Designing optimum solutions for lossless data compression in space

Jordi Portell[1,2], Alberto G. Villafranca[2], Enrique García-Berro[3,2]

[1] Astronomy and Meteorology Department, University of Barcelona
C/ Martí i Franquès s/n, 08028 Barcelona, Spain
Email: jportell@am.ub.es

[2] Institute for Space Studies of Catalonia (IEEC)
C/ Gran Capità 2-4, 08034 Barcelona, Spain
Email: agonzalez@fa.upc.edu

[3] Department de Física Aplicada, Universitat Politècnica de Catalunya
Escola Politècnica Superior de Castelldefels, Avda. del Canal Olímpic s/n, 08860 Castelldefels, Spain
Email: garcia@fa.upc.edu

ABSTRACT

The standard solutions currently available for lossless data compression have difficulties in fulfilling the tight requirements of modern space missions. High compression ratios are often needed, while the available processing power is usually small. Adaptive algorithms requiring large amounts of data for their optimum operation cannot be used in most cases, due not only to their high processing requirements but also to the limited reliability of the communications channel. We describe a method for pushing the information theory to its limit while guaranteeing a reliable downlink, achieving high compression ratios at a low processing cost. This is done using a two-stage compressor, namely, an adequate pre-processing stage followed by an entropy coder. While the pre-processor should be tailored for each case, we have developed a new coder which offers a robust operation in front of noisy data, which has comparable or even better performances than those offered by the current CCSDS standard.

Keywords: Lossless data compression, noisy data, pre-processing, coding algorithm, CCSDS, Rice

I. INTRODUCTION

The currently available algorithms for lossless data compression can be classified in two major groups, namely, statistical coding and dictionary coding [1]. Huffman [2], Rice-Golomb [3,4] and arithmetic [5] coding algorithms belong to the first group, while solutions based on Lempel-Ziv methods [6] or the Burrows-Wheeler transform [7] belong to the latter. Statistical coding basically generates a variable-length code for each symbol to be compressed (except in the case of arithmetic coding). It limits the maximum achievable compression ratio to the symbol size. Despite of this, statistical coders represent a powerful enough and simple solution for data compression. Dictionary-based coders, on the other hand, assign a single code to a set of symbols, and hence can offer much higher compression ratios than their statistical competitors. The counterpart is that they require larger block sizes in order to achieve the expected performances. Additionally, their processing requirements are usually larger.

Data compression systems for satellite payloads have tight restrictions, being the first one the block size of the compressor. Due to the nature of the communications channel, data compression must be applied to independent blocks of data with at most some kilobytes of length [8]. It contradicts with the fact that most of the adaptive data compression systems perform optimally only after a large amount of data is processed. The combination of data coming from different instruments, leading to non-uniformities in the data stream, also decreases the compression ratios of such adaptive systems. Moreover, on-board processing requirements – in the case of software implementations – are rather tight. Thus, low-complexity algorithms are desirable. All in all, it means that the most usual data compression solutions typically used on ground cannot be applied to space data systems. The Consultative Committee for Space Data Systems (CCSDS) recommends the usage of two stages for lossless data compression [9], namely, a pre-processing stage followed by an entropy coder. While the recommendation does not strictly specify the pre-processing stage, the coder is based on the Rice algorithm [3]. The system operates with blocks of just 8 or 16 samples, analyzing them and deciding the best coding option. Hence, this is an adaptive system but without the typical restrictions of ground-based algorithms. The CCSDS recommendation has been used in several missions during the last years [10], including hardware implementations [11]. Despite of the powerful features of this standard compression system it is not exempt of problems. First of all, it is clear that an adequate pre-processing stage must be defined for each case. It means that some tailoring has to be done for each case in order to achieve the highest compression ratios. Another problem arises at the

coding stage, because the Rice algorithm is not intended for the compression of noisy data. Actually, its efficiency significantly decreases with the amount of noise included in the data. Since most space-based measurements are contaminated with noise and outliers, it turns out that the CCSDS recommendation is not an optimum solution for most of the cases.

In this work we propose a strategy for designing optimum lossless data compression systems. In Sect. II we briefly review some basic concepts, and we also present the context of our study. Sect. III illustrates the first steps to follow when designing such a system. In Sect. IV we explain how to design and prototype the system itself. A new coding algorithm is presented here as well. Sect. V illustrates some methods to validate and assess the design. Finally, in Sect. VI we summarize our major findings and we draw our conclusions.

II. SCIENTIFIC CONTEXT

A. Basic concepts

The two-stage data compression strategy is an otherwise typical approach used by several systems [1]. It is partly due to the availability of many coding algorithms. However, the main reason is that the introduction of a relatively simple pre-processing stage, adapted to the data to be compressed, can easily boost the final compression ratios. Fig. 1 illustrates a simple pre-processing stage based on a predictor followed by a differentiator. The predictor estimates the value of a sample from the previous samples combined with other complementary data, such as instrument models or the sampling configuration. Afterwards, the difference between this estimation and the actual value of the sample is output. That is, such pre-processing stage outputs *prediction errors*. We highly recommend the usage of data prediction in the pre-processing stage, since it makes possible the generation of the lowest possible values and, from that, the achievement of larger compression ratios. The goal of the pre-processing stage is to generate a data stream with the lowest possible entropy. Once such data stream is available, the second stage must generate an adequate variable-length code (VLC) for each prediction error. It is usually implemented in the form of an *entropy coder* [1], that is, a statistical coder. The objective is to generate shorter codes for the most probable values and vice versa. In this way, the overall size of the output stream will decrease. While several solutions of entropy coders exist, the pre-processing stage must be specifically designed for each case. Actually, a separate and tailored pre-processing stage must be designed for each type of data to be compressed if the highest compression ratios are to be obtained [8]. This is especially important for payloads combining data from different instruments into a single data stream. In this way, the adequate data predictors for each data type can be used. The results of the pre-processing stage must be evaluated in order to verify that it really decreases the data stream entropy. This is done by computing the entropy:

$$H = \sum_i P(i) \log_2 \frac{1}{P(i)} \tag{1}$$

where *P(i)* are the different symbol probabilities. This quantitative measurement of the performance of the pre-processing stage must obviously be calculated for each design, that is, for each of the resulting *sub-streams* of different data kinds, as well as for the original data prior to the pre-processing. The results will reveal if the first stage is behaving optimally. In fact, the goal of the pre-processor is to achieve an entropy equivalent to the noise level and intrinsic uncertainties of the measurements. Also, the maximum theoretical compression ratio that can be achieved with an entropy coder applied to the resulting data stream can be computed. This is the Shannon Limit, which can be easily calculated from the computed entropy and the original symbol size (for example, 16 bits) and indicates whether or not the requirements on the compression ratio can be fulfilled:

$$ShLim = \frac{SymbolSize}{H} \tag{2}$$

Additionally, the pre-processing stage should be *qualitatively* evaluated. It can be done using histograms of the data before and after the pre-processing stage. A histogram can reveal, for example, offsets in the measurements or wrong predictions. As an example, in Fig. 2 we show a Gaussian and a Laplacian distribution, while practical examples will be shown later. Note the logarithmic scale in the abscissa, which helps in better appreciating the prediction errors obtained

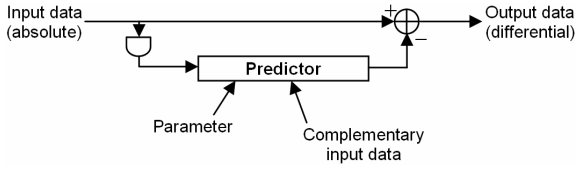


Fig. 1. Generic pre-processing stage based on data prediction.

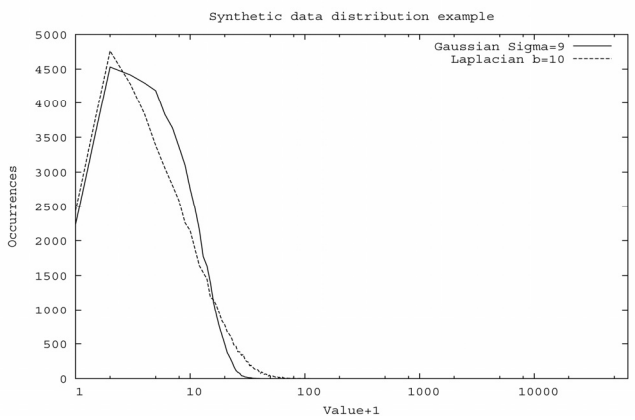


Fig. 2. Histograms for Gaussian and Laplacian distributions.

from the pre-processing stage. Also, only the absolute values (modulii) of the prediction errors are shown, which should be enough in most of the cases – considering a coding of the sign with a separate bit. It leads to an apparent decrease in the number of occurrences of the '0' value. This is due to the fact that occurrences of non-zero values include both positive and negative occurrences. In case a value mapping is applied after the pre-processing stage (such as PEM, in the CCSDS recommendation), the resulting histogram and entropy should be evaluated as well. We do not recommend such mapping because a separate sign bit can be useful in some cases. Additionally, a value mapping can degrade the entropy achieved by the pre-processor.

In some cases, an adequate pre-processing stage followed by one of the already available coding algorithms in the second stage can be enough. Otherwise, an adaptation of some coding algorithm or even the development of a new one will be required. Whichever option is chosen, the efficiency of the coder must be evaluated as well:

$$Eff = \frac{H_{pre-proc}}{AvgSymbolSize} \tag{3}$$

where the average symbol size output by the coder must be obtained with simulations, an $H_{pre\text{-}proc}$ is the output entropy of the pre-processor. It is clear that the target efficiency is 100%, this is, the Shannon Limit.

B. Background of the study

The strategy described in this paper for the design, development and testing of optimum lossless data compression systems has been followed in the case of the Gaia mission [12] with successful results. Gaia is an ambitious space observatory, adopted within the scientific programme of ESA and approved for a launch in late 2011. Gaia aims to measure the positions and proper motions of more than 10^9 stars with unprecedented accuracy. As a result, a three-dimensional chart of more than 1 percent of the stars of our Galaxy will be obtained, as well as Solar System objects and extragalactic sources. Gaia will be a technological challenge in all its aspects, and data compression is not an exception. A large and complex dataset will be generated by its instruments, including positional, imaging, photometric and spectroscopic data. The data stream of more than 10 Mbps in average must be compressed a ratio of about 2.5 without any losses. Previous studies [8,13], already following the two-stage approach, revealed promising results. Nevertheless, they did not take into account the tight restrictions in the payload, namely, very small data blocks and highly limited processing power. Other studies approached to these limitations, but were not tested under realistic scenarios including cosmic rays and realistic noise. For this reason, ESA released an invitation to tender for the design of an optimum data compression algorithm for Gaia. The telemetry packets had to be compressed one by one and independently, with compression ratios of 2.6 for most astrometric and photometric data and 2.5 for spectroscopic data. The system had to be completely lossless and the algorithm had to be software-implemented, consuming at most 5% of a CPU with about 1500 MIPS when compressing a data stream of about 5Mbps. GTD Sistemas de Información S.A., a Spanish software company, with the Institute for Space Studies of Catalonia as its scientific partner, won this competition. The resulting study, GOCA (Gaia Optimum Compression Algorithm), served as an assessment of the feasibility of such tight requirements of the mission [14].

III. STUDY OF THE DATA COMPRESSION PROBLEM

The first step for successfully designing an optimum data compression system is to study the data to be compressed. It can be done in different ways depending on the kind of data, but in general a good option is to plot the stream of values

as received by the compressor. This reveals some simple structures or patterns in the data, for example incremental measurements or small variations, thus indicating that a simple (hence quick) pre-processing stage should be enough. If that is not the case, the next step is the analysis of the sample values as a function of some measurement parameter, for example, the star magnitude in the case of Gaia. Also, the representation of the values after rearranging them in some adequate two-dimensional structure could help, as in the case of images. For the case of Gaia, we show some examples below. They have been obtained from GIBIS, the Gaia Instrument and Basic Image Simulator [15,16].

A. Imaging data

Fig. 3 illustrates the data stream obtained from the narrow-field images acquired by some instruments of Gaia, namely, the Sky Mapper (SM) and the Astrometric Field (AF). The SM covers about 10 square arcseconds with 20×3 or 40×6 samples. It leads to a large fraction of dark samples, while the bright ones obviously depend on the star magnitude. The AF measurements cover at most 2.2 arcsec^2 with 18×12 samples or less, hence offering much more resolution. As a result, just a few samples are actually dark, while most of them follow the response of the instrument. That is, the image is the convolution of the true image of the star with the Point Spread Function (PSF). It is worth noting that the AF measurements will be performed 9 times for each star, as shown in the right panel of Fig. 3. It will lead to very similar images, although their high resolution and the imperfections of the instrument will obviously lead to some differences.

B. Repeated measurements

To make possible the on-board management and downlink of the huge amount of measurements that Gaia will perform, the size and resolution of the images is decreased for the faintest stars, which are the most numerous. More specifically, the pixels are *binned* (or summed) in one direction, thus leading to a profile or shape of the star – better said, of its PSF. With the adequate application of a centroiding algorithm in the on-ground data processing systems it will allow to determine the positions of the stars with the highest accuracy. Such positional measurements are also performed 9 times for each star. Again, it will lead to very similar (but not identical) data sets. The right panel of Fig. 3 illustrates an observation.

C. Data from dispersive optics

The last set of data generated by Gaia corresponds to observations done using the blue and red photometers (BP and RP), which obtain high-resolution photometry in such colour bands, and using the radial velocity spectrometer (RVS), which obtains medium and high resolution spectra of the sources. Fig. 4 illustrates some of these measurements. Three RVS measurements are acquired for each star. Note that the RVS samples are significantly contaminated by noise.

IV. SYSTEM DESIGN

A. Pre-processing stage

Once the kind of data to be compressed is known, the design of the pre-processing stage can be done. We recall that the goal is to obtain a stream of the smallest possible values, usually by means of some kind of data prediction. We start with the imaging data, and more specifically with the SM data. As previously mentioned, most of its sample can be safely predicted as *dark*. Thus, a differential processing could be applied. That is, the difference between consecutive samples is output. Nevertheless, in the case of the SM of Gaia, tests applied to simulated (but realistic) data have revealed that it leads to worse results than predicting most samples as dark, most probably due to the measurement noise. Regarding the bright (central) samples, there is a strong dependence with the star magnitude and the neighbour

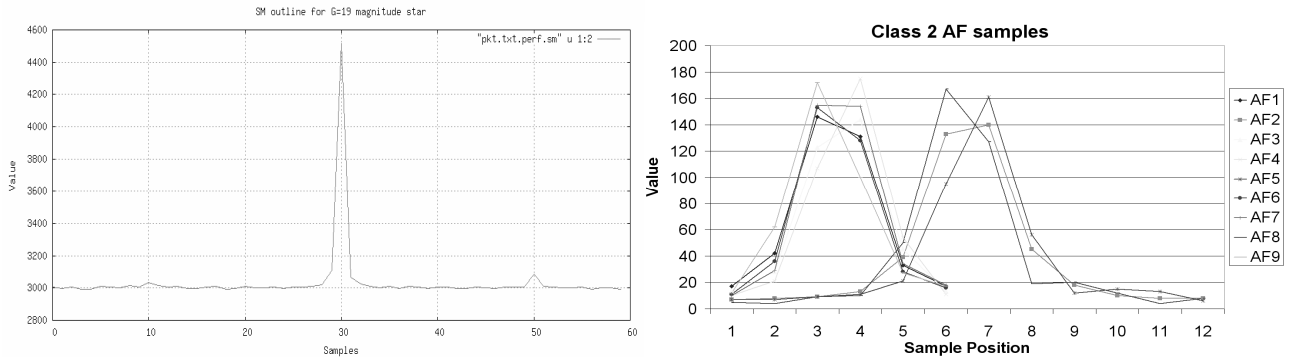


Fig. 3. Data streams from the SM (left panel) and from the nine AF for faint stars (right panel).

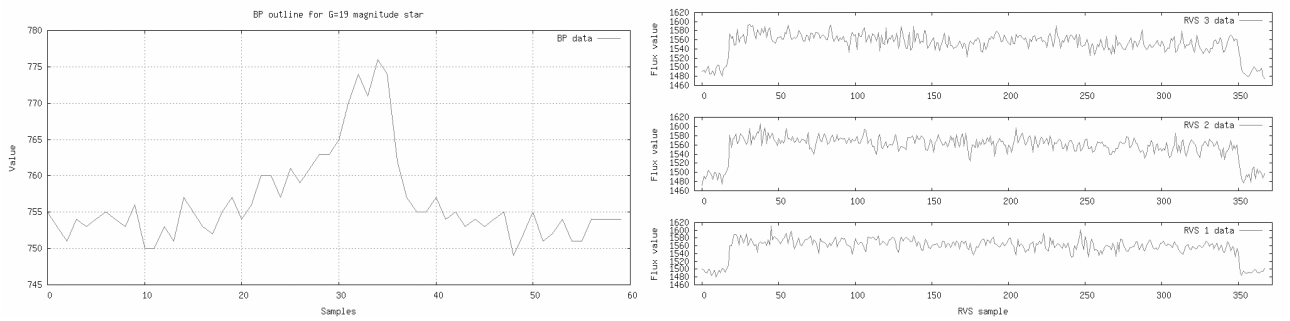


Fig. 4. BP (left panel) and RVS (right panel) measurements acquired by Gaia.

samples. Since the telemetry packets include a header that indicates the star magnitude as estimated on-board, such SM samples can be safely predicted using it.

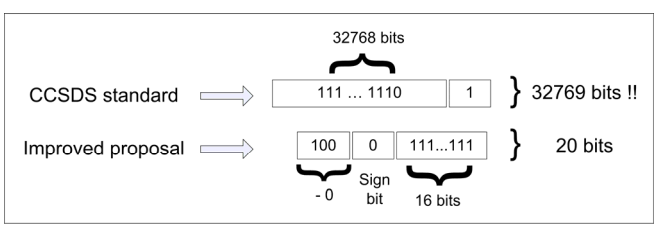
The case of the images obtained by the AF for bright stars is slightly simpler. Since they are much smoother than SM images, a pixel (or sample) can be predicted from the previous one. Thus, a simple differential processing is applied, although some corrections are applied taking into account the repetitions of such AF measurement. Regarding the multiple AF measurements acquired for faint stars, the pre-processing is more complex due to the small amount of data acquired in each measurement. First of all, the first measurement or AF1 is predicted again from the estimated star magnitude. For the rest of AF measurements, a quick shape analysis is performed, grouping the profiles as *wide* or *steep*. Next, the peak of each measurement is found, thus making possible their correct alignment. Finally, the differences between a measurement and its "best reference" (taking into account its shape) are output.

Data from BP and RP can be processed in a very simple way, just outputting the differences between consecutive samples. Due to their smooth shapes, it leads to low prediction errors. Some correction is also applied to the brightest samples, using again the star magnitude. Finally, RVS data is more complex to process due to the large amount of noise. Noise cannot be compressed and, hence, its level will limit the minimum entropy achievable. What can be done in this case is to filter the noise in the prediction loop. In this way, large variations in the data stream caused by noise can be significantly reduced. A retroactive loop, compensating the current prediction with the two previous ones, is also used. In this way the continuum of the spectrum is estimated, predicting all the samples from it.

B. Coding stage: the Prediction Error Coder

Although the CCSDS recommendation does not require a large processing power, its implementation in Gaia is not feasible due to CPU restrictions. For this reason, the use of a *trained* Rice coder was recommended by the main contractor, using a fixed value of k for a given data type [3], to be determined with calibration datasets. This solution drastically decreases the CPU requirements, but an unacceptable risk appears. Any significant deviation of the data received by the compressor with respect to the calibration dataset would lead to a *large* number of bits output. This is illustrated in the left panel of Fig. 5, where a saturated 16-bit value is coded with k=1. This situation can occur frequently. Fortunately, it can be mitigated modifying the signed Rice coder. If the sign is coded with a separate bit, the "−0" code can be used as an escape sequence. The reception of such sequence by the decoder indicates that a special code follows, more specifically, the coding of the data in raw 16-bit. This *Limited-Length Signed Rice Coder* devised by our group solves the problem, as illustrated in the left panel of Fig. 5, where the original data is only slightly expanded.

Although the Limited-Length Signed Rice Coder already led to fairly acceptable ratios, we looked for a really optimum solution for Gaia. Its measurements are severely contaminated not only with noise, but also with outliers due to cosmic rays and energetic particle impacts on its instruments. For this reason, a much more robust coder was necessary, not only to guarantee a limited code length for outliers but also to minimize the expansion ratio of lower-level outliers. The



Segment size	4	6	7	16
Segment range	0-14	15-78	79-206	207-65535

1st segment ⇒ 4 bits range + 1 sign bit = 5 bits
2nd segment ⇒ 6 bits range + 1 sign bit + 4 prev. range = 11 bits
3rd segment ⇒ 7 bits range + 5 bits {-0} + 1 sign bit + 1 flag = 14 bits
4th segment ⇒ 16 bits range + 5 bits {-0} + 1 sign bit + 1 flag = 23 bits

Fig. 5. Resilient entropy coders: Limited-Length Signed Rice Coder (LLSRC) and Prediction Error Coder (PEC).

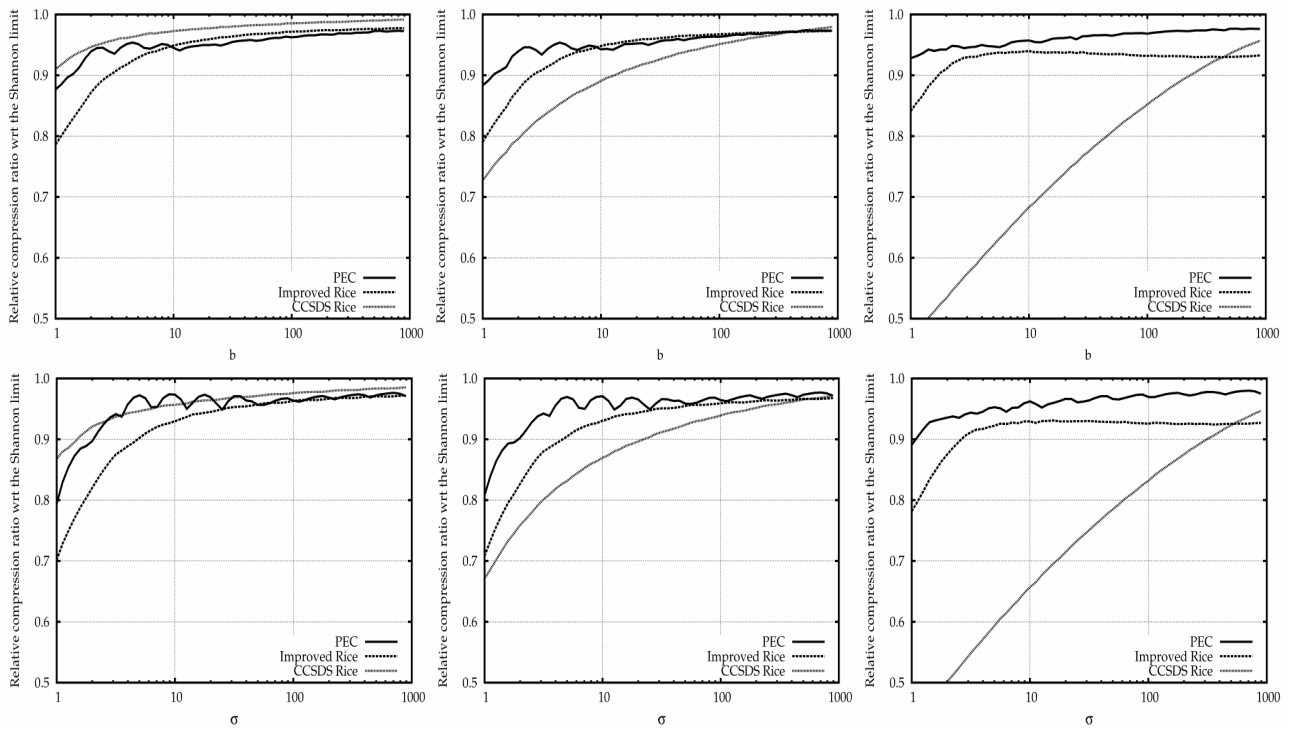


Fig. 6. Efficiencies of the CCSDS, Improved Rice and PEC coders with respect to the Shannon limit, for Laplacian (top panels) and Gaussian (bottom panels) distributions. From left to right: data sets with 0.1%, 1% and 10% uniform noise.

result was the Prediction Error Coder (PEC), a new coding algorithm specifically designed for prediction errors. It can be considered a semi-adaptive coder, owing to its operation based on coding segments. Its operation is shown in the right panel of Fig. 5. This coder must also be calibrated with a representative dataset. This calibration leads to a *coding table*, indicating the sizes of the several code segments. Afterwards, PEC uses the adequate segment for each of the values received, thus generating small code lengths for low values. When receiving larger values, one or more segments are concatenated. The codes for the largest values are *smoothed* by using the "−0" code again. We have done several tests of the performance of PEC with respect to the CCSDS standard and the Limited-Length Signed Rice Coder. These tests have been done using synthetic data reproducing Laplacian and Gaussian distributions. Also, different noise levels have been tested. The results are shown in Fig. 6 are and are self-explanatory. As the noise contribution increases, the CCSDS standard rapidly decreases its performance with respect to the Shannon limit. On the other hand, the improved Rice coder keeps offering a good efficiency, although the best results are definitely obtained with PEC. Regarding the processing requirements, PEC is twice faster than its competitors.

C. Implementation of prototypes

A realistic implementation of the data compression system is necessary to assess its feasibility regarding the on-board processing requirements. For this, a modular approach is the best option. The several elements of the data compressor must be implemented as separate routines, including the pre-processing and coding stages and the input/output routines as well. Also, an adequate processing manager reproducing the on-board system calling the compressor routine helps in obtaining reliable results. Finally, several statistics must be collected during the operation of the compressor, including the compression ratios for each telemetry packet, the average ratios for different amounts of packets, and the processing times. Some examples of the validations that can be done with such routines are illustrated below.

V. EVALUATION OF RESULTS

Once the pre-processing and coding stages have been designed and adequately implemented, they must be tested to find any possible weakness. There are several ways to do this, being the simpler and straightforward one a comparison of the input and output file sizes. It will indicate the total compression ratio, averaged for all the telemetry packets and scenarios. But such validation is definitely not enough. Firstly, we must verify separately if each stage performs properly. Then, the short-term ratios achieved for different scenarios must be checked. The following subsections illustrate the kind of validations that could be done for any lossless data compression system.

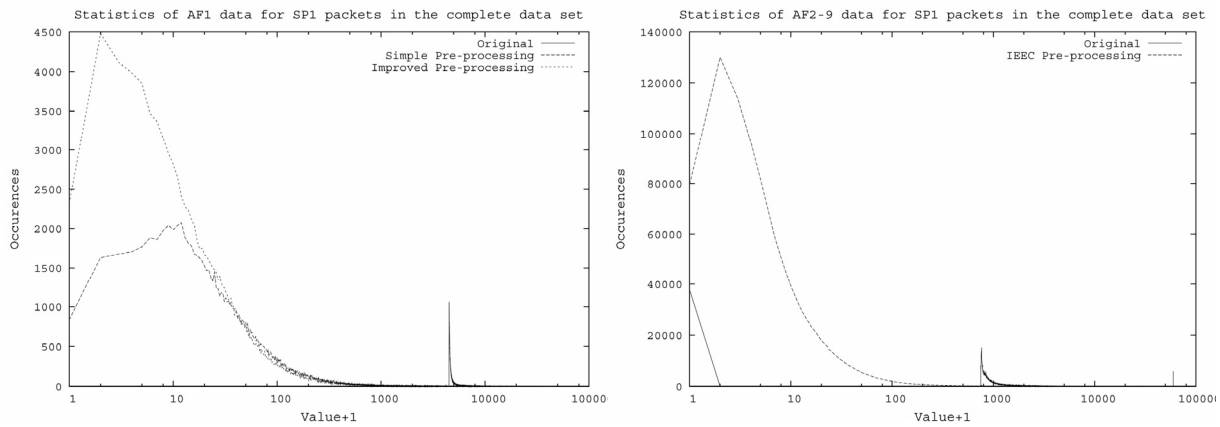


Fig. 7. Histograms of AF1 data (left panel) and AF2-9 data (right panel), before and after the pre-processing stage.

A. Pre-processing results

Fig. 7 illustrates the improvements obtained with the pre-processing stage of GOCA. More specifically, it shows the results achieved for AF1 and AF2-9, which are the repeated positional measurements (with AF1 serving as the reference measurement). The left panel also illustrates the difference between a naïve and a complex pre-processing. It is fairly easy to account for measurement offsets by means of a simple differentiator. This already improves the results, leading to fewer bits generated by the entropy coder in the second stage. Nevertheless, an improved pre-processing stage makes the histogram much steeper, increasing the probability peak for smaller values and, hence, reducing the entropy. The right panel illustrates the excellent results achieved with the quick analysis of shapes and peak positions of the repeated measurements. Numerically, the Shannon limit obtained for AF1 increased from just 1.50 to 1.75, while it increased from 1.48 to 2.02 for AF2-9. For the spectroscopic data, the filtering stage increased the Shannon limit from 1.95 to 2.50, that is, just 6.4 bits are required on average. Sky Mapper images could be compressed up to 2.43. The most impressive quantitative improvement, nevertheless, was achieved for the photometric data, reaching 3.33 for BP and 3.19 for RP, while their original entropy values were leading to limits of 2.40 and 2.10.

B. Total ratios

The efficiency of the coding stage can be evaluated by comparing the input and output sizes of the telemetry packets or data blocks. It should be done separately for each type of data, so that Eq. (3) can be applied. Finally, the total results must be evaluated, that is, combining the two stages. The left panel of Fig. 8 illustrates the average ratios obtained on the astrometric and photometric data of faint stars in Gaia. There are many packets for which the requirement of 2.6 is fulfilled, corresponding to the nominal operation mode of Gaia. The other packets, reaching ratios around 2.2 on average, correspond to the maximum load, with about 1400 observations per second. In these cases, the instrumentation of Gaia restricts the measurements in order to avoid saturation of its Video Processing Units. As a result, some telemetry packets are truncated or have the BP and RP measurements removed. Fortunately, Gaia will operate under such circumstances during a small fraction of the life of the mission. Additionally, the right panel of Fig. 8 illustrates another result that helps detecting weaknesses in the system. This is the analysis of the ratios achieved as a function of a measurement feature. We show a two-dimensional histogram of the compression ratio as a function of the star magnitude. The ratios increase for fainter stars, an otherwise expected result. Note that there is a set of packets around magnitude 19th for which the ratios are below 2, corresponding to the maximum star densities previously commented.

The final results were very satisfactory, fulfilling most of the requirements with margin. SP1 packets for faint stars are compressed about 2.73, while the requirement was 2.6. SP2 packets for faint stars are compressed about 2.56, whereas the requirement was 2.5. SP1 packets for mid-bright and bright stars, representing about 10% of the observations, are compressed by 1.94 and 2.17 respectively. Although the requirement of 2.0 is not fulfilled for mid-bright stars, this is compensated by the larger ratios obtained for faint stars. The prototype used less than 10% of CPU in a representative platform, which was later confirmed with an instrument prototype.

VI. CONCLUSIONS

We have proposed a strategy and different tools for designing, implementing and validating an optimum lossless data compression system for payloads on board space missions. Such a strategy simplifies the problem by dividing it in two

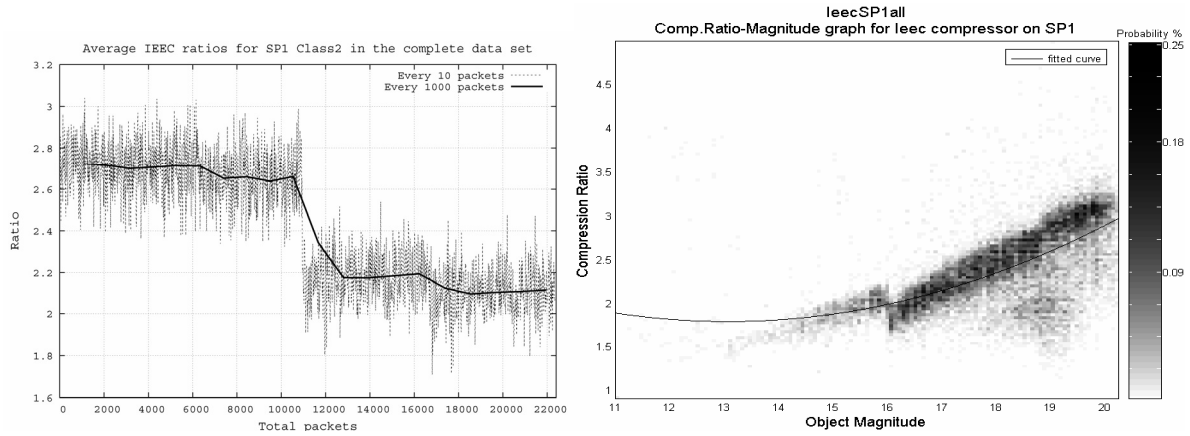


Fig. 13. Total compression ratios obtained over different averages of telemetry packets (left panel), and as a function of the star magnitude (right panel).

stages. In this way the best compromise between high compression ratios, simplicity, and robustness is obtained. The first stage consists in pre-processing the data. Consequently, it must be adapted to the specific payload. The pre-processing is based on data predictors, thus generating prediction errors. We look for much steeper histograms at the output of the pre-processor. The Rice algorithm, used by the current CCSDS standard, performs very well under almost ideal circumstances. Nevertheless, noise and outliers can lead to prohibitive code lengths at the output of this stage, thus expanding the data. We have improved this algorithm, making use of the "−0" code as an escape sequence. Simulations reveal that this approach is robust when unexpected data and noise are present. We have gone one step further, devising the Prediction Error Coder (PEC). This new semi-adaptive algorithm outperforms any variant of Rice under extreme situations, taking advantage of any redundancy of the data despite of the presence of large amounts of noise and outliers, while its performance under nominal conditions is very similar to Rice. The two-stage approach with tailored pre-processors and PEC, as well as the validation tools explained here, have been applied to the case of GOCA, an optimum compression study for the Gaia mission. The excellent results obtained demonstrate the usefulness of this solution, with large compression ratios while keeping a very modest CPU usage. Other space missions currently in their design or conception phase, as well as future missions, would benefit of this solution.

REFERENCES

[1] D. Solomon, *Data Compression, the Complete Reference*. Springer-Verlag (2004)

[2] D. Huffman, "A method for the construction of Minimum Redundancy Codes", *Proceedings of the IRE*, vol. 40(9), pp. 1098–1101 (1952)

[3] Robert F. Rice, "Some practical universal lossless coding techniques", Tech. Rep. (1979)

[4] S. W. Golomb, "Run-lengths encodings," IEEE *Transactions on Information Theory*, vol. 12(3), pp. 399–401 (1966)

[5] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30(6), pp. 520–540 (1987)

[6] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23(3), pp. 337–343 (1977)

[7] M. Burrows and D. J. Wheeler, "A Block-Sorting Lossless Data Compression Algorithm", Digital Systems Research Center report 124, Palo Alto, California (1994)

[8] J. Portell, E. García-Berro, X. Luri, and A. G. Villafranca, "Tailored data compression using stream partitioning and prediction: application to Gaia," *Experimental Astronomy*, vol. 21, no. 3, pp. 125–149 (2006)

[9] CCSDS, "Lossless data compression, blue book," Tech. Rep., 121.0-B-1, (1993)

[10] P.-S. Yeh, "Implementation of CCSDS lossless data compression for space and data archive applications," Tech. Rep. (2002)

[11] R. Vitulli, "PRDC: an ASIC device for lossless data compression implementing the Rice algorithm", *IEEE International*, (2004)

[12] Perryman,M.A.C., de Boer, K.S., Gilmore, G., Hoeg, E., Lattanzi, M.G., Lindegren, L., Luri, X., Mignard, F., Pace, O., de Zeeuw, P.T.: A&A, 369, 339 (2001)

[13] J. Portell, "Payload data handling, telemetry and data compression systems for Gaia", Ph.D. Thesis, (2005)

[14] GTD and IEEC, "Gaia Optimum Compression Algorithm", Summary Report, ESTEC contract 20615/07/NL/LvH (2008)

[15] C. Babusiaux, "The Gaia Instrument and Basic Image Simulator", *Proceedings of the Symposium "The Three-Dimensional Universe with Gaia"*, ESA-SP-576, pp. 417–418 (2005)

[16] X. Luri, C. Babusiaux and E. Masana, "Modelling the instruments and simulating the data stream", *Proceedings of the Symposium "The Three-Dimensional Universe with Gaia"*, ESA-SP-576, pp. 357–358 (2005)