Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

A FITS Image Compression Proposal

W. Pence

NASA Goddard Space Flight Center

R. L. White, P. Greenfield

Space Telescope Science Institute

D. Tody

IRAF Group, National Optical Astronomy Observatories

Abstract. We have developed a general technique for storing compressed images in FITS binary tables. The image is first divided into one or more rectangular sub-images or tiles, then each tile is compressed and the resulting byte stream is stored in a variable length row of a binary table. By dividing the image into tiles it is possible to extract and uncompress subsections of the image without the expense of uncompressing the whole image. Several commonly used algorithms for compressing the image tiles will be supported initially, and in principle, support for any other compression algorithm may be added later. We are in the process of making trial implementations of this technique within the IRAF image kernel and within the CFITSIO subroutine library for accessing FITS files. Once completed, these implementations will allow application programs to transparently read (and perhaps write) compressed images without needing any knowledge about the compression algorithm.

1. Introduction

With the development of larger and larger imaging detectors there is a growing need for a data format that will allow the images to be stored and directly used in a compressed format. To this end we have developed a general technique for compressing FITS images based on the scheme first proposed by White & Greenfield (1999) that has a number of advantages over simply using `gzip` or UNIX `compress` on the entire FITS file:

- Only the image data are compressed; the FITS header and any other extensions in the file remain uncompressed and hence can be read by FITS file browsers without having to first uncompress the whole file;
- FITS keywords are used to fully and transparently document how the image has been compressed;
- Any number of different compression algorithms can be supported; some algorithms are potentially much more effective for compressing astronomi-

cal images (especially floating point images) than the commonly used `gzip` or UNIX `compress` algorithms;

- Optionally, the image may be divided into a rectangular grid of sub-images (tiles) which are each compressed separately. Reading programs may then randomly access sub-images without having to uncompress the entire image;
- The compressed format is efficient both in terms of disk space and read access time and so is suitable as a run time data analysis format. The additional CPU time needed to uncompress the image is offset by the reduced disk I/O times since significantly fewer bytes need to be read from disk.

In the following sections we describe the current prototype implementation of this compression technique. Some of the details may change as we gain more experience, so readers should consult the latest on-line version of the format description (available from any of the authors) for the precise details of the format.

2. General Description

The general principle used in this convention is to first divide the n-dimensional image into a rectangular grid of sub-images or "tiles". Each tile is then compressed as a continuous block of data, and the resulting compressed byte stream is stored in a row of a variable length column in a FITS binary table. By dividing the image into tiles it is generally possible to extract and uncompress subsections of the image without having to uncompress the whole image. The default tiling pattern treats each row of a 2-dimensional image (or higher dimensional cube) as a tile, such that each tile contains `NAXIS1` pixels. Any other rectangular tiling pattern (including treating the whole image as a single tile) may be defined using the `ZTILEn` keywords that are described below.

3. Keywords

The following keywords are defined to describe the structure of the compressed image:

- `ZIMAGE` (required keyword) This keyword must have the logical value T. It indicates that the FITS binary table extension contains a compressed image, and that logically this extension should be interpreted as an image and not as a table.
- `ZCMPTYPE` (required keyword) The value shall contain a character string giving the name and version of the algorithm that must be used to decompress the image. Currently, values of `GZIP_1`, `RICE_1`, `PLIO_1`, and `HCOMPRESS_1` are reserved to refer to several commonly used algorithms (`PLIO` stands for the IRAF Pixel List compression algorithm). We intend to provide a detailed description of how to uncompress each of these formats in the final version of the document.
- `ZBITPIX` (required keyword) The value shall contain an integer that gives the value of the `BITPIX` keyword in the uncompressed FITS image.

- `ZNAXIS` (required keyword) The value shall contain an integer that gives the value of the `NAXIS` keyword in the uncompressed FITS image.
- `ZNAXISn` (required keywords) The value shall contain a positive integer that gives the value of the `NAXISn` keyword in the uncompressed FITS image.
- `ZTILEn` (optional keywords) The value of these indexed keywords (where `n` ranges from 1 to `ZNAXIS`) shall contain a positive integer representing the number of pixels along axis `n` of the compression tiles. All the pixels within each tile are compressed as a contiguous data array and stored in a row of a variable-length vector column in the binary table. The size of each image dimension (given by `ZNAXISn`) is not required to be an integer multiple of ZTILEn, and if it is not, then the last tile along that dimension of the image will contain fewer image pixels than the other tiles. If the `ZTILEn` keywords are not present then the default 'row by row' tiling will be assumed such that `ZTILE1 = ZNAXIS1`, and the value of all the other `ZTILEn` keywords equals 1. The compressed image tiles are stored in the binary table in the same order that the first pixel in each tile appears in the FITS image.
- `ZNAMEn and ZVALn` (optional keywords) These pairs of optional array keywords (where n is an integer index number starting with 1) supply the name and value, respectively, of any algorithm-specific parameters that are needed to compress or uncompress the image. The value of `ZVALn` may have any valid FITS data type. The order of the compression parameters may be significant, and may be defined as part of the description of the specific decompression algorithm.
- `Other Keywords` The binary table header may contain any additional keywords to provide information about the image. In general, all the keywords in the header of the FITS image will be copied verbatim into the header of the compressed binary table and these keywords will have the same meaning in the binary table as they did in the image. The mandatory `BITPIX`, `NAXIS`, and `NAXISn` keywords are not copied and are instead replaced by the `ZBITPIX`, `ZNAXIS`, and `ZNAXISn` keywords as described above.

4. Columns

The following columns in the FITS binary table are defined by this convention. The order of the columns in the table is not significant. The column names (given by the `TTYPEn` keyword) are shown here in upper case letters, but the case is not significant. Any number of other columns besides those defined here may be present in the table to supply other parameters that relate to each image tile.

- `COMPRESSED_DATA` (required column) Each row of this variable-length column contains the byte stream that was generated as a result of compressing the corresponding image tile. The data type of the column (as given by the `TFORMn` keyword) will generally be either `'1PB'`, `'1PI'`, or `'1PJ'`, depending on whether the compression algorithm generates an output stream of 8-bit bytes, 16-bit integers, or 32-bit integers, respectively. If it is not possible to efficiently compress a particular image tile, then the `COMPRESSED_DATA` vector in the corresponding row will have a length of zero, and the uncompressed tile pixels will be written instead to the `UNCOMPRESSED_DATA` column described below.

- `UNCOMPRESSED_DATA` (optional column) This variable length column will contain the uncompressed pixels for any tiles that cannot be compressed. The data type of this column will usually correspond to the data type of the original image. If all the tiles in an image are compressed, then the `UNCOMPRESSED_DATA` column is not required.
- `ZSCALE and ZZERO` (optional columns) These columns give the linear scale factor and zero point offset which may be needed to transform the raw uncompressed values back to the original image pixel values (or at least a close approximation to the original values) using the following formula:

  image_pixel_value = uncompressed_value * `ZSCALE + ZZERO`

  `ZSCALE` and `ZZERO` generally have double precision values and have default values of 1.0 and 0.0, respectively. If the same values of ZSCALE and ZZERO apply to every tile in the image, then they may be given as header keywords rather than as table columns. `ZSCALE` and `ZZERO` are typically used to scale floating point images (with `BITPIX` = -32 or -64) into integers before compression, since most compression algorithms are not very efficient with floating point data.
- `ZBLANK` (optional column) In cases where floating point images are converted to integers before being compressed, this column gives the the integer value that is used to represent undefined pixels (if any) in the image. These pixels would have an IEEE NaN (Not a Number) value in the uncompressed floating point FITS image. If every tile uses the same null value, then `ZBLANK` may be given as a keyword instead of as a table column. If there are no undefined pixels in the image then `ZBLANK` is not required.

5. Trial Implementations

We plan to release the prototype implementation of this compression scheme within the IRAF kernel and the CFITSIO libraries for trial use by other software developers and data providers. Eventually it is hoped that this convention will be widely supported and perhaps adopted as part of the official FITS Standard.

References

White, L. R., & Greenfield, P. 1999, in ASP Conf. Ser., Vol. 172, Astronomical Data Analysis Software and Systems VIII, ed. David M. Mehringer, Raymond L. Plante, & Douglas A. Roberts (San Francisco: ASP), 125