

Initial Implementation Strategy for Drizzle with ACS

W.B. Sparks, W. Hack, R.N. Hook
April 25, 2001

ABSTRACT

In order to provide geometric correction for single pointing ACS images, and to provide geometric correction together with simple image combination for associations of ACS images, we describe plans to implement the “drizzle” code by means of a python wrapper, and to use this wrapper in calacs. The initial strategy will endeavour to be robust and scientifically accurate, although not necessarily optimal. An upgrade path is outlined which could lead to significantly improved processing, involving an iterative pass through the data. The tools will be available stand-alone, offering a greater degree of flexibility than in pipeline implementation. The output product will be a multiple extension fits file containing the data (units counts per second), a weight image and a context image. The latter are provided by the drizzle program and are related to the variance and data quality arrays respectively.

Background

Development work on implementation of geometric distortion correction and combination of associated images for ACS data is underway. The utility to be used is the *iraf* task *drizzle* developed by Richard Hook at ST-ECF and Andy Fruchter at STScI. It is expected to be available as stand-alone software, and if time permits, to be incorporated in an experimental way in the ACS pipeline *calacs*. To accomplish the latter, a number of primarily scientific decisions need to be made. This document describes and defines the initial choices and some elements of the design.

ACS images have large geometric distortions. Additionally, the use of “dither” strategies when observing has become popular in order to remove the influence of small scale detec-

tor defects and to recover some information lost by undersampling by pixels large compared to the PSF, see Koekemoer et al. “*HST Dither Handbook*”. *Drizzle* deals with both of these considerations in a single software program. Dither strategies for ACS specifically are discussed in detail by Stiavelli et al. in ISR/ACS 98-02.

Accordingly, the decision was made to proceed with development of an interface to the *drizzle* program which would be suitable for ACS and other HST instrument data, with a view towards incorporating it in the ACS pipeline if feasible. From the outset placeholders have been provided in the ACS pipeline and data products in anticipation of this effort. The optical characteristics of the WFC3 camera are similar, and it is expected that this work will carry over directly to WFC3 and other instruments. The version of *drizzle* used within the python wrapper, and hence in the ACS pipeline, will be the one available for use within *stsdas* by the user directly.

Here we describe an implementation strategy that we expect to lead to the inclusion of *drizzle* in *calacs*, highlight decisions that need to be made, and identify a possible upgrade path. Our philosophy is to provide (i) the simplest, robust implementation which (ii) does not compromise scientific integrity of the data, and (iii) which should be useful for a moderately high fraction of observers. We do NOT attempt to provide the ultimate, optimal image combination in this implementation. Nevertheless, the framework will be present to adjust parameters as experience is gained. In stand alone mode, the python wrapper should enable the full capabilities of the single *stsdas drizzle* task (NB: not the entire contents of the *stsdas.dither* package).

Description of the Computational Approach and Data Flow

Warren Hack has written a python wrapper, “*pydrizzle*”, which provides an interface to the core software *drizzle* task maintained by Richard Hook at ST-ECF and implemented within the *stsdas dither* package. Richard Hook has been serving as a consultant in that effort. The python wrapper underwent a code design review by SSG and with some modifications was approved for development of the code. Documentation is in preparation, Hack 2001 ACS ISR, to describe how the wrapper was implemented and how to use it. The *calacs* design may be found in various Instrument Science Reports including “*calacs* operation and implementation” ISR 99-03 (Hack), available through the ISRs link under the WWW page <http://www.stsci.edu/cgi-bin/acs>. See also Figure 1 which is extracted from Figure 11.1 of the ACS Instrument Handbook.

- *For a single pointing, the wrapper will provide the parameter set necessary for drizzle to carry out a geometric distortion correction.*
- *For an associated set of dithered images, the wrapper will provide the parameters needed to run drizzle on each input image and will allow drizzle to combine the data, or optionally combine the output drizzled images externally to the drizzle program.*

The archive catalog is currently built from the two-extension data products (in the WFC case) that serve as input to the image combination module. Given the somewhat experi-

mental nature of the image combination implementation in the pipeline, we do not propose to change this, nor indeed to eliminate any of the current *calacs* pipeline capabilities and deliverables. We will however populate placeholders that are currently empty and we will combine the two extensions of the WFC channel of ACS into a single image. In the *stsdas dither* package, normal usage requires iteration and multiple drizzles in order to identify cosmic rays and defects on the original input image, rather than on the output drizzled image where they are smeared out. Such iterations may also be used to adjust pointing information based on science data, and to adjust estimation of background sky levels.

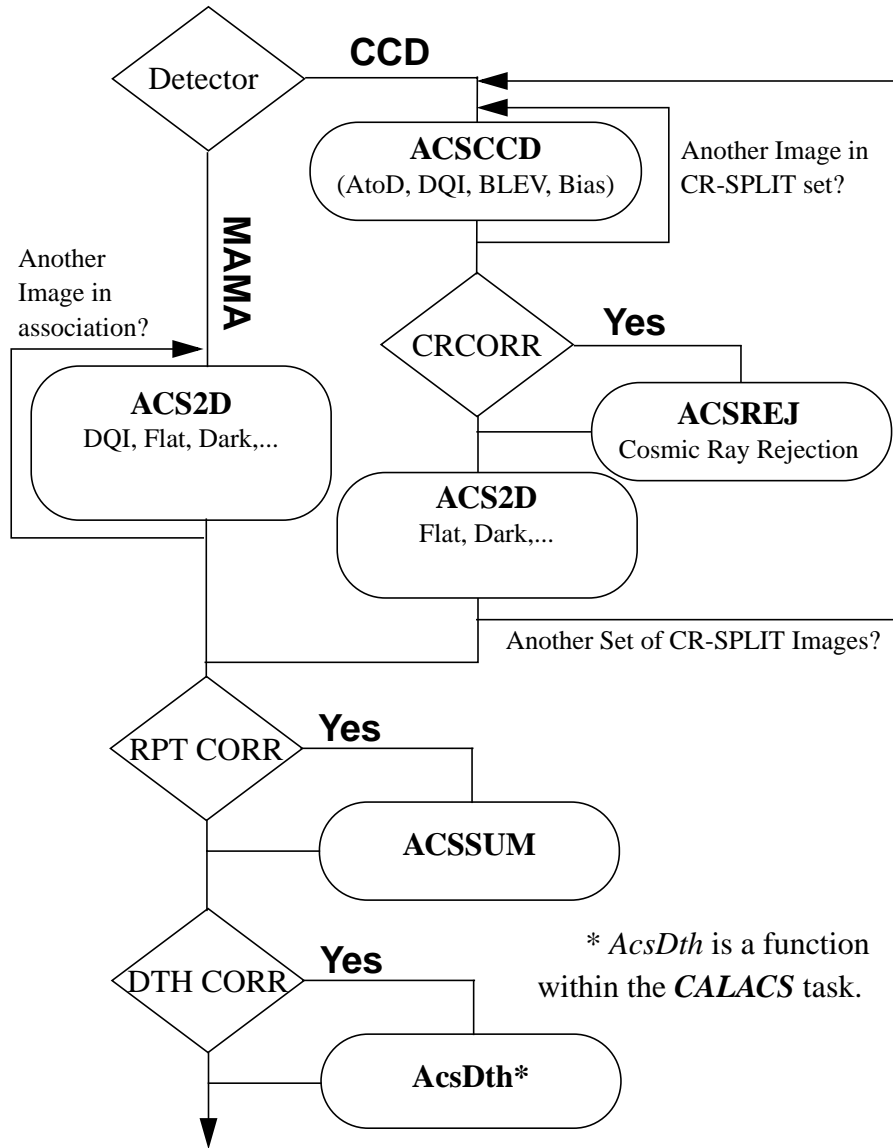
- In an initial *calacs* implementation we do NOT propose to invoke any iterative steps.
- Pointing offsets as commanded to the telescope will be used. We do not propose to use jitter information in the initial *calacs* implementation.

In non-pipeline usage, users will be able to iterate and adjust parameters as they wish. Once the package is fully developed for off-line use, documentation on its use will be made available.

Substantial observing overheads may lead some observers to carry out a dither pattern without cosmic ray rejection at individual pointings. To carry out cosmic ray rejection post-drizzle therefore represents a simple, albeit not ideal, compromise that may improve the quality of the data compared to doing nothing at all.

- We are investigating the utility of inserting an extra cosmic ray rejection stage in *calacs*. If the advantages seem significant, we will offer this option. This would be an upgrade over the initial implementation however.

Figure 1: Flow Diagram for ACS data shown with *calacs* task names.



Output Products and Formats I

The following lists define standard ACS filenames as they are presently implemented.

Normal unassociated exposure

Calibrated	_trl _spt _flt
Uncalibrated	_trl _spt _raw

Association

Calibrated	_trl _spt _asn _crj/_sfl _dth(_dth for the products, _crj/_sfl for the subproducts, _spt/_trl for the Sub-products and exposures)
Uncalibrated	_trl _spt _asn _raw (for all the exposures that make it up)

Sub-products of an Association

Calibrated	_trl	_spt	_asn	_crj/_sfl
Uncalibrated	_trl	_spt	_asn	_raw (for each exposure that makes it up)

Exposure of an Association

Calibrated	_trl	_spt	_asn	_flt
Uncalibrated	_trl	_spt	_asn	_raw

For any association, you will have either `_crj` or `_sfl` subproducts, not both. *calacs* will *not* delete any of the calibrated subproducts during processing, so they will be present after the final product has been generated.

- We propose that these conventions be slightly modified. We require an additional file for the geometrically corrected image in the case of a single pointing. For consistency, rather than using `*_dth` therefore we propose changing the name to `*_drz` and using that product as the repository for the final single image in *both* the dithered and single pointing cases.

Output Products and Formats II

Following the application of *drizzle* to an associated set of images, the following products are created:

- a single scientific data image
- a weight image
- a context image

Consistent with the design strategy of the ACS pipeline, and other calibration pipelines, we will provide these as three image extensions within a single output file (currently the `*_dth` file). The FITS extension system has been in place for a number of years now, beginning with STIS and NICMOS for HST, and is mature.

- Given the exposure time differences in different regions of a dither-combined output image, no single exposure time characterizes an image. The choice of output data units becomes an issue. Counts per second is the current *drizzle* default. A polling of representative members of the community, including STUC, other instrument groups at STScI and the IDT, indicated this is acceptable for ACS data.
- We propose to use exposure time weighting, uniform for each image; see Casertano et al. (2000) AJ, 120, 2747 for a detailed discussion of weighting. This provides unbiased estimation of the local count rate.

The use of exposure time weighting is appropriate for background dominated observations. Exposure time squared is available as an option within *drizzle*, as is provision of a general weighting image. Simple “exposure time” weighting is likely to be valid for most ACS observations, and uniform weighting per image is desirable, as was used with the HDF. (Alternatives would be to use a $1/\sigma^2$ scheme, or to use the local S/N, as obtained from “error images”, however these both bias the count rate estimation by an uncertain amount.) It will be relatively straightforward to change the default weighting scheme should subsequent experience indicate that to be desirable.

calacs processing flags bad pixels and cosmic-rays using data quality images. Pixels with bad-data flags set will be ignored (i.e. not used) in the *drizzle* combination. However, in the initial implementation, cosmic rays identified by *acsrej* in the pipeline are replaced with good values (from frames unaffected by cosmic rays) and so these will be accepted in the *drizzle* combination.

- The context image is a bitmap which offers a key to the input images that were used at each pixel in the output image. A series of zeros or ones together with a lookup table identifies the input images, and parameters associated with those input images, such as exposure time, sky level, pointing offset and so on. Following NICMOS, we propose to utilize the existing association table as the lookup table. We may write back parameters into that, or adjust them for running offline. Note that the input association table is the natural information source for the python wrapper to interface with *drizzle* and to provide the input source list of images and their characteristics. In stand-alone use, a fits table with similar entries to the association table provides the image information for combination.

Simple tools will allow generation of, for example, an exposure time image from the context image since in a dither combination different regions of the output have very different exposure times, given a typical set of overlapping images. (A caveat is that under the present configuration information on exposure time per pixel is lost where cosmic rays have been replaced with scaled good values. A future upgrade should allow recovery of that information.)

Default Parameter Choice for CALACS

The wrapper utilizes the newly populated IDCTAB (see ACS/ISR 00-11 by Hack and Cox) that encapsulates our best knowledge of the geometric distortions as a set of polynomial coefficients in a single file. The data are consistent with the Science Instrument Aperture Files. The file design is general and applicable to other HST instruments, and was developed in collaboration with programmers assigned to other instruments. By using this file in conjunction with the world coordinate system in the image header we obtain a sensible positioning of the input images on the output image.

The parameter list below for the *drizzle* task contains the values used for processing an ACS HRC image. A few parameter values are calculated directly by the wrapper; in particular,

- *outnx, outny*: size of the final output image based on corrected positions of the corners of all the inputs.
- *coeffs*: file containing the *drizzle* coefficients; for ACS, it will be a temporary file containing the converted IDCTAB coefficients.
- *xsh, ysh, rot*: offset in X and Y and rotation to be applied to each input image. These will be set to zero as the distortion coefficients will account for these for each chip in a single observation. Multiple observations will cause these values to be adjusted for each position in the dither pattern.

- *shft_un, shft_fr*: units and frame which the shift values will be in; set to ‘output’ pixels since we have calculated everything based on output frame.
- *scale*: Ratio of output pixel scale to input pixel scale; default based on IDCTAB (pipeline) pixel scale or it can be specified by the user for sub-sampling.
- *out_un*: Output units will default to ‘cps’ (counts per sec), but may be overridden by the user.
- *kernel*: kernel shape used for drizzling pixels into output frame; default of ‘square’ corresponding to traditional drizzling will be used.

The complete parameter list for the *drizzle* task is:

```

data      = j61a01031_crj.fits[sci,2]  Input data image
outdata   = wfc_drizout.fits           Output data image
(outweig  = wfc_drizout_weight.fits)  Output weighting image
(outcont  = wfc_drizout_context.fits)  Output context image
(in_mask  = )                          Input weighting mask
(wt_scl   = exptime)                    Weighting factor for input data image
(outnx    = 4277)                        X dimension for created images (if outdata not extant)
(outny    = 4302)                        Y dimension for created images (if outdata not extant)

                                GEOMETRIC PARAMETERS
(kernel   = square)                  Shape of kernel function
(pixfrac  = 1.0)                      Linear size of drop in input pixels
(scale    = 0.98)                      Linear size of output pixels in terms of input pixels
(coeffs   = j61a01031_crj_coeffs1.dat) Geometrical distortion coefficients file
name
(lambda   = 555.0)                    Effective Wavelength (nm), Trauger coefficients only
(xsh      = 0.0)                       X shift to be applied to input image
(ysh      = 0.0)                       Y shift to be applied to input image
(rot      = 0.0)                       Rotation of input image to be applied
                                (degrees anti-clockwise)
(shft_un  = output)                   Units of shifts (input or output pixels)
(shft_fr  = output)                   Frame in which shifts are applied
(align    = center)                   Reference point: corner or center of pixel

                                DATA VALUE SCALING PARAMETERS
(expkey   = exptime)                  Exposure time keyword in input data image header
(in_un    = counts)                   Units of input image (counts or cps)
(out_un   = cps)                       Units for output image (counts or cps)
(fillval  = INDEF)                     Value to be assigned to undefined output points
(mode     = h)

```

- The default value for *scale* is based on the pixel scale provided in the IDC table. The software will be written to function with arbitrary values, while suitable defaults will be selected by the ACS group at STScI. We anticipate using *scale* of nearly 1.0 and an output pixel size of 0.025, 0.05 arcsec/pix for the HRC and WFC respectively, in the initial delivery. *As experience is gained, it is very straightforward to change these values, should it prove desirable.*
- The default value of *pixfrac* will be 1.0 in the initial pipeline implementation, although this may easily be overridden in offline use. This does not offer enhanced spatial resolution, but is the most robust choice. Interpixel dependencies lead to spatially dependent noise, e.g. as seen in geometrically corrected FOC images, but a non-unity *pixfrac* can also lead to spatially dependent background ripples if background offsets are not corrected.

An optimal selection of *pixfrac* will depend on the number of images and the observing strategy employed (subpixel, integer pixel etc.). We do not feel prepared to make decisions

changing `pixfrac` from 1.0 (in the pipeline) without experience of actual ACS science observations.

Potential Upgrade Path

The first phase of upgrade may be adjustment of existing parameters, such as the scaling parameters described above. Also, modifications to the adopted weighting scheme may prove desirable, utilizing for example a noise model based on mean background level within the image. The “`exptime`” weighting assumes background dominated, yet there will certainly be observations that are read noise limited. A simple noise model may account for the transition and include other noise sources.

The following additional steps may be used in processing ACS data offline, and in principle therefore also offer a potential route for upgrading the ACS pipeline. They represent capabilities available in the *stsdas dither* package.

1. *Blot* the initially *drizzled* image back onto the input images, re-evaluate for cosmic rays and adjust the dq masks for the input images. May loop back to “`_flt`” or “`_crj`” (or even all the way to “`_raw`” files). The `EXPSCORRflag` allows generation of processed individual images (“`_flt`”) even where there are “`CR-SPLIT`” associations.
2. Examine jitter files for losses of lock or other pointing anomalies
3. Apply an image registration algorithm such as cross correlation to tweak the image offsets; possibly including offsets between individual `_raw` files.
4. Adjust image background levels in order to eliminate edges from mosaics.
5. Determine an optimal `pixfrac` and scale according to the observing strategy.

The association table defines the primary inputs to the wrapper and may be used to record adjusted parameters associated with individual input images. It may in principle also be used to summarize other relevant information such as the wavelength, photometric calibration, sky levels.

Formally, the association table is currently generated at the time of proposal preparation. An alternative more general approach is to conceive a new type of association, one generated on-the-fly or in the database but derived from available archival data rather than from exposure logsheet lines. In that way, any data (different filters, different pointings, different visits, different orientations...) can be “associated” similar to the ECF initiative on WFPC2 associations. A description of how more general association tables can be built for use by the python wrapper will be included in the documentation when it becomes available. With a general tabulation of images and their characteristics, it should be straightforward to run the python wrapper for *drizzle*. We hope this will provide a versatile offline capability for combining image data, and will require no fundamentally new capabilities. Additional utilities to enhance the data quality array information provided as input

may be useful, as this is the primary means by which the *pydrizzle* implementation rejects bad data.

Schedule

The approximate schedule follows. A certain amount of flexibility is required to avoid interference with potentially critical phases such as SMOV.

Overall design outline: complete mid November timed to be available during Richard Hook's visit Nov 16/17. **Completed.**

Coding phase (python wrapper; Hook's core *drizzle* routine) stand alone utilities: through February 2001. **Completed**

Test phase; eliminate bugs, initial choice of parameters: February and March 2001. **Current**

Release pyraf stand-alone version, begin pipeline testing. Beta release, Mid-May 2001

Second test phase, May to September 2001

Incorporation into calacs and OPUS: ~September 2001. Actual implementation will depend on launch date.