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

Implementation of the Advanced Camera for Surveys Calibration Pipeline

W. J. Hack, P. Greenfield

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21218

Abstract. The calibration pipeline for the Hubble Space Telescope's (HST's) Advanced Camera for Surveys (ACS), CALACS, was developed to efficiently process the large datasets generated by ACS. This pipeline evolved from the code for CALSTIS, the calibration pipeline for HST's Space Telescopes Imaging Spectrograph (STIS), since both instruments generate imaging data from similar detectors (CCDs and MAMAs). However, ACS can create images with up to 4096×4096 pixels, and like NICMOS (HST's Near-Infrared Camera), these images can be associated with other ACS images within the pipeline for cosmic-ray rejection and other processing. CALACS therefore used line-by-line I/O and new library routines to process the images with reduced memory usage, and expanded the concept of how associations are processed. The similar structure between ACS data, and STIS data, and even NICMOS data, therefore, allowed for the significant re-use of code from established pipelines to greatly reduce the effort in development and maintenance of the new software package.

1. Problems

ACS will produce a large amount of imaging data, especially from its Wide-Field Camera (WFC), and this data will need to be processed at STScI prior to being archived and shipped to the observer. This pipeline processing requires that the data be processed quickly, not only placing restrictions on processing time for ACS data, but it also can't delay the processing of other observations. While the observer may be satisfied to wait a little longer to re-process ACS data on their own machine, there are no guarantees that they will have large amounts of RAM to process the largest images in memory. Therefore, two primary issues need to be addressed by the ACS pipeline:

1. Run-times can not be excessive, even for the largest datasets,
2. Memory usage must be reasonable.

ACS data will be formatted so that each file will correspond to a single exposure and these exposures can be associated together. As a result, two unique complications arise for CALACS:

1. A single association can produce multiple output products,
2. Processing comments for all inputs must be logged with multiple output products.

The pipeline must take a set of exposures and combine them in a multitude of ways; including, cosmic-ray rejection and combining of CR-split images, summing repeat-obs exposures, and combining cosmic-ray rejected images from different dither positions into a dither-combined image. The pipeline will need to rely on the association table, originally developed for STIS and NICMOS, to keep track of what input exposures are part of an association and what products should be produced. Amidst the processing of multiple products, the pipeline will still need to keep track of what processing was performed on each input, but something that now needs to be spread out amongst all the individual files being processed.

2. Overall Design

ACS uses detectors designed for STIS, therefore it was natural to design CALACS on code used for processing STIS data with modifications to account for the larger ACS images and multiple output products. The pipeline is comprised of one primary task and 4 calibration tasks which can either be run individually or automatically from the primary task. Each task was named so that the observer would more easily recognize each tasks functionality.

CALACS This primary task interprets the association table or individual image header using CALNICB-based (NICMOS calibration pipeline) code and controls the operation of the proper tasks during processing.

ACSCCD It performs the initial processing steps required for all CCD data.

ACS2D It contains all the basic calibration functions applied to both MAMA and CCD data.

ACSREJ It performs the cosmic-ray rejection and creates the combined cleaned image for CCD CR-SPLIT data.

ACSSUM It combines REPEAT-OBS exposures into a single output image.

For ACS WFC data, a single IMSET, defined as the grouping of the science, error and data quality arrays requires 80MB of memory and combining 4 CR-SPLIT Insets would require 400MB of memory if the CALSTIS memory model was used. Memory usage was constrained by keeping only the output product in memory and by performing image operations involving a second image using line-by-line I/O.

Association tables specify all the input data sets that make up the association and the products to be produced by the pipeline calibration software. The products can represent the combination of all the input data sets or the combination of subsets of the input data sets. An example would be an observation taken using a dither pattern, where each dither pointing was CR-SPLIT for cosmic-ray removal. In this example, each set of CR-split exposures taken at each dithered position on the sky would be combined into a cosmic-ray rejected output image. A final association product will be produced as the combination of the CR-combined images from each dither position. Thus, this one association can result in the creation of CR-combined images (one for each dither pointing) and a dither-combined product of the whole set (though the latter has not yet been specified or implemented in CALACS).

With past HST instruments, each product produced from calibration pipeline processing has always been the only product of a calibration task invocation. It was simple to record the processing for each product by logging the executable's processing messages to a file associated with the output product by means of simple I/O redirection. CALACS is unique in that a single task invocation may produce many products. As a result, it was necessary for CALACS to explicitly log the processing messages for all the products.

3. Implementation

CALACS started with code from the CALSTIS tasks **cs0**, **cs1**, **cs2**, and **cs8**, tasks primarily responsible for STIS image reduction, while code from CALNICB provided the basis for the functions which interpret the association tables.

- **cs0** was used as the basis for the **CALACS** task to control the operation of the entire pipeline. The CALNICB code was inserted here to interpret the expanded association files.
- The CALSTIS functions were re-organized to more efficiently work with ACS data. CCD-only operations were pulled from **cs1** and turned into **ACSCCD**. Remaining functions were used to create **ACS2D**. Initialization functions for DQ and error arrays were duplicated in **ACS2D** so ACS2D could work with MAMA data as well.
- The cosmic-ray rejection task **ACSREJ** was derived from **cs2** with the revisions necessary for line-by-line I/O.
- Summing of calibrated repeat-obs exposures was handled by **cs8** in CALSTIS and this was copied and revised into the task **ACSSUM**.
- The pipeline C code used the CVOS bindings developed at STScI (see the poster paper by M. De La Peña 2000) and the HSTIO library to access the science and reference data.
- New functions were written to manage the trailer files along with trailer file buffers used to insure that all comments are recorded even when a trailer file hasn't been defined yet. These functions included special output functions used throughout all the code to direct standard output to STDIO as well as the trailer file buffer.
- New HSTIO functions were created for line-by-line access to IMSETs.

The majority of the revisions necessary to convert the CALSTIS code into code usable on ACS data involved the line-by-line I/O. This involved adding another layer of loops over the lines in the image to most of the functions. Revisions to **ACSREJ** were more extensive owing to the fact that several scrolling sections had to be maintained to support the cosmic-ray rejection algorithm. Other revisions included removing all code dealing with spectroscopic modes and binned MAMA data, since the ACS MAMA only has one resolution as opposed to the STIS detector. A more detailed description of the CALACS code and its implementation can be found in the Instrument Science Report "CALACS Operation and Implementation" found on the Advanced Camera for

Surveys Documents Advanced Camera for Surveys Documents WWW page at the URL `http://www.stsci.edu/instruments/acs`.

4. Summary

Work on coding CALACS began in March 1998, after the critical design review was completed, and the first version of the entire pipeline passed its initial integrated pipeline testing at the end of May 1999. This first version included all the basic functionality necessary for processing the most common ACS data, while processing special data remains to be tested due either to lack of test data (for sub-array data) or lack of a calibration procedure (for ramp-filter or dithered observations). Even so, the entire pipeline for applying basic calibrations was generated from STIS code in a little over a year, rather than taking several years like previous projects. The use of C for the previous pipelines made the conversions and integration of code easy to implement due to the tools available for developing C code.

The use of line-by-line I/O did not significantly affect the run-time efficiency of the ACS pipeline. Although it relies more heavily on the I/O sub-system than previous pipelines, current systems can still process ACS data with CPU usage over 70%, rather than having a lower CPU usage as the system waits for the I/O to complete. In the end, an Ultra-class Sun workstation can process a single WFC image in under 5 minutes, as opposed to 90 seconds for a Sparc-4 to process a STIS image (which is 1/16 the size). In addition, ACS associations can be processed with similar efficiency depending on the number of input exposures in the association and the processor speed.

Finally, memory usage peaks out at about 130MB while performing cosmic-ray rejection on associated ACS WFC images. This allows the pipeline to be run on workstations with 256MB of memory, an amount which can be fairly readily accommodated on most systems. In the end, the development of CALACS has demonstrated that re-use of code and development in C can result in the rapid generation of a new task.

References

De La Peña, M. D. & Farris, A. 2000, this volume, 677