EFFICIENT DATA COMPRESSION FOR SPACECRAFT INCLUDING PLANETARY PROBES

M. Cabral[(1)], R. Trautner[(1)], R. Vitulli[(1)], C. Monteleone[(2)]

[(1)]TEC-EDP, ESA/ESTEC, Noordwijk, The Netherlands
[(2)] TEC-EDD, ESA/ESTEC, Noordwijk, The Netherlands
Email: Roland.Trautner@esa.int

ABSTRACT

For all types of space missions, the available bandwidth for the transmission of data back to Earth is an important constraint with significant impact on vehicle design, onboard resources, and mission operations. The efficient compression of science and housekeeping data allows maximizing the utilization of available bandwidth and onboard resources.

While data compression is traditionally used for science payload data, the compression of spacecraft housekeeping data is becoming increasingly attractive in particular for exploration and science missions that have significant bandwidth constraints.

We present a summary of present and upcoming standardized data compression algorithms for on-board implementation, including ESA's latest related developments such as a new implementation of the CCSDS-122.0 (image data compression) standard, performant multispectral data compression algorithms, and research on simple pre-processing steps which improve the compression performance in particular for packetized data such as spacecraft housekeeping data. Test results for various compression algorithms are presented, and a new software tool is introduced which supports the evaluation of standardized data compression algorithms by industry and by payload teams.

1. INTRODUCTION

The reduction of redundancy / information content in data by means of data compression is an important technology, with widespread use in many application fields. Space missions were among the early adopters of data compression, with applications already implemented in the days of the Apollo program [1].

Due to the associated reduction of on-board data storage capacity and downlink bandwidth requirements, data compression is traditionally used in particular for science payload data. Recent European space missions like Huygens [2], Mars Express [3], or Venus Express [4] have implemented data compression for high bandwidth type instruments such as cameras and spectrometers. However, none of these missions used compression for instrument and spacecraft house-keeping data mainly due to conservatism in engineering approaches. This area is addressed by recent work [5] and it is expected that compression for platform data will be more common in the future.

Data compression can be implemented in hardware or software. The decision on the implementation platform, and the selection of a suitable algorithm, is based on a tradeoff of a number of factors, such as:

- Compression performance
- Lossy versus lossless compression
- Associated hardware requirements
- Associated software requirements
- Impact on system complexity
- Impact on reliability, including data integrity
- Compatibility with TM and ground systems
- Implementation cost

In order to minimize the added complexity needed for the implementation, algorithms are required that provide high compression performance with minimum consumption of on-board resources. All compressed data needs to be as robust as possible for minimizing the propagation of errors in the data. The impact on cost, reliability and system compatibility can be significantly reduced by standardization and re-use of efficient algorithms.

ESA has, in cooperation with major space agencies, supported the development and standardization of efficient data compression algorithms that provide high performance as well as low resource requirements. The standardization of algorithms supports the public availability of high quality documentation and the establishment of a broad user community. Algorithm specifications, test data, implementation guidelines and some software code are available via the services provided by CCSDS [6] and its members including ESA [7].

In the following chapters, data compression test hardware and test data sets are described. CCSDS standardized data compression algorithms and a standardization candidate for hyper/multispectral data compression are briefly introduced, and ESA implementations of the algorithms are presented, together with performance test results on a state of the art space qualified CPU [8]. The compression performance, memory requirements, processing efficiency, and error resilience are addressed. Based on test results, recommendations on the utilization of the individual algorithms are made.

A selection of simple pre-processing steps for packet telemetry data is introduced which allows achieving significant performance gains in combination with standardized compression algorithms. Test results for typical data sets are presented.

Finally, a new ESA software tool for the evaluation of standardized data compression algorithms by industry and by payload teams is introduced.

2. DATA COMPRESSION TEST HARDWARE AND DATA

The efficiency tests of the CCSDS 121.0 and 122.0 implementations were performed using a development board based on a SpaceWire Remote Terminal Controller (SpW-RTC) [8], [9]. This device includes an embedded LEON2 microprocessor, and a range of standard interfaces and resources (UARTs, timers, general purpose input output). The key specifications can be summarized as follows:

- RTC ASIC with a 50 MHz core clock speed
- 34 Dhrystone MIPS
- 16 Mbyte RAM, 16 Mbyte PROM, EEPROM
- SpW, CAN bus and serial interfaces

This computer board is representative to many contemporary on-board computers in terms of CPU architecture and performance.

Two types of images were used on these tests: images from the CCSDS image library [10] and completely black and random images, which provide a best and worst case for compressibility.

Four versions of black and random images were used, with different dimension and representation:

- 512x512x8 (512 by 512 pixels and 8 bits per pixel)
- 512x512x12
- 1024x1024x8
- 1024x1024x12

The four images selected from the CCSDS library have dimensions identical to the black and random images. Their filenames in the CCSDS test dataset are *marstest.raw, sun_spot.raw, b1.raw* and *solar.raw.*

3. CCSDS 121.0 – GENERAL PURPOSE LOSSLESS DATA COMPRESSION

The CCSDS 121.0 [11] standard is based on Rice coding, which was developed by Robert F. Rice at NASA. A lossless source coding technique preserves source data accuracy and removes redundancy in the data source. In the decoding process, the original data can be reconstructed from the compressed data by restoring the removed redundancy; the decompression process adds no distortion. This technique is particularly useful when data integrity cannot be compromised. The drawback is generally a lower compression ratio, which is defined as the ratio of the number of original uncompressed bits to the number of compressed bits including overhead bits necessary for signalling parameters.

The lossless Rice coder consists of two separate functional parts: the preprocessor and the adaptive entropy coder. Two of the factors contributing to the performance measure in the coding bit rate (bits/sample) of a lossless data compression technique are the amount of correlation removed among data samples in the preprocessing stage, and the coding efficiency of the entropy coder. The function of the preprocessor is to de-correlate data and reformat them into non-negative integers with the preferred probability distribution. The Adaptive Entropy Coder (AEC) includes a code selection function, which selects the coding option that performs best on the current block of samples. The selection is made on the basis of the number of bits that the selected option will use to code the current block of samples. An ID bit sequence specifies which option was used to encode the accompanying set of codewords.

Tables 1 and 2 show the performance of the CCSDS 121.0 lossless compression algorithm. They contain, respectively, the compression ratio and the execution time measured on the SpW-RTC. The tests were performed on the selected images available from [10].

Table 1. Compression ratios for test images using CCSDS 121.0 on the SpW-RTC

File ([width]x[height]x[bit depth])	Ratio
marstest.raw (512x512x8)	1,51
sun_spot.raw (512x512x12)	1,76
b1.raw (1024x1024x8)	2,18
solar.raw (1024x1024x12)	1,64

Table 2. Compression times for test images using CCSDS 121.0 on the SpW-RTC (ms)

	Black	Image	Random
512x512x8	619	1844	2202
512x512x12	637	2664	3540
1024x1024x8	2478	6454	8808
1024x1024x12	2548	10919	14160

It can be seen that black and random images provide a best and worst-case for execution time, with the real images having an intermediate value.

4. CCSDS 122.0 – IMAGE DATA COMPRESSION

The Image Data Compression (IDC) recommendation [12] has been designed for use in rockets, satellites, or spacecraft. It is expected that the technique is applicable to many types of instruments in such equipment. Therefore this coding system design is designed to satisfy all the memory and computation restrictions of this kind of equipment.

The IDC standard is designed for both lossless and lossy compression. When using lossless compression, the original image can be recovered exactly, while if lossy compression is used, approximations in different stages produce some loss of information that cannot be recovered even by transmitting the complete bit stream. The IDC technique is designed for monoband compression. The encoding process is divided in two functional parts. First, a Discrete Wavelet Transform (DWT) is performed in order to decorrelate the original data. Then, the transformed data are rearranged in 8×8 blocks. Blocks are grouped in segments. The number of blocks in a segment depends on the available memory to store the segment. Finally, each segment is independently encoded by the Bit Plane Encoder (BPE).

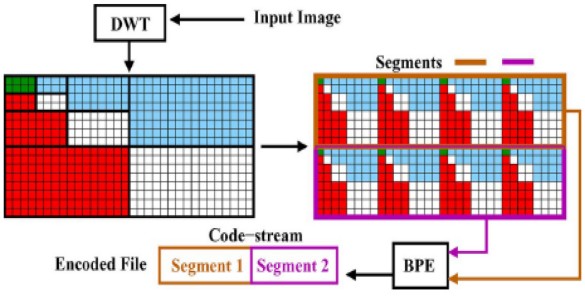

Fig. 1. IDC encoding process

Each 16 consecutive blocks within a segment are grouped conforming a gaggle. Blocks in a gaggle are entropy coded together. The number of blocks in a segment is usually selected using two criteria: the strip and the frame modes. The strip mode consists of selecting the number of blocks in a segment as the available blocks in a row. The frame mode consists of encoding all the blocks of the image in one single segment.

4.1 ESA's new algorithm implementation

Several implementations of the CCSDS 122.0 standard are already available [13]. However, among other problems, they typically require a lot of memory (at least 4 bytes for each pixel on the original image), can only read the input image from a file and output the compressed data to another file. These design features are not problematic when the software code is executed on a workstation, but they make it hard to port the application to a space platform.

Therefore, ESA has developed a new implementation of the standard. It is written in C and supports all the options defined in the CCSDS 122.0 standard. It uses less memory than previous implementations, and allows different input and output options such as files, memory, or network interfaces.

Memory usage

ESA's implementation uses a memory-efficient DWT calculation scheme, which significantly reduces the algorithm's memory requirements. We have characterized the memory usage of the implementation, and verified it using the *Massif* memory profiler. The required memory is divided into four main components:

1. Discrete Wavelet Transform (DWT) buffers
2. Buffer to store the transformed coefficents
3. Bit Plane Encoder (BPE) buffers
4. Other memory for variables and temporary buffers

The total memory usage M of the implementation can be calculated as

$$M = 207 * w + \frac{355 * S}{2} + 429 bytes \qquad (1)$$

where S is the segment size parameter and w is the image width in pixels, including padding (the CCSDS 122.0 standard requires images to be padded so that their width and height is a multiple of 8).

Note that this is a simplified formula, which only gives an exact value for values of S which are multiples of $w/8$. It is also a good approximation when $S \geq w/8$. For $S \leq w/8$, the memory usage will always be larger than that calculated using formula (1).

According to this, to compress an image of 512x512 pixels, the algorithm will require around 142Kb when a segment size of 128 is used. For a segment size of 1024 it would require around 525Kb. The memory requirements to compress a 1024x1024 image, using the same segment sizes, are respectively 229Kb and 612Kb.

Although these memory requirements are particular to our implementation, we believe that the value for any future implementations would be similar, as further optimizations of memory usage would most likely not be possible without a significantly lower execution speed.

Compression rate and efficiency

Table 3 shows the compression ratios obtained when compressing the selected images from the CCSDS library using our implementation of the CCSDS 122.0 standard. These values match very well with those reported in [13].

Table 3. Compression ratios using CCSDS 122.0 implementation (original size divided by compressed size)

File ([width]x[height]x[bit depth])	Ratio
marstest.raw (512x512x8)	1,67
sun_spot.raw (512x512x12)	2,07
b1.raw (1024x1024x8)	2,38
solar.raw (1024x1024x12)	1,93

Table 4 shows the compression times achieved using our implementation on the SpW-RTC, for both lossless and lossy compression, and using black images, random images and images from the CCSDS library. The results shown are for the same images as in Table 3 (*e.g.* the image represented on the *Image* column of the *512x512x12* line is *sun_spot.raw*).

Lossy compression was performed with a compression rate of 0.8 bits/pixel of the original image, *e.g.* an image with 512 by 512 pixels could have at most 512 * 512 * 0.8 bits.

Segment sizes of 64 and 128 were used for the 512x512 and 1024x1024 size images, respectively.

The *Black* and *Random* images provide a best and worst-case for the lossless compression execution time, with the realistic images having an intermediate value. However, this does not apply when lossy compression is performed. This happens because a fixed byte limit for the compression stream is set, which makes the algorithm halt when this limit is reached in each segment. The *Black* images compress so well that they do not reach this limit, making compression very fast. However, both the *Random* and real images reach this limit. The *Random* images have more entropy than the real ones, which means that they generate more information for each pixel of the original image, reaching the byte limit faster and making the algorithm stop sooner.

It can also be seen that the compression time is directly proportional to the number of pixels; when the resolution of the image doubles, the compression time increases fourfold. Also, the compression time depends on the image being compressed, as can be seen by the relatively small difference in compression time between the 512x512 images, in contrast with the large one for the 1024 ones.

Table 4. Compression times of CCSDS 122.0 implementation on the SpW-RTC (ms)

	Black	Image	Random
Lossless			
512x512x8	4462	11905	14068
512x512x12	4462	12941	17993
1024x1024x8	17872	37724	56092
1024x1024x12	17872	51280	71695
Lossy			
512x512x8	5052	7024	5776
512x512x12	5052	7494	5616
1024x1024x8	20247	27338	23375
1024x1024x12	20247	28051	22399

5. LOSSLESS IMAGE COMPRESSION COMPARISON

The CCSDS 122.0 standard allows both lossless and lossy compression of images. However, lossless image compression can also be performed using the CCSDS 121.0 general purpose lossless compression algorithm. This section compares these two alternatives, using images from the CCSDS image library for test runs performed on the SpW-RTC.

Table 5. Comparison between CCSDS 121.0 and 122.0 for lossless compression

	Ratio		Time (seconds)	
	121.0	**122.0**	**121.0**	**122.0**
marstest.raw	1,51	1,67	1,8	11,9
sun_spot.raw	1,76	2,07	2,7	12,9
b1.raw	2,18	2,38	6,5	37,7
solar.raw	1,64	1,93	11,9	51,3

Table 5 shows the compression ratio and time for the CCSDS 121.0 and 122.0 algorithms. Although CCSDS 122.0 can achieve better compression ratios than 121.0, the latter is executed 5 to 7 times faster. It is therefore recommended to perform a trade-off between compression rate and execution speed for each particular application.

6. HYPER/MULTISPECTRAL DATA COMPRESSION

Compression of hyperspectral images is a field of growing importance. New sensors are generating increasing amounts of data, especially in the spectral dimension, as scenes are imaged at a very fine wavelength resolution. This is particularly useful in terms of potential applications, as spectral features allow extracting important information from the data. However, it also makes the size of the acquired datasets very large. Since many sensor platforms, especially spaceborne ones, cannot store all the data but need to transmit them to a ground station, there is a problem of reducing the data volume in order to download all the acquired data.

In the following we describe a compression algorithm for lossless and near-lossless onboard compression of hyperspectral images [14]. In 2007, CCSDS created the Multispectral and Hyperspectral Data Compression (MHDC) Working Group [15] in order to design a new recommended standard for multi- and hyperspectral images. The algorithm described in the following paragraphs is a candidate for standardization.

Beyond good compression performance, onboard compression entails the following requirements:

- **Low encoder complexity**

  Since hyperspectral (and ultraspectral) sensors can generate very high data rates, it is of paramount importance that the encoder has low-complexity, in order to be able to operate in real time.

- **Error-resilience**

  Algorithms should be capable of contain the effect of bit-flippings or packet losses in the compressed file. These errors typically occur because of noise on the communication channel. Traditional compression algorithms can break down completely upon a single bit error in the data, preventing from decoding the remainder of the scene after the error. On the other hand, algorithms such as JPEG 2000 can use sophisticated tools to limit the scope of errors at the codestream level. However, there is a compression penalty to be paid for using these techniques.

- **Hardware friendliness**

  Since onboard compression algorithms for high data rates are typically implemented on FPGA or ASIC platforms, the algorithm design needs to support simple hardware implementation, i.e., it must be able to operate using integer arithmetic, fit into a relatively small FPGA, and use the available resources effectively, possibly avoiding the need for external memory. Moreover, it is desirable that the algorithm can be parallelized in order to speed up the compression process for high data-rate sensors.

Here we describe a compression algorithm that aims at fulfilling the criteria above. The algorithm performs lossless and near-lossless compression with very low complexity. The predictor has low complexity, as it computes a single optimal predictor for each 16x16 block of input samples. Working on 16x16 blocks, the predictor performs data partitioning, in that any 16x16 block can be decoded without reference to any other 16x16 block in different spatial locations in other bands. Thus, while there is a performance loss for working on a block-by-block basis, this is small as opposed to using more sophisticated tools.

The entropy coding stage is based on Golomb and Golomb power-of-two codes, which are known to be a good low-complexity alternative to the more powerful arithmetic codes. Moreover, a quantizer can optionally be inserted in the prediction loop so as to achieve near-lossless compression. The performance of the algorithm, for both lossless and near-lossless compression, can be improved by means of band reordering

Regarding compression performances, results for AVIRIS images are shown in the next Table. We compare the proposed algorithm using Golomb codes and GPO2 codes, but without band reordering, the LUT algorithm [16] and the FL algorithm [17]. As can be seen the proposed algorithm is significantly better than LUT, and almost as good as FL, but with lower complexity. This is a very good result, as FL has a very competitive performance. For comparison, 3D-CALIC using a BSQ format achieves 6.41, 6.23 and 5.62 bpp on sc0, sc3 and sc10 respectively. LAIS-LUT would score an average of 6.50 bpp, which is significantly larger than the proposed algorithm. The use of GPO2 codes does not significantly reduce performance.

Table 6. Performances of multispectral compression algorithms (bits per pixel)

	Proposed (Golomb)	Proposed (GPO2)	LUT	FL
sc0	6.44	6.45	7.14	6.23
sc3	6.29	6.30	6.91	6.10
sc10	5.61	5.62	6.26	5.65
sc11	6.02	6.04	6.69	5.86
sc18	6.38	6.39	7.20	6.32
Average	6.15	6.16	6.84	6.03

A rapid prototyping hardware implementation of the lossless compression algorithm has been performed. The design and modelling phase of the algorithm has been supported by the Matlab/Simulink Xilinx system generator tool, a rapid prototyping environment for the design and implementation of algorithms in FPGAs. The algorithm has been decomposed in elementary functional blocks communicating with ping-pong buffers. Each functional block is in charge of executing the macro computation, and the parallel execution of the functional block in FPGA hardware is exploited for obtaining very high data throughput.

VHDL code has been automatically generated starting from the rapid prototyping tool for two Xilinx FPGA components. The selected FPGAs components are also available in radiation tolerant versions, which is particularly interesting for space applications. Furthermore, a second implementation step has been performed by using a high-level C- to VHDL converter tool applying the same approach used in the modelling phase. Again, VHDL has code has been generated and FPGA algorithm resources have been computed.

Table 7. FPGA resource requirements for proposed multispectral data compression algorithm

Device	Xilinx xqr4vlx200	Xilinx xq2v3000
Used LUT	10306 (5%)	10248 (35%)
Ram 16s	21 of 336 (6%)	21 of 96 (22%)
Mult18x18s		9 of 96 (9%)
DSP48	9 of 96 (9%)	
Max freq (MHz)	81	79
Throughput (Msamples/sec)	70	69

Table 7 summarizes the data of the requested resources for the algorithm implementation in FPGA.

7. PACKET DATA COMPRESSION

Housekeeping or science data from on-board systems or payloads usually consists of packets that contain standardized packet headers and a set of parameters supporting monitoring and diagnosis of the space system's functions. Typically the size and structure of the data packets is fixed, and information is presented in standard data formats such as bytes, 16 bit integers, 32 bit integers or floating point numbers etc. Data files consisting of such packets can be compressed efficiently with standardized compression algorithms in particular if some suitable pre-processing steps are applied. In this chapter, we present some simple types of packet data pre-processing as well as the performance gains obtained on typical test data files.

7.1 Data Pre-processing

The purpose of data preprocessing is the reversible re-arrangement of data for optimizing the performance of the following data compression step. Many data compression algorithms, including the ones presented in this paper, attempt to exploit the similarity in a sequence of numbers to reduce the number of data bits needed for their representation. Any re-arrangement that allows to reversibly place similar numbers next to each other will therefore improve the compressibility of the data series. One example is a series of 16-bit numbers where the sequence of MSB and LSB is different to the representation used by the compression algorithm. If MSB and LSB are swapped, compression will improve dramatically. Another example is data sampled in sequence from different channels of a measurement system. While the data from different channels can be expected to exhibit significant differences, the data from one channel often consists of data elements that show much less deviation. A re-arrangement that groups data elements from individual channels will therefore improve data compressibility. For data sets that consist of an integer number of constant size records this can be done by applying a Transposition, a re-arrangement operation which is equivalent to the well-known matrix operation.

7.2 Transposition of fixed-size / fixed-structure data packages

A *Transposition* can be applied to a set of data consisting of N records of size M:

```
for i=1:N
    for k=1:M
        output((k-1)*N+i)=data((i-1)*M+k);
    end
end
```

It creates a new set of output data that consists of a sequence of all first elements of all original records, then all second elements of the original records etc. A

transposition can be applied on various levels. Typical possibilities include bit-level, byte-level, and 16- or 32 bit word level; in most cases choosing a level that corresponds to the input data representation gives the best results for the subsequent compression step. In this work we have restricted ourselves to transposition on byte and word level which is adequate for subsequent CCSDS RICE compression. A wider range of combinations of transposition type and compression algorithm has been evaluated in [5].

If the data records consist of inhomogeneous data elements (such as a 7-byte header followed by a sequence of 32 bit numbers) *padding* can help to improve data compression performance. In the example, adding 1 padding byte to the header will allow selecting a transposition on byte level, 16-bit or 32-bit word level, instead of byte level only for the unpadded data records, and open up more possibilities for optimizing compression performance.

7.3 Compression Test Results

Data compression tests have been performed on files consisting of fixed size and fixed structure data packets. The chosen compression method was the lossless RICE compression algorithm.

Venus Express (VEX) Orbiter Magnetometer HK data

This instrument generates HK data packets with a size of 98 bytes. Most data elements in the packets are 16bit integer numbers. Compression performance on the unprocessed data was poor (~ factor 1.2) due to the packet content which consists of HK parameters from uncorrelated sources.

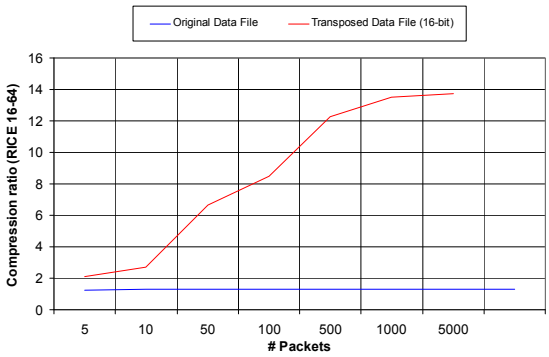

Fig. 2. VEX magnetometer HK data compression ratio

A transposition on 16-bit word level was performed before compressing the resulting dataset. In order to assess the impact of file size on the achievable compression performance, a number of files were generated with different numbers of packets. The compression ratio after transposition improved to a factor of 2 (for a file containing 5 packets) to ~13.9 for a file consisting of 5000 packets. The results are illustrated in Figure 2.

ExoMars WISDOM data

A second set of instrument data used for tests was science and housekeeping data for the ExoMars WISDOM Ground Penetrating Radar [18]. This instrument produces packets of 5151 bytes consisting of 143 bytes of header and housekeeping data plus 1001 32-bit words of science data. Compressibility was evaluated on packet level. The 32-bit parameters represent IEEE-754 floating point numbers containing 1 sign bit, 8 exponent bits, and 23 fraction bits. RICE compression of the original packets leads to a poor compression ratio of only 1.04.

The word size dominating most of the packet structure suggests to apply a transposition on 32-bit word, 16bit word or byte level before applying the RICE compression. Compression tests for all transposition levels show that transposition on byte level is best suited to exploit the redundancy contained in subsequent parameters, and a compression ratio of ~2.78 is achieved on average for a test data set of 18 packets. This result is on the same level with the universal, but much more complex, compression algorithms implemented in the popular ZIP / WinZIP software.

These results show that lossless CCSDS compatible compression can, in combination with simple pre-processing steps such as padding and transposition, significantly improve the compression results achievable for fixed size & fixed structure telemetry data files such as typical spacecraft HK data.

8. THE WHITEDWARF DATA COMPRESSION EVALUATION TOOL

WhiteDwarf is an application that supports the evaluation of compression algorithms by the prospective users of those algorithms. It allows users to compress and decompress their own data files, and optimize algorithm choice and compression parameters by testing with representative user-selected datasets. It also allows users to perform pre-processing functions on their files.

Using this tool, users can test how different combinations of algorithm and compression parameter perform when compressing samples of their own data. These combinations may be stored, exported and imported. The generation of test reports is also supported.

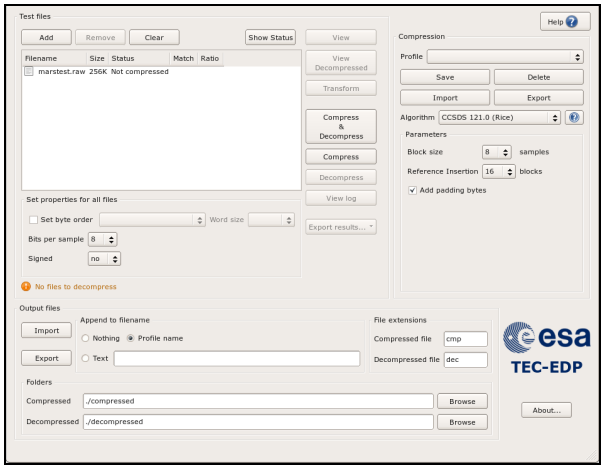

Fig. 2. The WhiteDwarf application's graphical user interface

WhiteDwarf currently supports the CCSDS 121.0, CCSDS 122.0 and DEFLATE algorithms. Additional compression algorithms will be added in the future once the related standardization processes are completed.

9. CONCLUSIONS

Data compression is an important technology with many applications in space projects. ESA is, in cooperation with partner agencies, actively supporting the development and standardization of selected algorithms that are particularly suitable for space applications due to high performance and low resource requirements. Algorithms for lossless generic data compression, lossy and lossless image data compression, and for multispectral data compression are available or under standardization.

Performance tests on selected CCSDS test data have been performed using a typical space qualified 50 MHz LEON2 based processor.

For generic lossless CCSDS 121.0 data compression typical performances of ~6.5 to 10.9 sec/Msample and compression ratios of ~1.5 to 2.2 have been observed.

A new ESA implementation of the CCSDS 122.0 image compression algorithm with significantly reduced resource requirements has been implemented. Typical performances for lossless IDC are ratios of ~1.7 to 2.4 and execution times of ~37.7 to 51.8 sec/Msample. Compression times for lossy IDC (at 0.8 bits per pixel, corresponding to a compression ratio of 10 or 15 for the selected images) ranged from ~27.3 to 30 sec/Msample.

For multispectral image compression, standardization of algorithms is underway, and one of the foreseen algorithms has been presented. High compression performance and low complexity / implementation requirements in both software and hardware are important characteristics of the proposed algorithm.

For the compression of packetized data, in particular for fixed size / fixed format housekeeping data, we have shown that the combination of simple pre-processing steps and lossless CCSDS 121.0 data compression allows a significant reduction of the data volume. Lossless compression factors of ~2 up to ~14 were achieved depending on the data structure and the number of packets in a data file. It is recommended to use similar techniques for platform and instrument HK data compression on missions where limited bandwidth is available.

Finally, ESA has developed a new application software, *WhiteDwarf*, which allows the independent evaluation of ESA-supported data compression algorithms by prospective users. It provides implementations of several algorithms, as well as options and tools for data preprocessing, data manipulation, and reporting. The application is available to users via the On-board Payload Data Processing section (TEC-EDP) at ESA/ESTEC

10. REFERENCES

1. Carlton et al., *A real-time Data Compression system for Apollo TCM Telemetry*, Johns Hopkins University, Technical Memo, December 1969

2. *Mars Express – the Scientific Payload*, ESA-SP 1240, August 2004

3. *Venus Express Payload and Mission*, ESA-SP 1295, November 2007

4. *Huygens Science Payload and Mission*, ESA-SP 1177, August 1997

5. Evans D. et al., *Housekeeping Data: Can you afford not to compress it?*, SpaceOps Conference, 2010

6. CCSDS website at
http://public.ccsds.org/ default.aspx

7. ESA TEC-EDP website at
http://www.esa.int/TEC/OBDP/

8. SpaceWire RTC Device, ESA website at
http://spacewire.esa.int/content/Devices/RTC.php

9. Ilstad J. et al., *SpaceWire Remote Terminal Controller*, DASIA 2008

10. Consultative Committee for Space Data Systems, *CCSDS Test Images*.
http://cwe.ccsds.org/sls/docs/sls-dc/

11. Consultative Committee for Space Data Systems, *Lossless Data Compression*, May 1997. ser. Blue Book, CCSDS 121.0-B-1. http://public.ccsds.org/publications/archive/121x0b1c2.pdf

12. Consultative Committee for Space Data Systems, *Image Data Compression*, Nov. 2005. ser. Blue Book, CCSDS 122.0-B-1. http://public.ccsds.org/publications/archive/122x0b1c2.pdf

13. Consultative Committee for Space Data Systems, *Image Data Compression CCSDS 120.1-G-1*, Jun. 2007, Washington, DC: CCSDS. ser. Green Book. http://public.ccsds.org/publications/archive/120x1g1e1.pdf

14. Abrardo A., Barni M., Bertoli A., Grimaldi R., Magli E., Vitulli R., *Low-complexity algorithm design and hardware implementation for onboard hyperspectral image compression*, Journal of Applied Remote Sensing, 2010.

15. Consultative Committee for Space Data Systems, *Multispectral Hyperspectral Data Compression Working Group*. http://cwe.ccsds.org/sls/docs/Forms/AllItems.aspx

16. Mielikainen J. and Toivanen P., *Clustered DPCM for the lossless compression of hyperspectral images*, IEEE Transactions on Geoscience and Remote Sensing 41, 2943–2946 (2003).

17. Kiely A. and Klimesh M., *Exploiting calibration-induced artifacts in lossless compression of hyperspectral imagery*, IEEE Transactions on Geoscience and Remote Sensing, to appear 2009. http://compression.jpl.nasa.gov/hyperspectral/

18. Trautner R., *Data Compression for ExoMars WISDOM Ground Penetrating Radar Science and Housekeeping Data*, ESA Technical Note TEC-EDP/2009.59/RT, 2010