEC provides basic pre-processing stages such as delta pre-processing, that is, outputting the differences between consecutive values. Although this approach already provides good results, in case of image compression a further improvement can be done by taking advantage of the inter-pixel correlations. Besides that, FAPEC only performs lossless compression and it would benefit from a lossy implementation. Here we describe a new image pre-processing algorithm, called Hierarchical Pixel Averaging (HPA), which has been developed as a pre-processing stage for FAPEC. HPA divides an image in blocks of 16 by 16 pixels which are subsequently divided into smaller blocks, defining different block levels up to the basic one where one block corresponds to one pixel. Average pixel values are determined for each level, from which differential coefficients are extracted leading to values which are smaller (and thus more compressible) than the original ones. This algorithm provides some advantages with respect to other solutions. It decreases the entropy levels of the data that are passed to FAPEC for compression, thus increasing the achievable compression ratios. It does not present the typical artefacts seen in wavelet-based image compression algorithms, and it provides better resolution in sharp image edges. It is based on simple arithmetic operations, allowing a very simple (thus quick) implementation, furthermore avoiding any floating-point operations – a feature which is interesting for satellite or embedded data compression. The algorithm allows the introduction of controlled losses with several quality levels, furthermore allowing to progressively decompress a given image – from the lowest quality to the highest one. We present a first implementation of the FAPEC+HPA image compression algorithm and the results obtained on a variety of images, both for the lossless and lossy cases with different quality levels. Our results indicate that this solution offers a performance comparable to that of the CCSDS 122.0 standard.

Keywords: Image data compression, FAPEC, HPA, lossless, lossy, hierarchical, CCSDS, 122.0.

## I. INTRODUCTION

Given the big amount of data in the new generations of satellites for space exploration and Earth observation, a very efficient compression algorithm is needed which can reduce drastically the size of the data to be transferred to the ground segment. Each space mission has strict requirements in terms of available bandwidth for the data downlink, available time for such transfer, and available computing resources on-board. Each of these resources is very expensive, so the cost of each mission can be strongly affected by the compression algorithm chosen.
Other limitations come from the architecture of the on-board processor or hardware implementation. In some cases they cannot perform floating-point operations, but in general it is advisable to avoid so – especially in case of lossless compression. This means that the algorithm must be kept as simple as possible. For the lossy compression it means avoiding wavelet-like or DCT-like (discrete cosine transform) processing [1, 2].
In this paperwork we introduce a new image compression algorithm, called Hierarchical Pixel Averaging (HPA). It is a pre-processing stage for FAPEC [3], an entropy coder which can offer better results than the current standard for

lossless data compression in space (CCSDS 121.0 [4]). The latter is based on the Rice-Golomb coding method, which is conceived for noiseless data following geometric distributions. Its performance rapidly degrades in the presence of outliers. FAPEC, on the contrary, is much more resilient in front of unexpected data, providing good compression efficiency under almost any situation.

The goal of the HPA approach is twofold. On one hand, we intend to achieve better compression ratios by taking advantage of the inter-pixel correlations typically present in an image. Its design has been kept simple: it only uses simple arithmetic calculations, which makes it suitable to be embedded on-board satellite payloads. On the other hand, its design allows progressively introducing controlled losses in the compression, further increasing the compression ratios at the expense of a reasonable quality loss in the reconstructed image.

## II. THE HPA CONCEPT

### II.1. Image blocks and levels

The basic image block of the HPA algorithm consists of just 16×16 pixels. Let us call the entire block a *Level-4 block*. It can be divided into 4 smaller areas or quadrants, each one with an 8-pixels side, which we call *Level-3 blocks*. In the same way, we can define *Level-2 blocks* (with 4-pixels sides) and *Level-1 blocks* (with 2-pixels sides). Finally, the *Level-0* consists of the 256 individual pixels. Summarizing, for each 256 pixels block, we define 256 level-0 pixels, 64 level-1 blocks (each made of 2×2 pixels), 16 level-2 blocks (each made of 4×4 pixels), 4 level-3 blocks (each made of 8×8 pixels), and finally 1 level-4 block (made of 16×16 pixels). A visual representation can be seen in Fig. 1, where only a fraction of the blocks is highlighted and numbered for the sake of clarity.

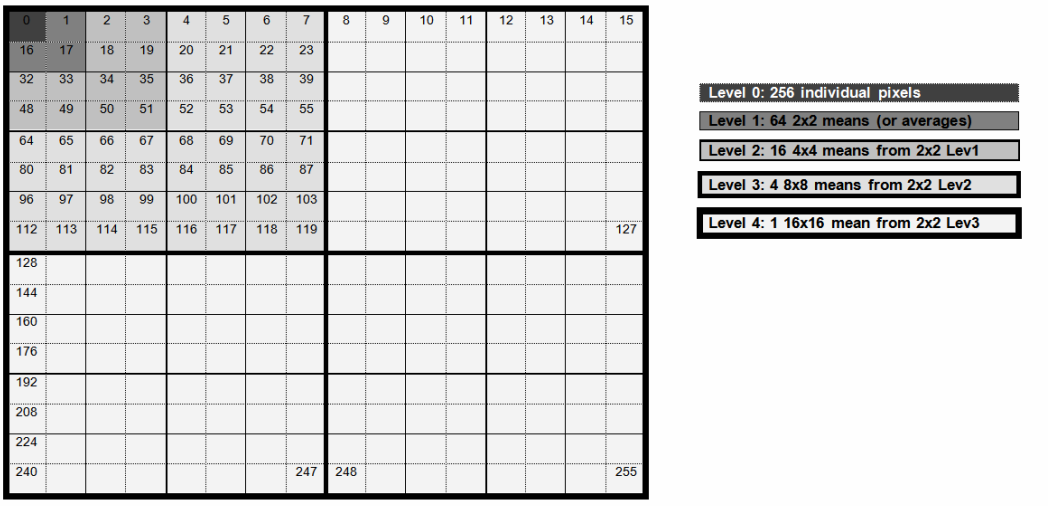


Figure 1: One HPA block, formed by 16×16 pixels

A consequence of such a hierarchy is that each block of level $n + 1$ is composed of 4 blocks of level $n$, and that the average value of the $n + 1$ level block is equal to the mean of the average values of the 4 sub-blocks of level $n$. Eq. (1) describes this relation as follows:

$$\overline{H_{n+1}(A)} = \frac{\overline{H_n(B)} + \overline{H_n(C)} + \overline{H_n(D)} + \overline{H_n(E)}}{4} + \rho_{n+1}(A) \tag{1}$$

Where $\overline{H_{n+1}(A)}$ is the average of block A from level $(n + 1)$, $\rho_n(A)$ is the remainder of its division by 4, $\overline{H_n(B)}$ is the average of the block B from level $n$, etc. By inverting Eq. (1), we can calculate one of the lower-level coefficients by knowing the one from the upper level and the other three of the present level as follows:

$$\overline{H_n(E)} = 4 \times \overline{H_{n+1}(A)} + \rho_{n+1}(A) - \overline{H_n(B)} - \overline{H_n(C)} - \overline{H_n(D)} \tag{2}$$

The basic concept behind the HPA algorithm is to extract differential coefficients from the average values and to compress them instead of the original pixels. The relation between the differential coefficient and its original value is shown in Eq. (3):

$$\Delta H_n(i) = \overline{H_n(i)} - \overline{H_{n+1}(A)} \tag{3}$$

Combining Eq. (2) with Eq. (3), the whole block of 16×16 pixels can be represented with the following coefficients:

- Level 0 block: 192 × $\Delta H_0$
- Level 1 block: 48 × $\Delta H_1$
- Level 2 block: 12 × $\Delta H_2$
- Level 3 block: 3 × $\Delta H_3$
- Level 4 block: 1 × $\overline{H_4}$

The total number of coefficients to be coded (that is, 256) remains, this way, identical to the number of original pixels in the block. Nevertheless, we have some overhead caused by the reminders:

- Level 1 block: 64 $\rho_1$ remainders, which can be coded in just 2 bits each – leading to 16 bytes in total.
- Level 2 block: 16 $\rho_2$ remainders, that is, 4 bytes.
- Level 3 block: 4 $\rho_3$ remainders, that is, 1 byte.
- Level 4 block: 1 $\rho_4$ remainder, that is, 2 bits.

The total overhead of the HPA algorithm is given by those reminders, which sum 21 bytes + 2 bits. This represents an overhead of 0.66 bits per pixel, which is compensated by the smaller entropy levels achieved with the differential *H* coefficients.

### II.2. HPI variant

A variant of the HPA algorithm is the Hierarchical Pixel Interpolation algorithm, or HPI. The difference is in the differential coefficient calculation. As we saw, in the HPA the difference is calculated, as in Eq. (3), by subtracting the lower-level average value from the high level average values. Instead, in the HPI, first of all, the average values of level 1 are interpolated in order to predict the pixel values. Subsequently, the differential coefficients are calculated by subtracting lower-level average values from the interpolated pixels just calculated. Eq. (3) is, therefore, modified into the following:

$$\Delta H_0(i) = P_0(i) - P_0(i)' \tag{4}$$

where $P_0(i)'$ is the interpolated pixel. Since the interpolated pixel is more similar to the original pixel than the average coefficient of the upper level, the differential coefficients of the HPI implementation have lower entropy and, hence, are more compressible.

### II.3. Implementation

A first prototype of the algorithm has been implemented in C and tested on an Intel i686-64bit platform running Fedora Linux. The GNU Compiler Collection (GCC) was used to compile the code to binary files.

## III. LOSSY HPA

One of the advantages of the HPA algorithm is that it is easy to introduce controlled losses. The evolution to the lossy version of the algorithm consists of removing some of the least significant bits (LSB) from the $\Delta H$ or the $\rho$ coefficients. Note that HPA allows this because we keep working in the image domain, whereas other approaches such as DWT would lead to strange and uncontrolled effects when removing such bits from the transformed domain coefficients. In the Lossy HPA we have opted to allow different configurations depending on the quality level required. In particular, in our strategy the number of bits removed from the $\Delta H$ coefficients varies from 0 to 8, and the number of bits removed

from the $\rho$ remainders varies from 0 to 2. Following this strategy, we define the Quality Levels (hereafter QL) of lossy HPA as follows. QL 0 is the lossless compression (no LSB removal). In QL 1 we remove 1 bit from $\Delta H_0$ and 1 bit from $\rho_1$, which in practice is equivalent to removing 1 bit per pixel. In QL 2, 2 bits from $\Delta H_0$ and 2 bits from $\rho_1$ are removed, meaning that we remove $\rho_1$ completely. In QL 3, we discard 3 bits from $\Delta H_0$, 1 bit from $\Delta H_1$, 2 bits from $\rho_1$ and 1 bit from $\rho_2$. In QL 4 we discard 4 bits from $\Delta H_0$, 2 bits from $\Delta H_1$, 2 bits from $\rho_1$ and 2 bits from $\rho_2$. This otherwise intuitive and progressive approach is continued until QL 8, where we drop 8 bits from $\Delta H_0$ (that is, this coefficient is completely removed), 6 bits from $\Delta H_1$, 4 bits from $\Delta H_2$, 2 bits from $\Delta H_3$, and all $\rho$ remainders.
The LSB are removed by right shifting the same amount of bits. The consequence of this removal is that the histogram of the differential coefficients results more steep and centred around 0. Fig. 2 compares the histograms of the $\Delta H_0$ coefficients of the lossless version against the Q2 lossy version.

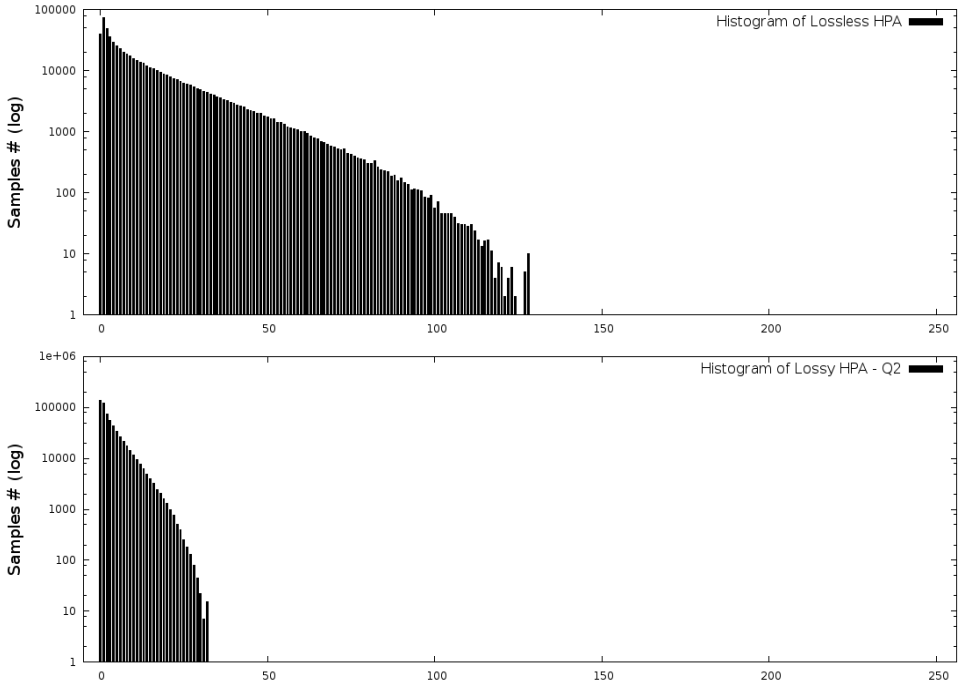


Figure 2: Qualitative comparison of the HPA coefficients entropy for lossy (bottom panel) and lossless (top panel)

## IV. TESTS AND RESULTS

We have run several tests of the HPA algorithm (using the HPI variant) on a set of different images in raw format. The main goal of these tests is to compare the HPA+FAPEC combination against the CCSDS 122.0 standard, nowadays used in space applications. Two test campaigns have been executed to compare the performance of these compressors, namely, using the lossless configuration (considering compression ratios and execution times), and using the lossy versions with different quality levels (considering also the Peak Signal to Noise Ratio, or PSNR). Please note that the CCSDS 122.0 implementation used for these tests uses a lossy compression approach which is different from the standard one, namely, here we completely remove a given number of quadrants of DWT coefficients depending on the lossy level. Specifically, for the first level we remove one fourth of the grandchildren coefficients (that is, one fourth of the total DWT coefficients). For the second level, we remove all the grandchildren coefficients (thus leaving only one fourth of coefficients to code). For the third level, we remove all the parent coefficients (thus leaving only $1/16^{th}$ of coefficients). And finally, for the fourth level we only leave the DC coefficients. More details can be found in [2]. In future tests we will include the comparison with the standard lossy compression approach.

### IV.1. Image compression corpus

Fig. 3 shows some of the images that have been used for the tests. They are, from left to right and from top to bottom: *banyoles*, *catedral*, *eixample*, *pirineus*, *Europe* (an EUMETSAT image), and *florida* (a NOAA image). The four first images are 1024×592 in 8-bit greyscale. The original meteorological images were in 8-bit colour (24-bit per pixel), but here we have reduced them to 8-bit greyscale. They are 3600×2992 and 1408×1248 pixels respectively.

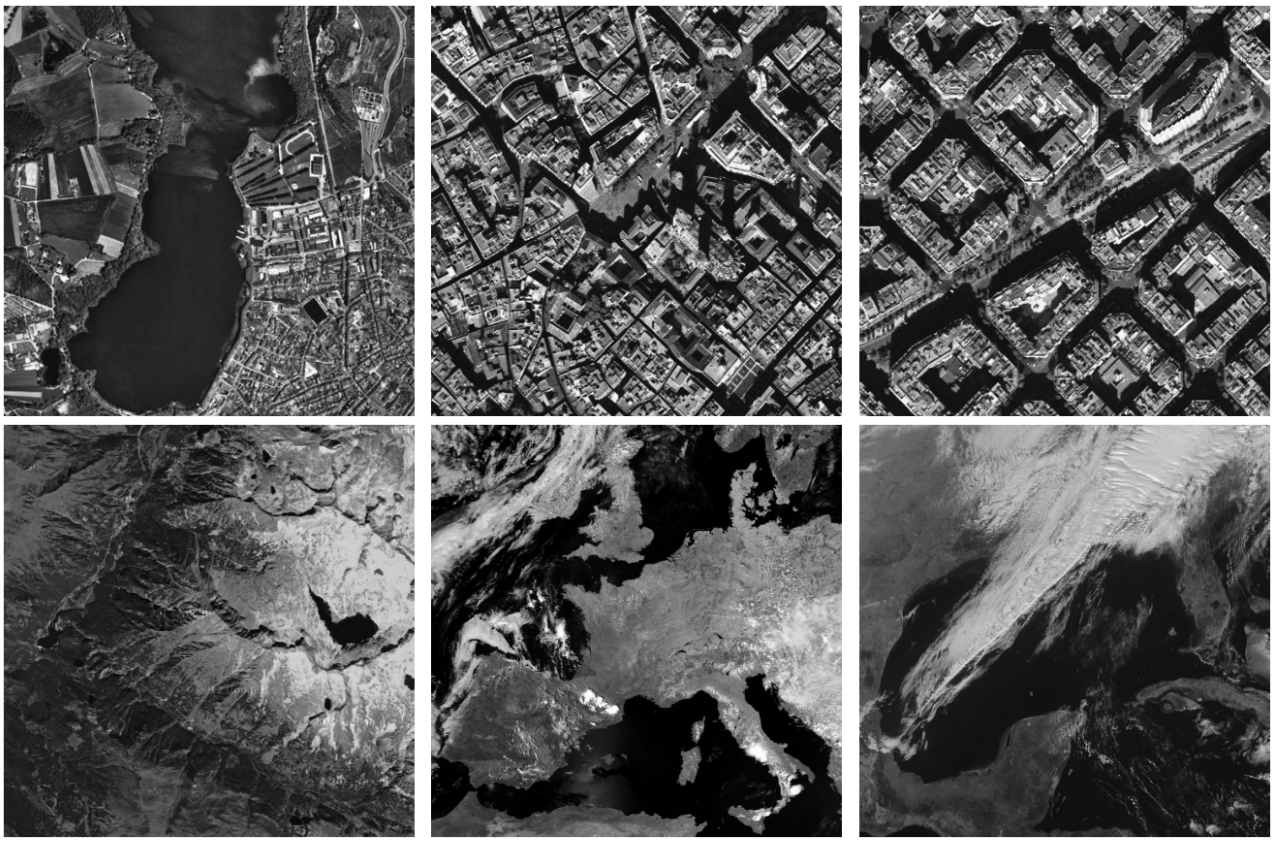

Figure 3: Image corpus used in our tests

### IV.2. Lossless compression performance

We first compare our prototype of HPA+FAPEC against the original FAPEC and the standard CCSDS 122.0. Table 1 compares the performance in terms of compression ratio ($C_R$) and execution time. The results show that the HPA+FAPEC combination performs more or less similar to FAPEC alone, in terms of $C_R$. The execution time is understandably larger than FAPEC alone, due to the HPA pre-processing. Comparing HPA with CCSDS 122.0 we can see that, in general, the latter compresses the images more than the former. This is an acceptable result, if we consider that the principal application of the HPA algorithm is to create a framework for lossy image compression. In terms of execution time, HPA runs in average 27% faster than CCSDS 122.0 – despite of being a prototype.

Table 1: Lossless performance comparison between HPA+FAPEC and CCSDS 122.0

| | **FAPEC** | | **HPA + FAPEC** | | **CCSDS 122.0** | |
|---|---|---|---|---|---|---|
| | $C_R$ | Exec. time | $C_R$ | Exec. time | $C_R$ | Exec. time |
| *banyoles* | 1.28 | 78 ms | 1.29 | 164 ms | 1.45 | 209 ms |
| *catedral* | 1.09 | 81 ms | 1.08 | 171 ms | 1.17 | 231 ms |
| *eixample* | 1.13 | 80 ms | 1.14 | 172 ms | 1.25 | 228 ms |
| *pirineus* | 1.32 | 75 ms | 1.29 | 166 ms | 1.47 | 211 ms |
| *florida* | 1.85 | 1201 ms | 1.76 | 2660 ms | 2.27 | 2999 ms |
| *europe* | 1.59 | 210 ms | 1.54 | 449 ms | 1.94 | 539 ms |

### IV.3. Lossless compression performance on noisy data

We must remind that one of the strengths of FAPEC is its resilience in front of outliers, and therefore the HPA+FAPEC combination should also reveal that. In order to prove the robustness of our solution on that regard, we have added some noise to the images and compared the loss in $C_R$ against CCSDS 122.0. We have introduced one noisy pixel every $100^{th}$ pixel – which is quite sensible in space instrumentation. Table 2 shows the results of this test, which reveals that the

percentage of loss in 122.0 $C_R$ is typically twice or three times the loss with FAPEC. This proves that the HPA algorithm is more robust to the noise than CCSDS 122.0.

Table 2: Percentage of $C_R$ loss between FAPEC + HPA and CCSDS 122.0 in presence of noise

| | **HPA + FAPEC** | | **CCSDS 122.0** | |
|---|---|---|---|---|
| | $C_R$ | $C_R$ loss | $C_R$ | $C_R$ loss |
| *banyoles* | 1.27 | 1.25 % | 1.40 | 3.22 % |
| *catedral* | 1.07 | 0.34 % | 1.16 | 0.77 % |
| *eixample* | 1.13 | 0.59 % | 1.23 | 1.44 % |
| *pirineus* | 1.27 | 1.05 % | 1.43 | 2.90 % |
| *florida* | 1.72 | 2.54 % | 2.17 | 4.50 % |
| *europe* | 1.50 | 2.49 % | 1.78 | 8.11 % |

### IV.4. Lossy compression performance

We have considered the HPA quality levels from QL1 to QL7 for the comparison of both the $C_R$ and the execution time. Fig. 4 shows the comparison of the compression ratios achieved by HPA and CCSDS 122.0. The left panel shows the results for images of similar size, namely, 1024×592, whereas the right panel presents the results for larger images. In general, HPA achieves a better PSNR than CCSDS 122.0. Only for the *florida* image we achieve slightly worse PSNR values, but on the other hand we offer more intermediate quality levels, also reaching a higher PSNR value for the QL1 case with respect to the 122.0 level 1.

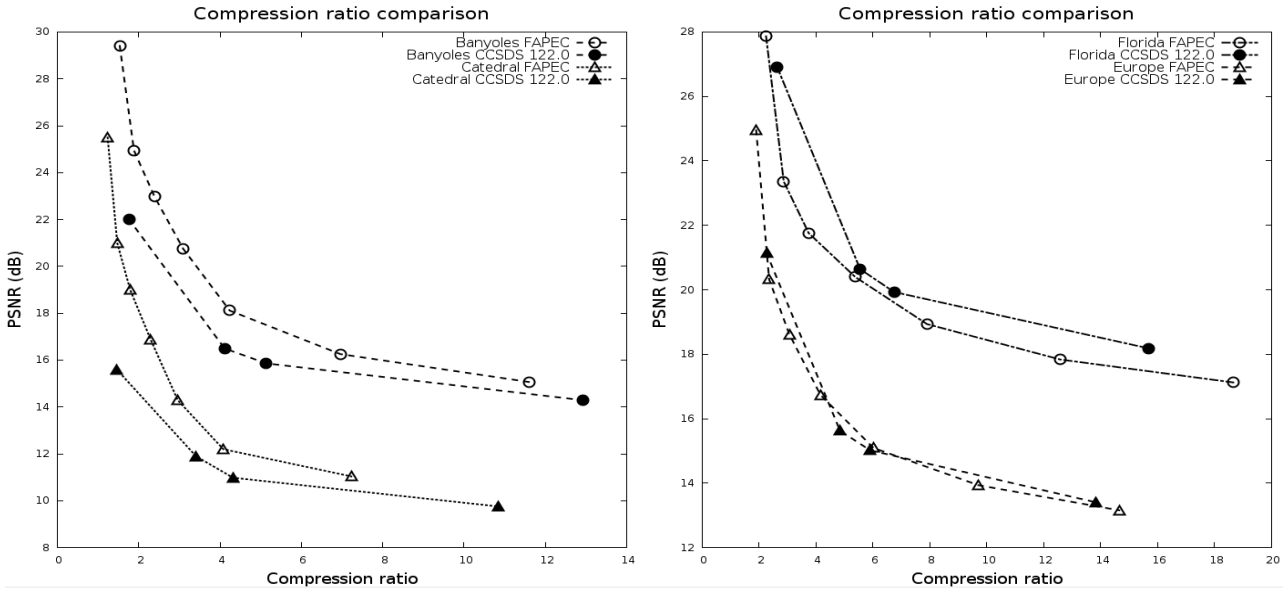


Figure 4: Comparison of the $C_R$ between Lossy FAPEC and CCSDS 122.0

Fig. 5 shows that CCSDS 122.0 typically runs slower than HPA+FAPEC in the quality level Q1, but it runs faster at higher quality levels. This behaviour is due to the fact that the implementation we used for CCSDS 122.0 processes fewer coefficients at each quality level – thus reducing the execution time required by the bit plane encoder. On the contrary, the HPA algorithm processes the image completely, sending always the same number of coefficients to FAPEC. The reduction in execution time for FAPEC is, hence, due to the lower entropy of the HPA coefficients. It is important to point out that this is the first prototype of the HPA algorithm and that further optimizations are still possible to make it more efficient.

Finally, Fig. 6 shows a detail of the *banyoles* image recovered after a lossy compression, using QL 6 for HPA and QL 2 for CCSDS 122.0. This is actually an "error map", where each pixel shows the error in the reconstruction of that pixel with respect to the original image. White pixels mean a perfect reconstruction, while dark or black pixels mean a larger error. In this comparison we can see a more random appearance of the reconstruction errors for the case of HPA, especially around sharp edges of the image, whereas in the case of 122.0 the errors are more evident in such edges. That

means that with lossy HPA the reconstruction errors are more smoothly distributed all over the image, whereas with 122.0 these are more evident in the sharp edges – effectively smoothing such edges in the reconstructed image.

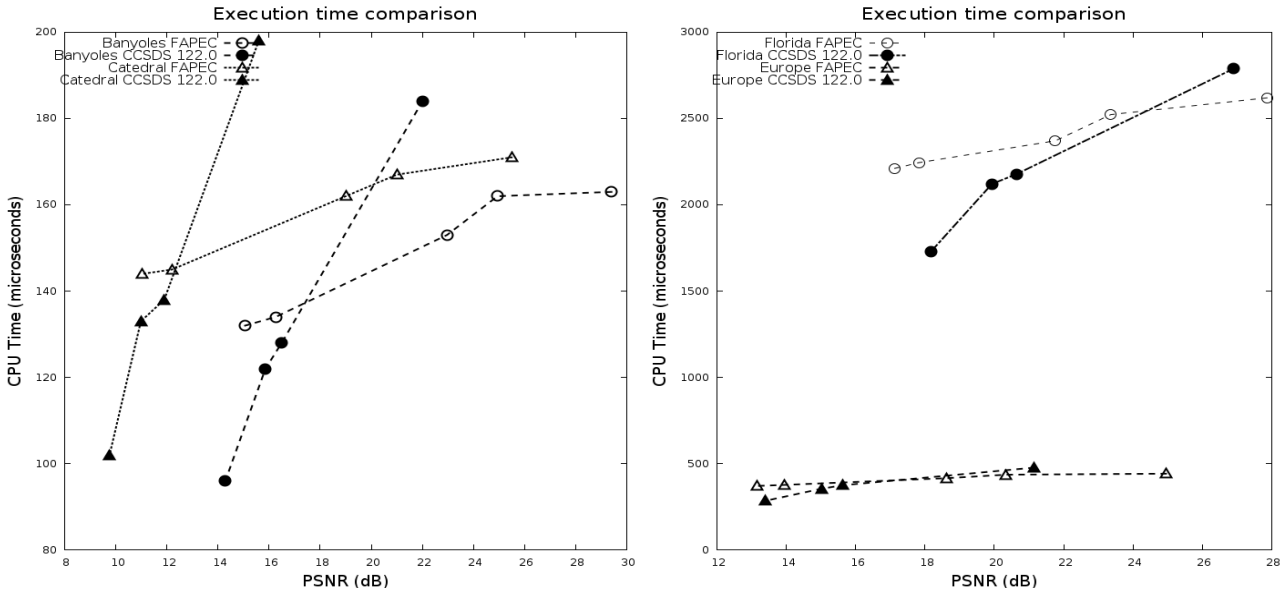


Figure 5: Comparison of the Execution time between Lossy FAPEC and CCSDS 122.0 for normal size and big image

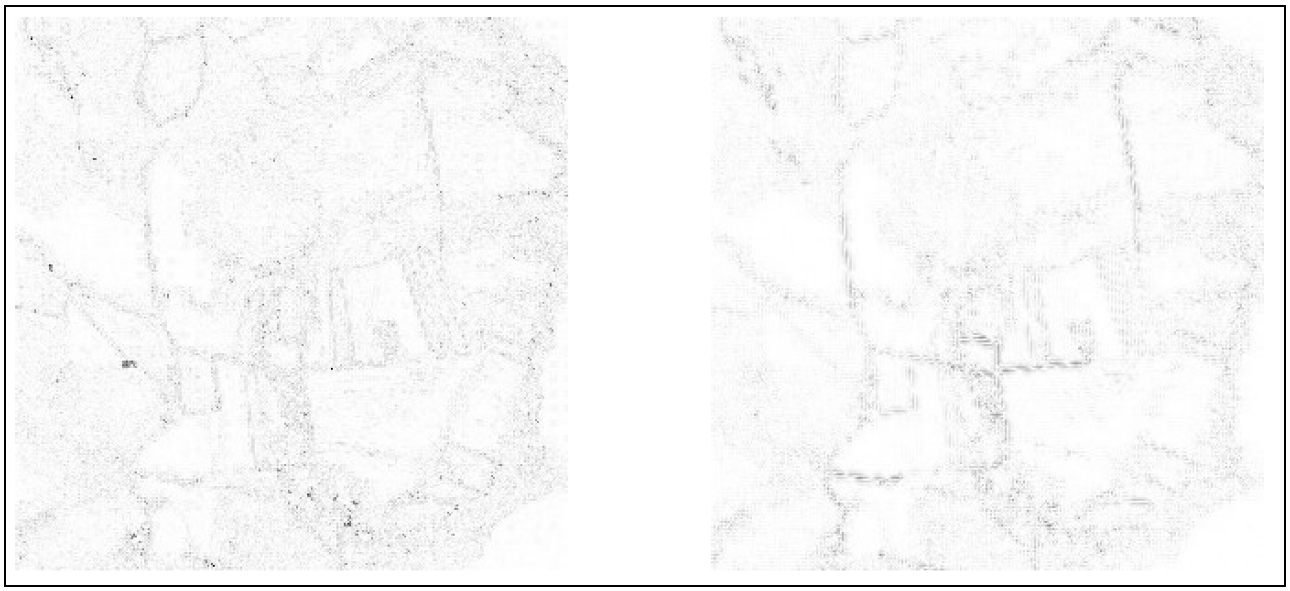

Figure 6: Error map showing the reconstruction errors after lossy compression with HPA+FAPEC (left panel) and CCSDS 122.0 (right panel) for the *banyoles* image

## V. FUTURE IMPROVEMENTS

The HPA algorithm and its HPI variant, despite of still being prototypes, have shown a great potential. Nevertheless there is much space for improvements. So far, the image dimensions are limited to multiples of 16 pixels on both dimensions, so a first improvement is to overcome this limitation and make HPA able to process images of any size. Additionally, the algorithm is only able to handle greyscale images. It is foreseen a more versatile implementation to compress colour and even hyperspectral images. Besides, the algorithm needs a final optimization which will reduce the execution time needed to compress an image. We also envisage improvements in the FAPEC core to improve the compression ratio for very low entropy data, which should improve the ratios achieved with HPA. Finally, regarding the

lossy compression, a feature that would improve the algorithm usability is a logic which, given a desired $C_R$ (or bits per pixel), the adequate QL is adaptively selected. This means to add a control that allows the HPA to compress not only with fixed quality, but also with fixed compression ratio or bit rate.

## VI. CONCLUSIONS

In this paper we have introduced a new image processing algorithm called Hierarchical Pixel Averaging (HPA). It has been designed as a new pre-processing stage for FAPEC, specific for images. The goal was to obtain an efficient image compression algorithm allowing to progressively move from lossless compression to lossy compression with different quality levels.

The HPA implementation has been kept as simple as possible, avoiding floating-point operation in order to make it more suitable to be embedded on satellite payloads. We must also note that we have taken this approach (instead of the typical wavelet-based or transformation-based approaches) in order to minimize the artefacts that can be seen in the restored images when they are lossy compressed with very low quality levels.

The results, when compared to the CCSDS 122.0 standard, shows that the HPA+FAPEC combination can compress an image significantly faster (27% on average) in the lossless scheme. The $C_R$ is slightly higher than with FAPEC alone, but still lower than lossless CCSDS 122.0. In the lossy scheme, HPA is able to achieve higher compression ratios and PSNR values. The quality of the recovered images are very good, and one of the interesting results is that the restored images after lossy HPA keep most of the sharp image edges, whereas the restored images after lossy 122.0 present a significant smoothing in such edges.

Summarizing, this new image compression approach appears as a very interesting alternative to the current image data compression standard for satellite payloads.

## VII. ACKNOWLEDGEMENTS

The images shown in this paper were obtained from websites of GOES-NOAA and EUMETSAT. This work was supported by the MINECO (Spanish Ministry of Economy) - FEDER through grant AYA2009-14648-C02-01, AYA2010-12176-E, AYA2012-39551-C02-01 and CONSOLIDER CSD2007-00050.

## REFERENCES

[1] Consultative Committee for Space Data Systems, "Image Data Compression, Blue Book Technical Report CCSDS 122.0-B-1", CCSDS, 2005.

[2] Artigues, B., Portell, J., Villafranca, A.G., Ahmadloo, H., García-Berro, E.: *DWTFAPEC: image data compression based on CCSDS 122.0 and fully adaptive prediction error coder*. Journal of Applied Remote Sensing (2013) 7 (1), 074592, SPIE.

[3] Portell, J., Villafranca, A.G., García-Berro, E.: *Quick outlier-resilient entropy coder for space missions*. Journal of Applied Remote Sensing (2010) 4, 041784, SPIE.

[4] Consultative Committee for Space Data Systems, Lossless Data Compression, Blue Book. Technical Report CCSDS 121.0-B-1, CCSDS, 1993.