Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/183-2
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18366
Fax: +43 (1) 58801-18392
E-mail: nabil@ieee.org
florian@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-090

June 28, 2004

On-Board Data Compression of Herschel-PACS Control Data

Ahmed Nabil Belbachir and Florian Schmitzberger

Abstract

The aim of this work is to find a dedicated concept for efficient compression of Herschel-PACS control data. These data represent the setting parameters of the different instruments within PACS, which have to be transmitted with the scientific data for diagnosis purpose. As these control data have a deterministic variability and because of the limited telemetry bandwidth, it was decided to lossless compress the control data for efficient exploitation of the downlink. In this report, an analysis of the control data has been made and a dedicated compression concept with different entropy coders has been developed and tested on simulated and real data. The obtained results have been evaluated and compared to a state-of-the-art codec.

Contents

1 Introduction

The **P**hoto-detector **A**rray **C**amera and **S**pectrometer (PACS) [17] is one of the three instruments housed inside the ESA- **H**erschel **S**pace **O**bservatory (HSO) [16], scheduled for launch on 2007 for exploring the Universe in the far InfraRed (IR) range (55-670μm). PACS will conduct dual band photometry and imaging spectroscopy using respectively the silicon bolometer arrays [12] and the Ge:Ga photo-conductor arrays [19, 25] in the 55-210 micron spectral range.

Due the high data rate (up to 4Mbits/s) generated on board PACS that rapidely overloads the available telemetry bandwidth (120 Kbits), one subsystem inside PACS, namely the **S**ignal **P**rocessing **U**nit (SPU) has been dedicated for on-board compression [3, 6] to fit the telemetry. In other words, an application software running on the SPU is responsible for the data reduction and compression in all dedicated PACS operating and observing modes [17].

PACS consists of two 25x18 photoconductor arrays for spectroscopy, read out at 256 Hz and two bolometer arrays with 32x16 and 64x32 pixels for photometry, read out at a frequency of 40 Hz. These detectors are used for astronomical surveys of actively star-forming galaxies in the young Universe. To achieve this objective, a complex instrumentation is used to support PACS detectors to track the energy released during the formation of stars and galaxies. It includes, an optical system, chopper, grating, filter wheel, cold readout electronic...etc More information can be found in [16].

A subsystem inside PACS, namely **D**etector and **M**echanic **C**ontroller (DMC), has been dedicated to contol and supervise this instrumentation. During Herschel flight, astronomers from the **I**nstrument **C**ontrol **C**enter (ICC) may command the different PACS instrumentation to adapt the planned observation through DMC. For diagnostic, analysis and time-stamping purpose, the control data set by the ICC is transmitted again with the scientific observation (detectors data). This control data is called DMC **H**eader data (DMCH) as it is transmitted as header for the science packets.

The data transmitted from DMC to SPU consist of successive frames (at 40 Hz in photometry and 256 Hz in spectroscopy) that contain control data in the packet header (DMCH) and detector data in the packet body. For an optimal exploitation of the telemetry bandwidth, it was decided to lossless compress the control data. This report describes the DMCH compression concept for PACS instrument. The method is concerned with the recognition and the exploitation of the redundancy in the data for the achievement of the highest possible compression ratio. The used compression method has been tested for simulated and real control data. Comparison of the method with state-of-the art "Zip" algorithm has been performed for evaluation.

This document can be subdivided into four main parts: In Section 2, DMC header is explicitely described. The used compression method is presented in Section 3 in de-

tail. Evaluation of the compression method on simulated and real instrument data and discussion of the results are depicted in Section 4. We conclude with a short summary

2 PACS Control Data Description

PACS control data represent the observation configuration, the detectors setting and the compression parameters. They are set by ground engineers that are responsible for running the planned observation. The control data are transmitted to PACS within the daily telecommunication period, executed by the DMC according to the prescribed commanding sequence, and routed again to ground engineers as header information of the science observation data. We call these data DMC header.
There are two sorts of DMC header: one used for photometry imaging and the other for spectroscopy imaging. In both cases, DMC header consists of 15 parameters/frame for a size of 64bytes/frame. This DMC is generated at 40 Hz in photometry and 256 Hz in spectroscopy. The goal of this work is to compress the control data(DMC header) lossless as much as possible such that the limited-bandwidth can be fully exploited for the science data as much as possible. To achieve this goal, an analysis of the control data has been made in order to adapt a compression scheme that exploit the redundancy.

First, the list of parameters that constitute the control data for photometry and spectroscopy are given in Table 1 and Table 2 below.

Paramater	**Size in Bytes**	**Dynamic Range**	**Variability**
Signal Processing Unit Identifier	4	2-3	0
Type	4	2	0
Observation Identifier	4	0-65535	100
Building Block Identifier	4	0-65535	1000
Label	4	0-255	255
Timing Parameters	8	0- 16777215	10000
Validity	4	0-255	255
Chopper Position Readback	4	0- 16777215	1
Wheel Position Readback	4	0-255	1
BOLC Status	4	0- 16777215	1
On-Board Time clock ticks	4	0- 16777215	1
Current Readout Count	4	0-65535	1
Data Block Identification	4	1-5	1
Bolometer Setup Identifier	4	0-255	255
Compression Mode	4	0-255	255

Table 1: Control Data Parameters for Photometry

The two columns on the left present respectively the parameters that constitute PACS control data in photometry and spectroscopy and their size. The other two columns on the

Paramater	Size in Bytes	Dynamic Range	Variability
Signal Processing Unit Identifier	4	2-3	0
Type	4	1	0
Observation Identifier	4	0-65535	100
Building Block Identifier	4	0-65535	1000
Label	4	0-255	255
Timing Parameters	8	0- 16777215	10000
Validity	4	0-255	255
Chopper Position Readback	4	0- 16777215	1
Wheel Position Readback	4	0-255	1
Grating Position Readback	4	0- 16777215	1
Current Readout Count	4	0-655350	1
Readouts in Ramp Readback	4	0-65535	512
Readout Count since last Set Time	4	0- 16777215	1
CRE Control Readback	4	0-65535	512
Compression Mode	4	0-255	255

Table 2: Control Data Parameters for Spectroscopy

right depict the dynamic range of each parameter and its variability. Variability means the difference between two parameter's value within two successive frames. All the above-listed parameters are unsigned integers with "n" discrete levels.
The aim of this work is to find the adequate method that perform the maximum compression for limited processing power on a 32-bit big endian **D**igital **S**ignal **P**rocessor (DSP). In other words, the objective of this work is to remove all redundancy from the control data and to code the residuals in minimum number of bits. According to the deterministic variability of the individual parameters in the control data, the gradient method followed by character-based encoding technique are used for the compression. This method is described in the following sections.

3 The Proposed Method

Figure 1 depicts the proposed concept for PACS control data compression. It consists of data compression (top in the Figure) and decompression (Bottom of the Figure). The compression phase makes use of the redundancy removal from the data and entropy coding of the residual data while its counter part "the decompression phase" represents the data decoding and reconstruction. The proposed approach consists of two main steps (programs):

- Compression:
  - On-board PACS control data compression using the program "compress.c"

- Decompression:
  - On-ground PACS control data decompression using the program "rledecompress.c OR srsdecompress.c" and "decompress.c"

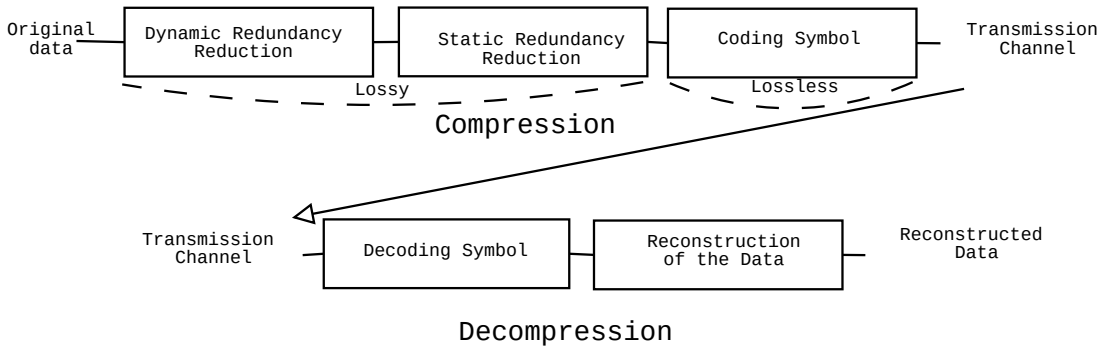

Figure 1: General Block Diagram for PACS Control Data Compression/Decompression

3.1 Raw Data Structure

The raw data consists of a number of words generated at different rates (40 HZ in photometry or 256 Hz in spectroscopy), that results on m frames per seconds.

```
Frame 1   Frame 2   Frame 3           Frame m
 ------    ------    ------            ------
|word 1|  |word 1|  |word 1|          |word 1|
|word 2|  |word 2|  |word 2|  ......  |word 2|
| .... |  | .... |  | .... |          | .... |
|word n|  |word n|  |word n|          |word n|
 ------    ------    ------            ------
```

For testing issue, these frames are stored in a file called "data.dmp". This file consists of a number of frames (nbframes) that contain a certain number of words (nbwords). Compress.c by default compresses the file "data.dmp" for an exposure time of 2s (512 frames) in spectroscopy and 3s (120 frames) in photometry.

3.2 The Compression Approach

The compression approach mainly exploits the redundancy in the **1-D**imensional (1-D) signal resulted from the individual parameters (pointed as word x). After removing this redundancy, an entropy encoder is used to compactly pack the residual data.

The compression algorithm is basically divided into a main-compression and the additional compression functions. All algorithms are implemented in ANSI-C for low algorithmic complexity and implementability in the Analog-Device DSP [1]. The compression algorithm is implemented in modular way (function call) for an easy update and/or exchange of an algorithm, if needed.

3.2.1 Redundancy Reduction

The redundancy reduction step is done in three steps and is implemented in the main()-function:

1. **Preprocessing**
   This step consists of a crop removal of redundant values in 1-D signal. Is the 1-D signal constant respective the time? (word x of frame 1 = word x of frame 2 =...= word x of frame m?)
   If yes, write "1" into position x of the header and just include the value of word x in the compressed file.
   If no, write "0" into position x of the header and include all values of word x of every frame in the file.

2. **Dynamic Redundancy Reduction:**
   This step makes use of the 1st derivative of each 1-D signal to exploit the redundancy in the control data parameters with incremental number. For instance, the 1st derivative of a linear function f(x) is a constant that can be easily exploited for compression.
   $$f(x) = A.x + B \tag{1}$$
   the 1st derivative for f(x) with a slope A and offset B is f'(x) that can be written
   $$f'(x) = \frac{\Delta f}{\Delta x} = A \tag{2}$$
   Starting with the second frame:
   For every word x of a frame y, substract the value of the word x of the frame (y-1) from word x of frame y. Therefore, the remaining constants for linear 1-D functions can be exploited by the lossless coder for high compression.

3. **Static Redundancy Reduction**
   The objective of this step is to decrease the dynamic range of the 1-D signal by substracting an offset signal. In case of a linear function, the offset signal would be the constant resulted from the previous step and the dynamic range will be zero that can be efficiently coded.
   Starting with the third frame:
   For every word x of a frame y, substract the value of the word x of the second frame from word x of frame y.

The purpose of these substractions is to reduce the amplitude of the individual signals (in the best case equal to 0).
When using the additional compression functions, lower values (especially 0's) provides a higher compression rate.

3.2.2 Entropy Encoding

After inducing and reducing the redundancy in the individual 1-D signals form PACS control data parameters, additional compression can be done by coding the resulting residual data using the folowing methods:

- **Simple Repetition suppression** (SRS),
- **Run length encoding** (RLE) OR RZIP

Simple Repetition Suppression

After reducing the dynamic range of the invidual 1-D signals using the redundancy reduction methods, it is expected to obtain multiple of zeros (zero-level signal), that is called tokens sequence. If in a sequence of tokens, a token appears multiple times in a row, replacing all sequential occurences of this token with a flag and the number of occurrences can significantly reduce the data volume and the space used. Of course, this is true with the assumption that the control data almost consist of a combination of multiple 1-D linear functions.
An example:
The sequence 123400000000 could be replaced by 1234f8;
'f' is the zero-flag and '8' is the number of occurrences of '0'. In compress.c SRS is reduced to just Zero-suppression - the main compression function tries to reduce every word to zero.
The zero-flag is the zero-word. (Therefore, a single zero-word takes up two words after SRS-compression because every character after a zero-word has to be the number of occurrences..)

Run length encoding

RLE [8] is a data compression method that physically reduce any type of repeating character sequence, once the sequence of characters reaches a predefined level of occurence. For the special situation where the null character is the repeated character (zero-runs), RLE can be viewed as a superset of SRS.
In case of PACS control data compression, the use of RLE can be efficient if the redundancy reduction step does not reduce the dynamic range to zero. Otherwise, SRS is enough for zero encoding and RLE will only lead to additional resource cost. RLE is used when the

data is expected to have a high number of runs.[1]
RLE works by organizing the data into pairs of symbols, the first representing the data, the second the number of these symbols in the current run.
An example:
The sequence 111122223333 could be reduced to 142434;

In the last few lines of the main-loop of compress.c, please select which additional compression function to use.
SRS creates the file "srscompressed.dat" which can be decompressed with "srsdecompress.c".
RLE creates the file "rlecompressed.dat" which can be decompressed with "rledecompress.c".
In this version of the program and with the expected data, it is highly recommended to use SRS.
In the worst case, RLE can nearly double the file-size.[2]
With the expected data, SRS clearly provides a better compression rate.

The RZIP Algorithm

For efficiency reason and the disadvantages of RLE described above, RZIP algorithm has been developed [4]. *RZip* is a character-oriented compression technique that is developed for DMC header compression. It is not intended for compression of any other data, though it turned out to be useful in other contexts as well. The emphasis was on writing an algorithm that runs *fast* on the DSP and compresses the science frame headers as efficient as possible.

The strategy of *RZip* for searching redundancies in the input buffer is closely related to the data granularity. Our header data words are 32-bit wide. Therefore, a symbol size of 32 bit ensures a good chance in finding reoccurring equal symbols. Another important factor to consider is the wordsize of the CPU or - in case of PACS - the DSP. Most DSP instruction sets only support 32-bit granularity.

So, *RZip* focuses on 32-bit words. Given an arbitrary 32-bit symbol of a data buffer, a logical question can be posed, "Does it reoccur, or not?" If so, "where in the buffer or how often does it repeat?"

Basically, *RZip* takes a symbol and looks ahead for recurrence within a certain index range. The index difference of the two occurrences is encoded taking already coded indices into account. After that, the next occurrence of the symbol is sought if not the end of the buffer is encountered. In case there are no more occurrences, the source buffer is investigated for the next symbol.

The distances can be encoded in different ways. One way is to use the maximum distance as an indicator for no more recurrences. In case of PACS a binary flag after a

[1] Runs are sequential occurrences of a symbol.
[2] If the average of the lengths of the runs is exactly 1.

symbol in the encoded data stream indicates either that an offset will follow or that there are no more occurrences for the current symbol.

Two parameters determine the performance of the algorithm:

- The size of a symbol quantifies the number of bits per symbol. In case of PACS this is fixed at 32 bit per symbol.
- Δ sets the width of the range to look ahead for recurring symbols. For instance, a Δ of 4 means that $2^4 = 16$ indices will be checked.

To get a little more into the algorithm, an explanatory example run is given:

Let `SOURCE` be the data buffer to be compressed. Let `DEST` be the destination buffer where the compressed data will be put.

Δ is the parameter that determines the number of bits to use for encoding ranges. In the following example, 2 is chosen, therefore the effective offset counter δ will be 0..3 ($= 2^2 - 1$). The size of a symbol shall be 32 bit.

the `SOURCE` (symbol buffer) may look like:

```
A A B C A A C C B B  {SOURCE}
```

There is also need for a workbuffer. At the beginning of the algorithm, it has to be cleared.

```
0 0 0 0 0 0 0 0 0 0  {WORK}
```

a) Select the first unused (workbuffer = 0) Symbol and the workbuffer to 1.

```
A A B C A A C C B B  {SOURCE}
1 0 0 0 0 0 0 0 0 0  {WORK}
```

b) Look ahead if the symbol recurs within δ. If yes, code 1 within 1 bit and δ within Δ bits. Set the proper position of the found symbol in the workbuffer to 1. Reset the δ to 0 and continue until no further occurrences are found, then code 0 in 1 bit.

c) Go back to a) until the end of the buffer.

First, all `A`s are coded.

```
A A B C A A C C B B  {SOURCE}
1 1 0 0 1 1 0 0 0 0  {WORK}
A y0 . . y2y0 n  {DEST}
```

The next symbol to code is `B`.

```
A A B C A A C C B B  {SOURCE}
1 1 1 0 1 1 0 0 0 0  {WORK}
Ay0y2y0n B n  {DEST}
```

Next one is `C`.

```
     A A B C A A C C B B  {SOURCE}
     1 1 1 1 1 1 1 1 0 0  {WORK}
Ay0y2y0nBn C . . y0y0 n  {DEST}
```

And finally, `B` again.

```
     A A B C A A C C B B  {SOURCE}
     1 1 1 1 1 1 1 1 1 1  {WORK}
Ay0y2y0nBnCy0y0n B y0 n  {DEST}
```

In this example, 10 symbols of 32 bit size are encoded to 4 symbols plus 10 flags plus 6 ranges $\Delta = 2$ bit. So, the input stream was 320 bit and the output stream is 150 bit. Therefore, the achieved compression ratio in this case is 2.13.

Note that the difference between `A A` will be encoded 0 (0 symbols are between them). The difference between `A X X A` will be encoded 2 if the `X` have not been coded before and if a range of 2 is allowed due to the set Δ (this should be the case). In case you have already coded all `A`s, the difference between the `B`s in `B A C B` will be encoded as 1 (the `A`s are already invisible due to the mask in the workbuffer).

Once a buffer has been compressed with a set of parameters, it can be encoded another time with different parameters. For example, DMC header compression works best with $\Delta = 6$ applied *twice*. Using more than three iterations did not yield any more compression in most cases.

4 Experimental Results

The compression algorithm has been implemented in ANSI-C using the gcc 2.96 compiler and has been tested on a 500 MHz Pentium-equipped machine. Compression method using RLE (Meth-RLE) and RZIP (Meth-RZIP) as backend encoder has been tested on simulated and real DMC header data. No data loss is reported while compression as the compression technique used is lossless.

Compression results using RLE and RZIP on DMC header data from four selected files (Test 1 simulated and Test 2-4 are from real detectors test '6-pack data') are reported in this Section and compared to state-of-the-art Winzip [10] method. The evaluation between these approaches is performed using both criteria: **C**ompression **R**atio (CR) and execution time. Experimental results are presented in Table 3.

We can notice from the obtained results that the generic compression Winzip achieved an acceptable CR (6–7.5) for a slow execution time. The proposed method using RLE has provided the fastest execution time with the lowest compression performance (4.6–6.1). However, RZIP algorithm combinedwith the proposed method provides the best compression performance (10–15.4) within an acceptable and fast execution time. Thus, the know-how of the control data dynamic ranges with an efficient coding algorithm has achieved these promising results.

Methods	Criteria	Test1	Test2	Test3	Test4
Winzip	CR	7.5	6.1	6.2	5.9
	time	10ms	10ms	10ms	10ms
Meth-RLE	CR	6.1	4.6	5.8	5.0
	time	2ms	2ms	2ms	2ms
Meth-RZIP	CR	15.4	10.1	13.5	9.4
	time	4ms	4ms	4ms	4ms

Table 3: Evaluation of the Proposed Method on Simulated and Real DMCH data. Comparaion of both Approaches Meth-RLE and Meth RZIP with Winzip

5 Conclusion

This report presents the description and the test results of the compression technique applied on PACS control data. These latter have been analyzed according to their entropy and variability and a dedicated technique has been developed to best reduce its size. Two entropy coders (Run-length and RZIP) have been evaluated and compared to the state of the art Winzip. RZIP is a dedicated entropy coder that has been created by the authors for an efficient and fast coding of these data.
It was proven that the proposed method, supported by the RZIP algorithm, presents the most adequate method for high compression efficiency and fast execution.
This work can be further extended by investigating more efficient coders like range-coder or Lempel Ziv for higher compression ratio.

References

[1] A.N. Belbachir, H. Bischof and F. Kerschbaum. A Data Compression Concept for Space Applications. *DSP-SPE'00, IEEE Digital Signal Processing Workshop in Hunt, TX, USA, October 2000.*

[2] A.N. Belbachir and H. Bischof. On-Board Data Compression: Noise and Complexity Related Aspects. *Technical Report Number 75, PRIP, TU Vienna, 2003.*

[3] A.N. Belbachir, H. Bischof, R. Ottensamer, H. Feuchtgruber, C. Reimers and F. Kerschbaum. A Compression Concept For Infrared Astronomy: Assessment for the Herschel-PACS Camera. *EURASIP Applied Signal Processing Journal, 2004.*

[4] A.N. Belbachir. Compression Challenges in Infrared Astronomy. *EURASIP Applied Signal Processing Journal, 2004.*

[5] A.N. Belbachir, T. Chilton, M. Dunn, M. Nunkesser, S. Sidhom and G. Szajnowski. Image Compression using Hartley Transform. *Technical Report Number 87 at PRIP, TU Vienna, 2003.*

[6] H. Bischof, A.N. Belbachir, D.C. Hoenigmann, and F. Kerschbaum. A data reduction concept for FIRST/PACS. *In J. B. Breckinridge and P. Jakobsen, editors, UV, Optical, and IR Space Telescopes and Instruments VI. SPIE, Munich, Germany, March 2000.*

[7] I.S. Glass. Handbook of Infrared Astronomy. *Cambridge University Press, Oct.1999.*

[8] G. Held. Data and Image Compression. *Wiley 1996.*

[9] F. Kerschbaum, H. Bischof, A.N. Belbachir, D. C. Hoenigmann, and T. Lebzelter. Evaluation of FIRST/PACS data compression on ISO data. *In J. B. Breckinridge and P. Jakobsen, editors, UV, Optical, and IR Space Telescopes and Instruments VI. SPIE, Munich, Germany, March 2000.*

[10] T. Kohno. Analysis of the Winzip Encryption Method. *IACR ePrint Archive 2004/078.*

[11] F. Kordes, R. Hogendoorn, and J. Marchand. Handbook of Data Compression Algorithms. *Spacecraft Control and Data Systems Division, ESA TM-06, 1990.*

[12] H. Moseley. Large Format Bolometer Arrays for Far Infrared, Submillimeter, and Millimeter Wavelength Astronomy. *Far IR, Sub-mm and mm detector Technology Workshop, CA, USA, April 2002.*

[13] R. Ottensamer, A.N. Belbachir, H. Bischof, H. Feuchtgruber, F. Kerschbaum, C. Reimers and E. Wieprecht. The HERSCHEL-PACS On-Board Software Data Processing Scheme. *Astronomical Data Analysis Software and Systems Conference, Victoria, Canada, September 2001.*

[14] R. Ottensamer, F. Kerschbaum, C. Reimers, A.N. Belbachir and H. Bischof. The Austrian HERSCHEL-PACS On-Board Reduction Work Package. *Hvar Observatory Bulletin, vol. 26, no. 1, p. 77-80, Hungary, 2002.*

[15] R. Ottensamer, A.N. Belbachir, H. Bischof, H. Feuchtgruber, , F. Kerschbaum, A. Poglitsch and C. Reimers. HERSCHEL/PACS On-Board Reduction/Compression Software Implementation. *SPIE International Symposium on Astronomical Telescopes, Glasgow, Scotland, UK, June 2004.*

[16] G.L. Pilbratt. The Herschel mission, scientific objectives and this meeting. *The Promise of the Herschel Space Observatory, ESA-SP, Volume 460, 2001.*

[17] A. Poglitsch, N. Geis, and C. Waelkens. Photoconductor Array Camera and Spectrometer (PACS) for FIRST. *UV, Optical, and IR Space Telescopes and Instruments VI,J.B. Breckinridge and P.Jakobsen, eds., SPIE 2000.*

[18] C. Reimers, A.N. Belbachir, H. Bischof, R. Ottensamer, H. Feuchtgruber, F. Kerschbaum and A. Poglitsch. Feasibility of the On-Board Reduction/Compression Concept for Infrared Camera. *7th International Conference on Pattern Recognition, Cambridge, UK, August 2004.*

[19] D. Rosenthal et al. Stressed Ge:Ga Detector Arrays for PACS And FIFI LS. *Far IR, Sub-mm and mm detector Technology Workshop, CA, USA, April 2002.*

[20] A. Said and W. Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Trans. on Circuits and Systems for Video Technology, Vol.6, pp.243-250, 1996.*

[21] J. Sanchez and M.P. Canton. Space Image Processing. *CRC Press, 1999.*

[22] K. Sayood. Introduction to Data Compression. *Second Edition, Morgan Kaufmann, 2000.*

[23] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Trans. On Signal Processing, Vol.41, pp. 3445-3465, 1993.*

[24] W. Wijmans and P. Armbruster. Data Compression techniques for Space Applications. *DASIA'96, Rome, Italy, May 1996.*

[25] E. Young, G. Rieke, and J. Davis. Development of Advanced Far-Infrared Photoconductor Arrays. *Far IR, Sub-mm and mm detector Technology Workshop, CA, USA, April 2002.*

[26] http://www.mpe.mpg.de/projects.html#first