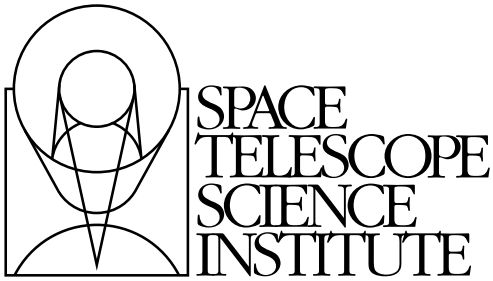
Version 1.0
June 2012

The DrizzlePac Handbook



Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
help@stsci.edu

Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration

User Support

For prompt answers to any question, please contact the STScI Help Desk.

- **E-mail:** help@stsci.edu
- **Phone:** (410) 338-1082
  (800) 544-8125 (U.S., toll free);
  from outside the United States and Canada, call [1] 410-338-1082.

World Wide Web

Information and other resources are available at the DrizzlePac Website:

- URL: http://drizzlepac.stsci.edu

DrizzlePac and MultiDrizzle Handbook History

Document	Version	Date	Editors
The DrizzlePac Handbook	1.0	June 2012	Gonzaga, S., Hack W., Fruchter, A., and Mack, J. et al. Susan Rose-Technical Editor
The AstroDrizzle Mini-handbook	1.0	February 2012	Gonzaga, S., Hack W., Fruchter, A., Lindsay, K., Dencheva, N., Sosey, M. Susan Rose-Technical Editor
The MultiDrizzle Handbook	1.0	November 2008	Fruchter, A., Sosey, M., Hack, W., Dressel, L., Koekemoer, A. M., Mack, J., Mutchler, M. and Pirzkal, N. Susan Rose-Technical Editor
HST Dither Handbook	2.0	January 2002	Koekemoer, A. M., Gonzaga, S., Fruchter, A., Biretta, J., Casertano, S., Hsu, J.-C., Lallo, M., Mutchler, M. and Hook, W. Susan Rose-Technical Editor
HST Dither Handbook	1.0	December 2000	Koekemoer, Anton M., et al.

Contributors

Drizzle documentation is the result of combined work by many individuals over the years. For this latest handbook version, primary contributors (in alphabetical order) are Amber Armstrong, Roberto Avila, Nadia Dencheva, Michael Dulude, Andy Fruchter, Shireen Gonzaga, Warren Hack, Kevin Lindsay, Ray Lucas, Jennifer Mack, John Mackenty, Larry Petro, Vera Kozhurina-Platais, Abhijit Rajan, Linda Smith, Chris Sontag, and Leonardo Ubeda.

Citation:

In publications, refer to this document as:
Gonzaga, S., Hack, W., Fruchter, A., Mack, J., eds. 2012, The DrizzlePac Handbook. (Baltimore, STScI)

For the design of AstroDrizzle and the enhancements to the FITS format it has introduced, please reference:
A.S. Fruchter, W. Hack, N. Dencheva, M. Droettboom, P. Greenfield, 2010, "BetaDrizzle: A Redesign of the MultiDrizzle Package" in STSCI Calibration Workshop Proceedings, Baltimore, MD, 21-23 July 2010, eds. Susana Deustua & Cristina Oliveira, Space Telescope Science Institute, pp 376 - 381

Send comments or corrections to:
Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218
E-mail:help@stsci.edu

Table of Contents

Acknowledgments

Information in this handbook represents the cumulative experience and contributions of many members of the STScI community, including the WFPC2, WFC3, ACS, NICMOS and STIS instrument groups, the Observatory Support Group and the Science Software Branch.

The Drizzle code, which is at the core of MultiDrizzle and AstroDrizzle software, was originally developed by Richard Hook and Andrew Fruchter. It was subsequently implemented in the pipeline as MultiDrizzle, in an effort led by Anton Koekemoer.

AstroDrizzle, which is written primarily in C and Python, replaces MultiDrizzle in the HST pipeline. Software development was led by Andrew Fruchter and Warren Hack, with contributions from Erik Bray, Nadia Dencheva, Michael Droettboom, Richard Hook (ESO), Chris Sontag, and Megan Sosey. Under the leadership of ACS and WFC3 team leads, Linda Smith and John Mackenty, respectively, the software was tested by Amber Armstrong, Roberto Avila, Howard Bushouse, Michael Dulude, Shireen Gonzaga, Ray Lucas, Jennifer Mack, Max Mutchler, Larry Petro, Norbert Pirzkal, Abhijith Rajan, and Leonardo Ubeda. We thank Sylvia Baggett, Matthew Bourque, Stefano Casertano, Matt Lallo, Janice Lee, Knox Long, Josh Sokol, and Brad Whitmore for valuable feedback on the software. Susan Rose and James Younger provided the technical expertise for publication of this document.

The *DrizzlePac Handbook* replaces the Draft *AstroDrizzle Mini-Handbook* and the *MultiDrizzle Handbook*.

Introduction

The *DrizzlePac Handbook* was written for both novice users and seasoned "drizzlers." While users are free to use this document any way they wish, the following are some recommendations:

- Users new to HST data processing and analysis should familiarize themselves with STSDAS before using this handbook–the *HST Data Handbook Introduction* is a good place to start. For work on a specific instrument, a review of the instrument's data handbook is recommended. After that, they may want to take a brief look at Chapters 1 and 4 of this handbook, then take an AstroDrizzle test drive using the example in Section 7.1.
- There is a wide range of drizzling experience among users who are already familiar with HST data analysis.
  - Users new to drizzling should first review Chapters 1 and 2 to become familiar with Drizzle concepts, then learn about the DrizzlePac package in Chapters 4, 5, and 6. To get a taste of the software, try the example in Section 7.1.
  - Experienced Drizzlers may wish to start with learning about the differences between MultiDrizzle and AstroDrizzle (Chapters 1 and 4), learn about the new Python interfaces in Chapter 5, then come up to speed with DrizzlePac in Chapter 6. The rest of the document could be used as a searchable reference.

The user interface for the latest STSDAS software packages, including drizzlepac, is not the same as that in IRAF. Although a few common IRAF commands have been retained for convenience, most command-line syntax is now in Python, and users also have the option to run many tasks from a GUI called TEAL that is similar to the PyRAF GUI. Additional information is available in Chapter 5.

CHAPTER 1:

Introduction to AstroDrizzle and DrizzlePac

In this chapter…

1.1 What are AstroDrizzle and DrizzlePac?

AstroDrizzle replaced MultiDrizzle in the HST data pipeline in June 2012 (please check the DrizzlePac website for the exact date). It is also available to users who wish to reprocess data using non-default parameter settings that could improve the quality of their images. *AstroDrizzle*, an abbreviation for *Astro*metric *Drizzle*, was designed from the ground-up to improve the handling of astrometry and geometric distortion.

DrizzlePac is a software package that replaces the Dither package in STSDAS. It is a suite of supporting tasks for **astrodrizzle** which include:

- **tweakreg** and **tweakback** for aligning images in different visits
- **pixtopix** transforms an *X,Y* pixel position to its pixel position after distortion corrections
- **skytopix** transforms sky coordinates to *X,Y* pixel positions. A reverse transformation can be done using the task **pixtosky**

The Drizzle algorithm (Fruchter and Hook, 2002) was developed as a powerful method for combining dithered HST images. MultiDrizzle was later created to serve

as a wrapper script that sequentially performs the steps needed to combine dithered images using Drizzle, as well as remove cosmic rays in the final combined image.

For many years, MultiDrizzle (Koekemoer et al., 2002) served as the pipeline software workhorse for combining HST observations. It was particularly suited for dithered HST images in the same visit, where the telescope pointing is offset between exposures. MultiDrizzle used pointing information in the headers to align dithered data, then combined them while correcting for geometric distortion and removing most artifacts such as cosmic rays and bad pixels. Users were also able to use MultiDrizzle to reprocess their data to include images from other visits, and in some situations, regain some of the resolution lost in the original undersampled images.

A description of the DrizzlePac package which replaced the Dither package, including information about AstroDrizzle which replaced MultiDrizzle, was first published in the paper by Fruchter, A., et al, "BetaDrizzle: A Redesign of the MultiDrizzle Package" in the STSCI Calibration Workshop Proceedings, Baltimore, MD, 2010. AstroDrizzle continues to perform the same functions as MultiDrizzle. But it also holds significant improvements in how the data is processed, including better handling of astrometric information. Bugs in MultiDrizzle have also been fixed in this new software. In order to implement the new astrometric features, calibrated `flt.fits`[1] files from the data now have a slightly modified structure[2]. These issues will be covered in greater detail in subsequent chapters.

1.2 What's new in AstroDrizzle?

1.2.1 Code Improvements

While AstroDrizzle maintains much of the same algorithmic base as MultiDrizzle, this new code has undergone a number of substantial internal changes. Core routines have been re-coded in C and Python, written in a modular fashion for easier maintenance and updates.

All user interaction is performed using Python, either as command lines and/or using the *TEAL* (*Task Editor and Launcher*) Graphical User Interface (GUI) which is structured for easier use in setting task parameters.

1. `flt.fits` files are flat-field calibrated images that have not been corrected for geometric distortion. New **calacs** data products are now available with the suffix `flc.fits`; these are identical to `flt.fits` except for CTE corrections to the images.

2. When old- and new-format `flt.fits` files are run in **multidrizzle**, almost identical images are created. While **multidrizzle** will continue to be available in STSDAS, it is no longer supported.

1.2.2 A Fundamentally Different Approach to Handling Image Distortions and Astrometry

In MultiDrizzle, non-linear distortions in an image are expressed as a set of polynomial function coefficients that are distinct from the image *world coordinate system (WCS)*. These coefficients, unique for different types of observing modes, are stored in a reference file called the *IDCTAB (Instrument Distortion Coefficients TABle)*. The Drizzle program uses this reference file to extract coefficient values that correspond to the images being processed.

In contrast, AstroDrizzle incorporates linear distortion corrections and higher order polynomial distortion corrections directly into the WCS in `flt.fits` headers, using the *Simple Image Polynomial* (*SIP*) convention (Shupe, et. al, 2005). This convention has, for some time, been in use for describing the geometry of Spitzer Space Telescope images, and it is expected to become a FITS standard in the near future. Representing image distortion corrections using the SIP convention will improve the handling of image combination and astrometric information.

When AstroDrizzle becomes part of the *on-the-fly reprocessing* (*OTFR*) calibration system, users will notice some changes in calibrated (`flt.fits`) ACS and WFC3 images[3] retrieved from the HST Archive: distortion information is now directly incorporated into the image headers using FITS standard conventions. This new feature makes it easier for users to access distortion information for use in their own code. Publicly available software such as **ds9**, that can interpret these FITS conventions, are now able to convert between pixel positions and sky coordinates. For instance, when a `flt.fits`[4] image from the HST Archive is displayed using **ds9**, the astrometric Right Ascension (R.A.) and Declination (Dec.) positions from **ds9** show “undistorted” positions using the full polynomial distortion corrections.

1.2.3 Non-polynomial Distortions in ACS Data

ACS data have additional non-polynomial distortion corrections[5].

- ACS/WFC requires pixel column width corrections.
- All ACS cameras have residual optical distortions.

In the previous pipeline, MultiDrizzle obtained small scale distortion corrections from reference files identified by the header keyword `DGEOFILE` (Differential GEOmetric distortion). These reference images, as large as 256 MB, are similar in size to `flt.fits` images. `DGEOFILE` images were created by interpolating tables that contained residual distortion corrections. For instance, an ACS/WFC `DGEOFILE`

3. WFPC2 and NICMOS images, processed with MultiDrizzle, are in a static archive and will not undergo further pipeline processing.

4. `flt.fits` images from the HST Archive were processed by AstroDrizzle as part of the OPUS pipeline. While the image data remains unchanged, header keyword values and data quality information is updated.

5. Non-polynomial distortion corrections are currently not needed for WFC3 data.

reference image was created by interpolating a set of 32x64-entry tables representing residual distortion corrections into a full-size image, then combining that image data with the detector column width corrections. (More about this in Section 3.2.)

The new OPUS pipeline with AstroDrizzle inserts the non-polynomial distortion corrections, in tabular form, directly in the `flt.fits` file as FITS extensions. (**multidrizzle** can still be run using these new-format `flt.fits` images, and the output will remain the same as before. However, the software still requires `IDCTAB` and `DGEOFILE` reference files.)

As a result, ACS data files from the current OPUS pipeline have several new FITS extensions. The *D2IMARR* extension, only required for ACS/WFC, contains information on the geometry of the detector not described by the image SIP coefficients.

The *WCSDVARR* FITS extensions hold tabular data describing small-scale distortion effects due to optics. (There are two such extensions per detector: one for corrections to the x-axis, the other for y-axis corrections. Therefore, a WFC `flt.fits` image has four `WCSDVARR` extensions while HRC and SBC `flt.fits` images each have two `WCSDVARR` extensions.)

1.2.4 Astrometric Operations Made Easier with the DrizzlePac Package

The DrizzlePac package contains tasks that provide enhanced handling of astrometric information. These include,

- Updating image WCS information with a user-supplied catalog as a reference: for instance, in comparing a catalog of object positions derived from a HST image with a user-supplied astrometric catalog, the software can update the HST image WCS to bring its coordinate system into agreement with the user's astrometric catalog. This astrometric catalog could be a set of positions, perhaps created from another AstroDrizzle-processed HST image, or it could be derived from a standard catalog (such as 2MASS or Kepler).
- Corrected astrometric information can be fully captured in small FITS files called *headerlets* These headerlets can be sent to collaborators working on the same data, allowing just the image's astrometry component to be sent instead of the whole image. In addition, users will be able to attach several solutions to an image, for instance, one solution that best matches the 2MASS catalog and another that better agrees with UCAC3.

1.3 What's Different about Data from the Archive

As mentioned earlier, `flt.fits` images from the HST Archive now contain information that describes an image's full astrometric solution. As a result, image headers have several new keywords (listed in Section 3.4), and some image files, such as ACS data, have new FITS extensions that carry small-scale distortion corrections. This format includes the ability to give each astrometric solution a unique name, and to send it to another user as a FITS headerlet.

The PyRAF command **catfits** can be used to display the new FITS extensions structure for ACS `flt.fits` images, as shown in the example below for ACS/WFC:

```
--> catfits jb7310ymq_flt.fits<br>
 EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT


 0     jb7310ymq_flt jb7310ymq_flt.fits                       16
 1       IMAGE       SCI                   1     4096x2048    -32
 2       IMAGE       ERR                   1     4096x2048    -32
 3       IMAGE       DQ                    1     4096x2048    16
 4       IMAGE       SCI                   2     4096x2048    -32
 5       IMAGE       ERR                   2     4096x2048    -32
 6       IMAGE       DQ                    2     4096x2048    16
 7       IMAGE       D2IMARR               1     4096         -32
 8       IMAGE       WCSDVARR              1     65x33        -32
 9       IMAGE       WCSDVARR              2     65x33        -32
 10      IMAGE       WCSDVARR              3     65x33        -32
 11      IMAGE       WCSDVARR              4     65x33        -32
```

The first seven extensions ([0] thru [6]) should be familiar to ACS users:

- The primary header (group [0]).
- The chip 2 image, error, and data quality headers (groups [1], [2], and [3], respectively).
- The chip 1 image, error, and data quality headers (groups [4], [5], and [6], respectively).

Extensions [7] through [11] are new. As noted in Section 1.2, these extra extensions incorporate distortion correction information beyond that captured in the world coordinate system (WCS).

- The D2IMARR extension in group [7], in the case of ACS/WFC, is a one-dimensional vector containing corrections for physical distortions in the detector columns.
- The WCSDVARR[6] extensions, groups [8], [9], [10], and [11], describe residual geometric distortion corrections imposed by the optics, that is, corrections to distortions not modeled by the polynomial fit. For ACS/WFC, these are four 65x33 interpolation arrays.

The tabular information in FITS extensions [7] through [11] were previously represented by the DGEOFILE reference image that was used by MultiDrizzle, which was well over 250 MB in size.

A comparison of output drizzled images between the new AstroDrizzle pipeline and the old MultiDrizzle pipeline may also show small file size differences, or differences in the pixel number placement of a particular position in the output image. Nonetheless, astrometry of images with the default orientation, as used in the pipeline, are identical for AstroDrizzle and MultiDrizzle.

However, when AstroDrizzle was being tested, a problem was found for cases where MultiDrizzle was re-run outside the pipeline using a non-default orientation (the ***final_rot*** parameter set to a value other than ***INDEF***). In these instances, MultiDrizzle introduces a small (typically of order 0.1 arcseconds) error in the astrometry of output drizzled images. This bug has been fixed in AstroDrizzle. But it will not be corrected in any version of MultiDrizzle because development for MultiDrizzle has been stopped in favor of finishing AstroDrizzle in a timely fashion. This MultiDrizzle bug does not affect previously-retrieved MultiDrizzled archive data because those drizzled images were produced using default orientations.

During AstroDrizzle development and testing, several other problems were found in MultiDrizzle that affected data from the old pipeline. These bugs have been fixed in the new pipeline running AstroDrizzle:

- MultiDrizzle mishandles distortion corrections for some non-standard image formats.
  - For WFC3/UVIS binned data, MultiDrizzle does not properly account for binning when determining the distortion-corrected chip centers. Among other effects, this produces an extremely large gap between the two chips in the final drizzled image.
  - The ACS polarizer distortion model was not properly applied by MultiDrizzle; it was incorrectly specified in the IDCTAB reference table (which has since been fixed) and not correctly interpreted by MultiDrizzle software. Polarizer images, which are treated as sub-arrays, were not properly corrected to align with a distortion-corrected full-frame image at the same pointing. This error induced a scale difference of nearly 3%, and a rotation of nearly a full degree, relative to a properly distortion-corrected direct image of the same field. Drizzled polarizer images from the AstroDrizzle

6. Its name is in compliance with a FITS convention.

pipeline are now properly de-distorted, and distortion corrections are correctly represented in the flt.fits header SIP coefficients.

- Identification of cosmic rays during MultiDrizzle processing can be slightly skewed by high sky values in the image. Bad pixels and cosmic rays are identified by comparing each distorted input image with a comparison image created by applying the distortion model to an undistorted median-combined "clean" image. In MultiDrizzle, this comparison occurs after subtracting the sky from the input image. Pixels flagged as bad (cosmic rays and other defects) are identified based on a statistical noise model that does not account for sky values that have already been removed. This often results in pixels being flagged as "bad" by MultiDrizzle when they were actually fine. Such overzealous flagging of bad pixels is particularly noticeable in the diffraction spikes of bright stars.
- When MultiDrizzle computes the parameters necessary for running drizzle using *PyDrizzle* (an older implementation of the Drizzle code), some pixels get reported as "dropped" during the single drizzle step and final drizzle step. This happens because the computed size of the output image is not large enough, or the final image is not centered as expected by the code. This computation has been dramatically improved in AstroDrizzle to reduce (if not entirely eliminate) the number of pixels that end up outside the drizzled image.

CHAPTER 2:

Description of the Drizzle Algorithm

In this chapter…

2.1 Image Reconstruction and Restoration Techniques

There are two basic techniques used to recover spatial information in images while preserving the signal-to-noise ratio (SNR):

- Reconstruction, which attempts to recreate the image after it's been convolved with the instrumental Point Spread Function (PSF)
- Deconvolution, which tries to remove the effects of the PSF imposed on the "ideal" image by enhancing high frequency components which were suppressed by the optics and the detector.

The primary aim of these techniques is to recover image resolution while preserving the SNR. These goals are unfortunately not fully compatible. For example, non-linear image restoration procedures that enhance high frequencies in the image, such as the Richardson-Lucy (Richardson 1972; Lucy 1974; Lucy & Hook 1991) and maximum-entropy methods (Gull & Daniel 1978; Wier & Djorgovski 1990) directly exchange signal-to-noise for resolution, thus performing best on bright objects that have ample signal-to-noise.

An implementation of the Richardson-Lucy method is available as the IRAF/STSDAS task **acoadd**. However, this technique is unable to handle large dithers, and is limited by typical computing capabilities to combining either small

regions of many images, or the entire image of only a few dithers. Furthermore, the task is unable to accommodate geometric distortions and the changing shape of the PSF across the field of view. This technique, like all non-linear techniques, produces final images with noise properties that are difficult to quantify. In particular, this method has a strong tendency to clump noise into the shape of the input PSF.

The rest of this section focuses on a family of linear reconstruction techniques that, at two opposite extremes, are represented by the *interlacing* and *shift-and-add* techniques, with the Drizzle algorithm representing a continuum between these two extremes.

2.1.1 Interlacing

If the dithers are particularly well-placed, one can simply interlace the pixels from the images onto a finer grid. In the interlacing method, pixels from the independent input images are placed in alternate pixels on the output image according to the alignment of the pixel centers in the original images. For example, the image in the lower right of Figure 2.1 was restored by interlacing a 3x3 array of dithered images. However, due to occasional small positioning errors by the telescope, and non-uniform shifts in pixel space across the detector caused by geometric distortion of the optics, true interlacing of images is generally not feasible.

2.1.2 Shift-and-Add

Another standard simple linear technique for combining shifted images, descriptively named “shift-and-add”, has been used for many years to combine dithered infrared data onto finer grids. Each input pixel is block-replicated onto a finer subsampled grid, shifted into place, and added to the output image. Shift-and-add has the advantage of being able to easily handle arbitrary dither positions. However, it convolves the image yet again with the original pixel, thus adding to the blurring of the image and to the correlation of noise in the image. Furthermore, it is difficult to use shift-and-add in the presence of missing data (e.g., from cosmic rays) and geometric distortion.

2.1.3 Drizzle

In response to the limitations of the two techniques described above, an improved method known formally as variable-pixel linear reconstruction, and more commonly referred to as Drizzle, was developed by Andy Fruchter and Richard Hook (Fruchter and Hook 1997), initially for the purposes of combining dithered images of the Hubble Deep Field North (HDF-N). This algorithm can be thought of as a continuous set of linear functions that vary smoothly between the optimum linear combination technique (interlacing) and shift-and-add. This often allows an improvement in resolution and a reduction in correlated noise, compared with images produced by only using shift-and-add.

The degree to which the algorithm departs from interlacing and moves towards shift-and-add depends upon how well the PSF is subsampled by the shifts in the input images. In practice, the behavior of the Drizzle algorithm is controlled through the use of a parameter called *pixfrac*, which can be set to values ranging from 0 to 1, that represents the amount by which input pixels are shrunk before being mapped onto the output image plane.

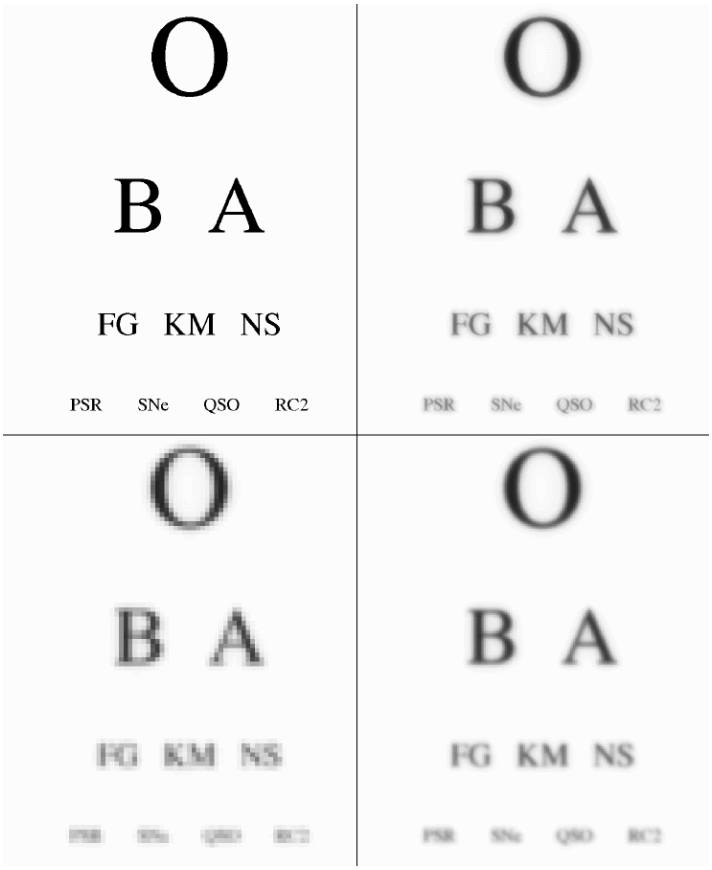
A key to understanding the use of pixfrac is to realize that a CCD image can be thought of as the true image convolved first by the optics, then by the pixel response function (ideally a square the size of a pixel), and then sampled by a delta-function at the center of each pixel. A CCD image is thus a set of point samples of a continuous two-dimensional function. Hence the natural value of pixfrac is 0, which corresponds to pure interlacing. Setting pixfrac to values greater than 0 causes additional broadening of the output PSF by convolving the original PSF with pixels of non-zero size. Thus, setting pixfrac to its maximum value of 1 is equivalent to shift-and-add, the other extreme of linear combination, in which the output image PSF has been smeared by a convolution with the full size of the original input pixels.

The Drizzle algorithm has also been designed to handle large dithers, where geometric distortion causes non-uniform subsampling across the field, and takes into account missing data resulting from cosmic rays and bad pixels. Other useful discussions on the reconstruction of Nyquist images from undersampled data, as well as the merits of various types of dither patterns, are presented by Lauer (1999a, 1999b), Arendt, Fixsen and Moseley (2000), and Anderson and King (2000). It is beyond the scope of the present documentation to provide an extensive discussion on the levels comparable to these papers, therefore we refer interested readers to these papers instead.

2.2 Drizzle Concept

High spatial frequency information in an image that is permanently smeared out by the detector pixel response can be partly recovered by combining subpixel dithered images. Each dithered image can be thought of as sampling a final higher resolution image–a “true image” of the sky. But the images are also convolved with the optical PSF and pixel response function of the detector. The effect of undersampling is illustrated in a set of four eye chart image examples shown below Figure 2.1. The upper left image represents a “true” image, as seen by a telescope of infinite aperture. The upper right image has been convolved with the HST/WFPC2 PSF. In the lower left of the set, the previously-mentioned image has been sampled by the WF2 CCD. The loss of spatial information is immediately obvious.

Figure 2.1: The Drizzle 'Eye Chart' Illustrating Convolution and Sub-Sampling

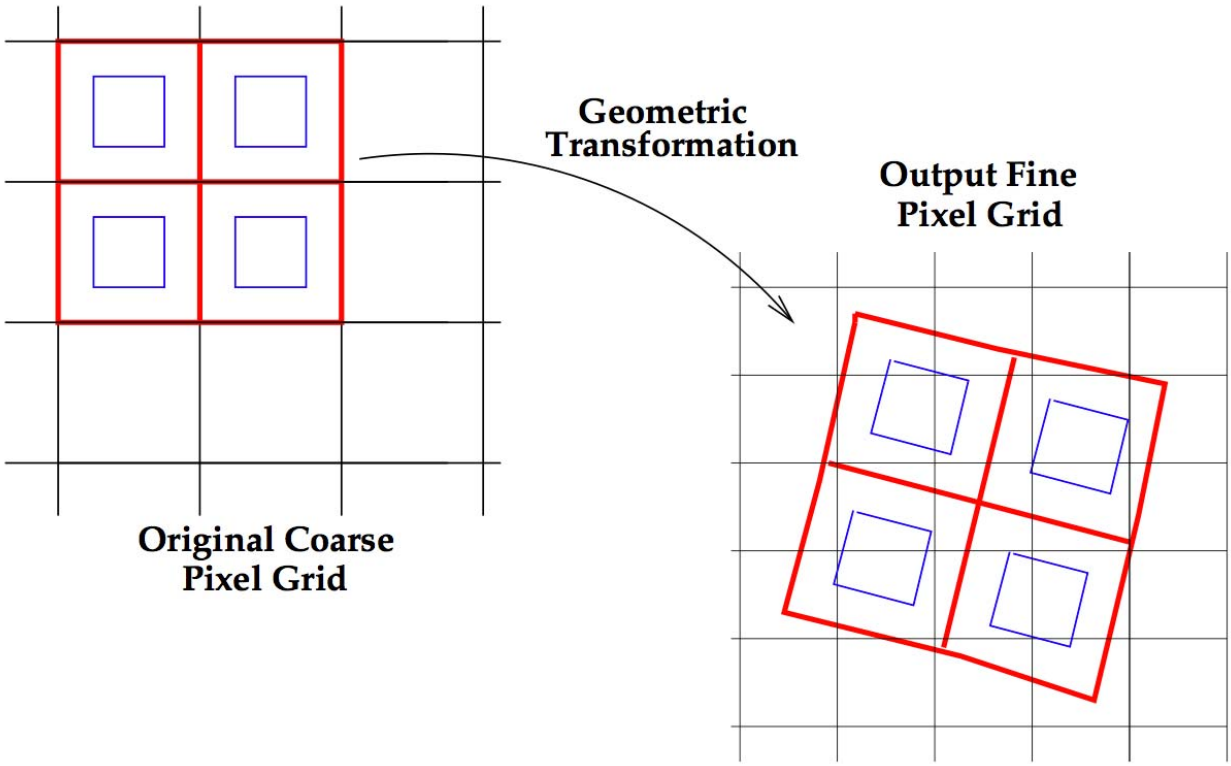


The effects of image convolution and subsampling The upper left image represents a "true" image, as seen by a telescope of infinite aperture. The upper right image has been convolved with the HST/WFPC2 PSF. The effect of sampling it with the WF2 CCD, as seen in the lower left image, shows even more loss of spatial information. The lower right image has been reconstructed using the Drizzle algorithm.

Much of the information lost to undersampling can be recovered. This is shown in the lower right of Figure 2.1, where the image has been recovered using a method from a family of techniques known as "linear reconstruction." However, simple implementations of these techniques generally introduce additional blurring due to convolution with the pixel shape. This effect can be seen directly in the present example by comparing the upper and lower right-hand images: the deterioration in image quality between these two images is due entirely to convolution of the image with the pixel.

The Drizzle algorithm is conceptually straightforward, as shown in Figure 2.2. Pixels in the original input images are mapped into pixels in the subsampled output image, taking into account shifts and rotations between images and the optical distortion of the camera. However, in order to avoid convolving the image with the large pixel "footprint" of the camera, Drizzle allows the user to shrink the pixel before it is averaged into the output image through the pixfrac parameter.

Figure 2.2: Schematic Representation of how Drizzle Maps Input Pixels onto the Output Image



The new shrunken pixels, or "drops," rain down (or "drizzle") upon the subsampled output image, as shown in Figure 2.2. The "drop" size is controlled by the parameter pixfrac, the ratio of the linear size of the "drop" to the input pixel (before any adjustment due to the geometric distortion of the camera). The size of the drop is further adjusted internally by the Drizzle code to take into account the camera geometric distortion, before the overlap of the drop with pixels in the output image is determined. A second parameter, scale, allows the user to specify the size of the output pixels in arcseconds[1].

The flux value of each input pixel is divided up into the output pixels with weights proportional to the area of overlap between the "drop" and each output pixel. If the drop size is too small, not all output pixels have data added to them from each of the input images. One should therefore choose a drop size that is small enough to avoid convolving the image with too large an input pixel footprint, yet sufficiently large to ensure that there is not too much variation in the number of input pixels contributing to each output pixel.

1. In the case of the Hubble Deep Field North (HDF-N), the drop size linear dimensions were one-half of the input pixel (i.e., ***pixfrac=0.5***). This drop size was slightly larger than the dimensions of the output subsampled pixels which were four-tenths (0.4) the size of WFC pixel (***scale***=0.04, units in arcseconds).

2.3 Weight Maps and Correlated Noise

2.3.1 Weight Maps

When images are combined using Drizzle, a weight map can be specified for each input image. The weight image contains information about bad pixels in the image (in that bad pixels result in lower weight values). This weight image can be provided by the user, or it can be created by AstroDrizzle according to several automatic schemes (that will be discussed in Chapter 4). When the final output science image is generated, an output weight map which combines information from all the input weight images, is also saved.

When a drop of value i_{xy} and user defined weight w_{xy} is added to an output image I_{xy}, with weight W_{xy} and a fractional pixel overlap of $0 < a_{xy} < 1$, the resulting value of the image I'_{xy} and W'_{xy} is:

$$W'_{xy} = a_{xy} w_{xy} + W_{xy}$$

$$I'_{xy} = \frac{a_{xy} i_{xy} w_{xy} + I_{xy} W_{xy}}{W'_{xy}}$$

Drizzle has a number of advantages over standard linear reconstruction methods. Since the pixel area can be scaled by the Jacobian of the geometric distortion, it is preserved for surface and absolute photometry. Therefore, the flux in the drizzled image, that was corrected for geometric distortion, can be measured with an aperture size that's not dependent of its position on the image. Since the Drizzle code anticipates that a given output pixel might not receive any information from an input pixel, missing data does not cause a substantial problem as long as the observer has taken enough dither samples to fill in the missing information.

The output pixels in the final drizzled image are not independent of one another, causing the noise in the output image to be correlated to some degree. In principle, the correlated noise can be fully described by creating a correlation image. However, the implementation of such schemes becomes complicated when images are shifted at subpixel scales. A more practical approach is to use the weight maps generated by Drizzle to calculate the expected RMS noise. The weight appropriate to a given value of the scale parameter (expressed here as the ratio of the output to input pixel size), can be calculated in the following way (as described by Casertano et al. 2000):

For WFPC2 and NICMOS, by definition, the inverse flat field is contained in the flat-field reference file, *f*. In the pipeline, the image is multiplied by *f*. Therefore,

$$Var = \frac{[(f(D+B)/g) + \sigma^2]}{(f^2 t^2)}$$

For STIS, ACS, and WFC3, by definition, the flat field is contained in the flat-field reference file, *f*. In the pipeline, the image is divided by *f*. Therefore,

$$Var = \frac{[(f(D+B)/g) + \sigma^2 f^2)]}{(t^2)}$$

Therefore, the weight is

$$W = \frac{1}{(Var \times scale^4)}$$

where

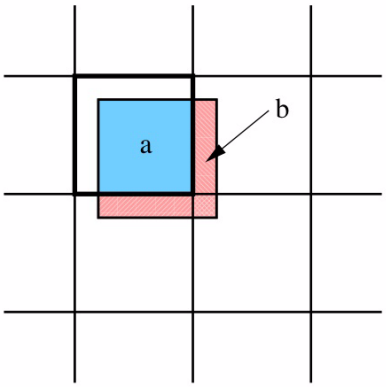
- *D* and *B* are the counts per pixel (in DN) due to the dark current and background, respectively, averaged over the entire image
- *t* is the exposure time in seconds
- *g* is the gain of the detector–users should be aware of the units of their image and use the appropriate gain value
- σ is the read noise in DN/pixel–a more in-depth discussion of noise in drizzled images can be found in Section 4.2.12

2.3.2 Weight Maps and Correlated Noise

Drizzle frequently divides the power from a given input pixel between several output pixels. As a result, the noise in adjacent pixels will be correlated. Understanding this effect in a quantitative manner is essential for estimating statistical errors when drizzled images are analyzed using object detection and measurement programs such as *SExtractor* (Bertin and Arnouts 1996) and *DAOPHOT* (Stetson 1987).

This noise correlation of adjacent pixels implies that a measurement of noise in a drizzled image–on the output pixel scale–underestimates the noise on larger scales. In particular, if one block-sums a drizzled image by $N \times N$ pixels, even using a proper weighted sum of the pixels, the per pixel noise in the block-summed image will generally be more than a factor of *N* greater than the per pixel noise of the original image. The factor by which the ratio of these noise values differ from *N* in the limit as $N \to \infty$ is referred to as the noise correlation ratio, *R*. One can easily see how this situation arises by examining Figure 2.3.

Figure 2.3: Distribution of Noise from a Single Input Pixel between Neighboring Output Pixels



A schematic view of the distribution of noise from a single input pixel between neighboring output pixels. (This figure and much of the discussion of correlated noise are taken from Fruchter and Hook 2002.)

In Figure 2.3 an input pixel (broken up into two regions, *a* and *b*) is being drizzled onto an output pixel plane. Let the noise in this pixel be ε and let the area of overlap of the drizzled pixel with the primary output pixel (shown with the heavier border) be a, and the areas of overlap with the other three pixels be b_1, b_2 and b_3, where $b = b_1 + b_2 + b_3$ and $a + b = 1$. Now, the total noise power added to the image variance is ε^2; however, the noise that one would measure by simply adding up the variance of the output image pixel-by-pixel would be:

$$(a^2 + b_1^2 + b_2^2 + b_3^2)\varepsilon^2 < \varepsilon^2$$

This inequality exists because all cross terms (ab_1, ab_2, b_1b_2 ...) are missed by summing the squares of the individual pixels. These terms, which represent the correlated noise in a drizzled image, can be significant.

The Calculation

In general, the correlation between pixels, and thus, the noise correlation ratio R, depends on the choice of Drizzle parameters, as well as geometry and orientation of the dither pattern. And R often varies across an image. While it is always possible to estimate R for a given set of Drizzle parameters and dithers, in the case where all the output pixels receive equivalent inputs (in both dither pattern and noise, though not necessarily from the same input images) the situation becomes far more analytically tractable. In this case, calculating the noise properties of a single pixel gives one the noise properties of the entire image.

Consider the situation when pixfrac, p, is set to zero: there is no correlated noise in the output image since a given input pixel contributes only to the output pixel which lies under its center, and the noise in the individual input pixel is assumed to be independent. The expected variance of the noise in that output pixel, when $p = 0$, is simply:

$$\sigma_c^2 = \frac{\sum d_{xy} \varepsilon C w_{xy}^2 s^4 \sigma_{xy}^2}{\left(\sum d_{xy} \varepsilon C w_{xy}\right)^2}$$

where,

- d_{xy} represents a pixel from any of the input images
- C is in the set of all d_{xy} with centers that fall on a given output pixel of interest
- ε is the standard deviation
- w_{xy} is the pixel weight
- s is the scale
- σ_{xy} is the standard deviation of the noise distribution of the input pixel d_{xy}

Here, σ_c is the standard deviation that is calculated only for cases when the input pixel value fell on the center of the output pixel.

Now consider a drizzled output image where the pixfrac $p > 0$. In this case, the set of pixels contributing to an output pixel will not only include input pixels with centers that fall on the output pixel, but also those pixels where a portion of the drop lands on the output pixel of interest (even when the input pixel center is not included in that portion). The set of all input pixels with overlapping drops on a given output pixel is P, where $C \subset P$.

The variance of the noise in a given output pixel is then:

$$\sigma_p^2 = \frac{\left(\sum d_{xy} \varepsilon P a_{xy}^2 w_{xy}^2 s^4 \sigma_{xy}^2\right)}{\left(\sum d_{xy} \varepsilon P w_{xy}\right)^2}$$

where,

- a_{xy} is the fractional area overlap of the drop of input data pixel d_{xy} with output pixel o

The symbol σ_p represents the standard deviation calculated from all pixels that contribute to the output pixels when pixfrac = p.

The degree to which σ_p^2 and σ_C^2 differ depends on the dither pattern and the values of p and s. However, as more input pixels are averaged together to estimate the value of a given output pixel in P than in C, $\sigma_p^2 \leq \sigma_C^2$. When $p = 0$, σ_p is, by definition, equal to σ_C.

Now consider doing a weighted sum of the image pixels, where a region of $N \times N$ pixels in the final drizzled image is block-averaged. This sum is equivalent to having drizzled onto an output image with a scale size N_s. But as $N_s \gg p$, this approaches the sum over C; or in the limit of large N, it becomes $N\sigma_C$. However, a prediction of the noise in this region, based solely on a measurement of the pixel-to-pixel noise, without taking into account the correlation between pixels would produce $N\sigma_C$. Therefore,

$$R = \frac{\sigma_C}{\sigma_p}$$

For a given set of Drizzle parameters and dither patterns, R can be obtained by calculating σ_C and σ_p and performing the division. However, there is a further simplification that can be made: by the assumption that inputs to each pixel are statistically equivalent, it follows that weights of the individual output pixels in the final drizzled image are independent of the choice of p. To see this, notice that the total weight of a final image (the sum of the weights of all the pixels in the final image) is independent of the choice of p. Ignoring edge pixels, the number of pixels in the final image with non-zero weight is also independent of the choice of p. Yet as the fraction of pixels within p of the edge scales as $1/N$, and the weight of an interior pixel cannot depend on N, it can be seen that the weight of an interior pixel must also be independent of p. As a result,

$$\sum_{d_{xy} \varepsilon C} w_{xy} = \sum_{d_{xy} \varepsilon p} a_{xy}^2 w_{xy}$$

Therefore,

$$R^2 = \frac{\sigma_C^2}{\sigma_p^2} = \frac{\sum d_{xy} \varepsilon C a_{xy}^2 w_{xy}^2 \sigma_{xy}^2}{\sum d_{xy} \varepsilon p a_{xy}^2 w_{xy}^2 \sigma_{xy}^2}$$

Although R must be calculated for any given set of dithers, there is one case that is particularly illustrative when there are many uniformly placed dithers across the pixel–this can approximate the effect of the dither pattern on the noise by assuming that the dither pattern is entirely uniform and continuously fills the output plane. In this case, the sums in the equations above become integrals over the output pixels, and therefore, it is not hard (though somewhat tedious) to derive R. If $r = p/s$, where p is pixfrac and s is *scale*, then in the case of a filled uniform dither pattern,

$$\text{If } r \geq 1,\ R = \frac{r}{1 - \frac{1}{3r}}$$

$$r \leq 1,\ R = \frac{1}{1 - \frac{r}{3}}$$

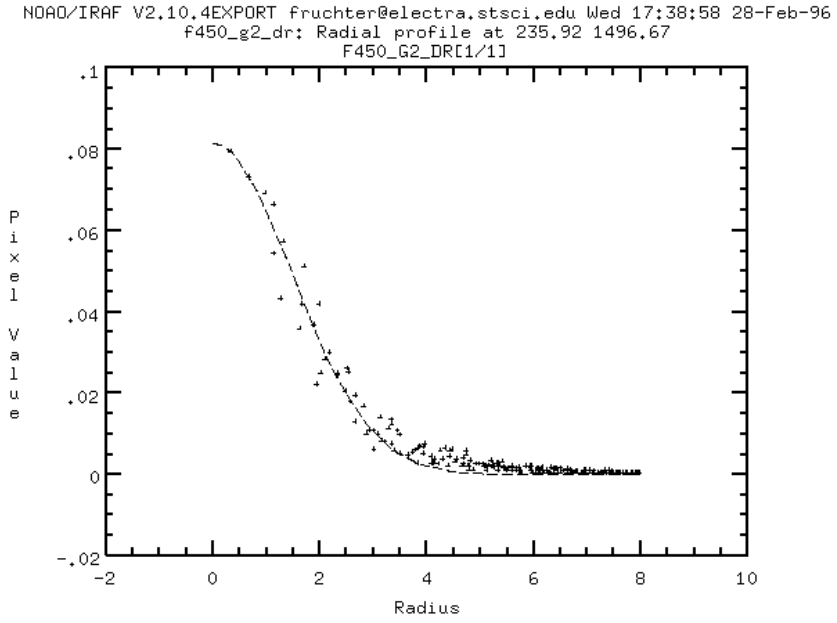
Using the relatively typical values of $p = 0.6$ and $s = 0.5$, one finds $R = 1.662$. This formula can also be used when block summing the output image. For example, a weighted block-sum of $N \times N$ pixels is equivalent to drizzling into a single pixel of size $\mathbf{N_s}$. The correlated noise in the block-summed image can be estimated by replacing s with $\mathbf{N_s}$ in the above expressions.

2.4 Characteristics of Drizzled Data

2.4.1 Characteristics of Drizzled Data: Sampling

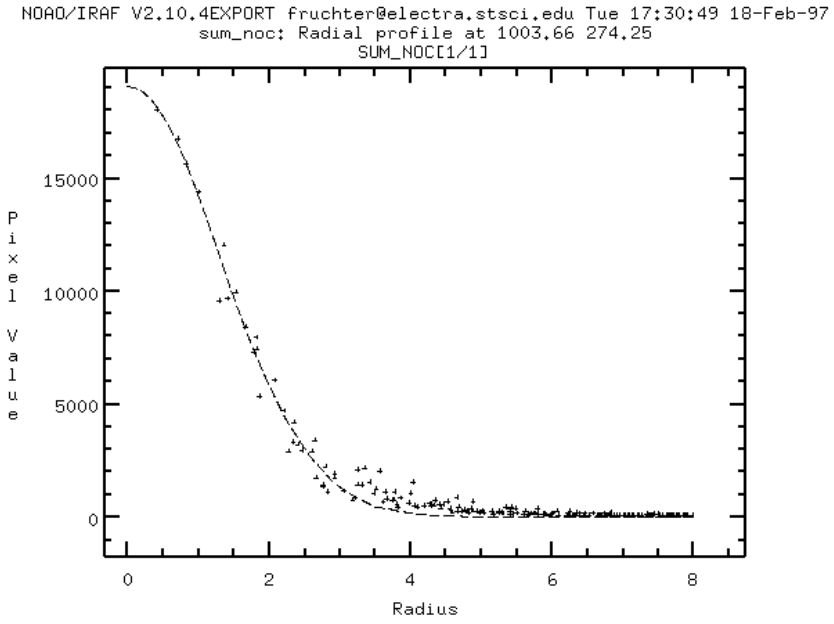
The theory of the Drizzle algorithm posits that the weight of an input pixel in the final output image is independent of its position on the chip. Therefore, if the dithered images do not uniformly sample the field, the center of light in an output pixel may be offset from the center of the pixel, and this offset may vary between adjacent pixels. Furthermore, the distortion present in the imaging instruments on board HST produces sampling patterns that are not uniform across the field, due to the changing pixel size. This directly impacts the uniformity of the output PSF

Figure 2.4: PSF Taken Directly from the HDF-N F450W Drizzled Image



This PSF was taken directly from the HDF-N F450W drizzled image. It shows substantial variation about the Gaussian due to the effects of non-uniform sampling, as well as possible additional charge transfer effects in the CCD.

Figure 2.5: Bright Star PSF from a Deep Image with Nearly Perfect Four-point Dither



This PSF from the HDF-N (Fruchter and Hook 1997) is a bright star taken from a deep image with a nearly perfect four-point dither. The plot clearly shows an improvement in the PSF resulting from more uniform sampling.

This effect is seen in the HDF-N images, where some pointings were not at the requested position or orientation. Figure 2.4 and Figure 2.5 show two PSFs compared with best-fitting Gaussians. Although Gaussians are only a crude approximation to the real PSF, they nevertheless suffice to illustrate the point of this particular example. The upper PSF is taken directly from the HDF-N F450W drizzled image, and displays a substantial amount of variation about the Gaussian fit. In contrast, the lower PSF is a bright star taken from a deep image with a nearly perfect four-point dither, in which the uniform sampling has produced a much smoother PSF. (Note that the difference in the apparent widths of the PSFs is due to the use of larger output pixels in the second image than in the HDF-N: 0.05 vs. 0.04 arcseconds).

Changes in PSF can also result from other problems, such as charge transfer errors in the CCD (Whitmore & Heyer 1997; Heyer 2001). Generally, however, these variations are likely to be less noticeable than effects due to non-uniform subsampling of the PSF.

2.4.2 Characteristics of Drizzled Data: Photometry

AstroDrizzle Aperture Photometry Accuracy

A photometric study of ACS/WFC images, done during extensive testing of the DrizzlePac package, has demonstrated that it is possible to achieve optimal aperture photometry using output from AstroDrizzle, provided that the combined images are carefully aligned and cosmic rays are properly removed.

Tests were run on ACS/WFC images, taken in F606W and F814W, of the open cluster NGC 6791 (Program 9815, PI: I. King). Observations for each filter were taken during separate visits, For each filter, three exposures, each 30 seconds, were taken using `POS TARG` shifts in a three-way subpixel dither.

For each set, images were aligned using the **tweakreg** task in **drizzlepac** that computes residual shifts between input images (`flt.fits`) and updates their headers with new WCS information that aligns the images. AstroDrizzle was used to combine each set of images, with careful attention to cosmic ray rejection parameters and image alignment accuracy, to avoid pixels that were not properly masked.

AstroDrizzle is able to provide improved sampling of the PSF relative to the individual input images. This is especially important for wide field HST cameras such as ACS, WFC3/UVIS, and WFC3/IR, where the size of a native pixel is comparable to the full width at half maximum (FWHM) of the PSF. However, for testing purposes, the native size of the ACS/WFC input pixel (0.05 arcseconds) was used for the combined image output scale.

Aperture photometry was performed using the PyRAF package DAOPHOT on a catalog of about 1500 stars. Stellar instrumental magnitudes calculated using individual `flt.fits` images from the HST pipeline were compared with instrumental magnitudes of the same sources using the AstroDrizzle-combined image.

Results indicate that optimal aperture photometry can be obtained from AstroDrizzle-combined images as long as the processing carefully accounts for image alignment and proper cosmic ray removal. A complete description of the NGC 6791 study is available online at the DrizzlePac website.

A more elaborate independent study with similar results, is provided by Kalirai, et al. 2012, AJ, 143, 11.

2.4.3 Characteristics of Drizzled Data: Astrometry

Astrometry of images taken within the same visit and orbit are generally limited by

- HST's pointing precision which is controlled by the Fine Guidance Sensors (FGS)
- Positional uncertainties of guide stars in the Guide Star Catalog

Information about the field, from ground- or space-based images, could be used to improve the inherent absolute astrometry of image.

For images with sufficient well-exposed point sources, relative astrometry for each image can be improved through source detection and centroiding. These updates can then be folded back into the astrometry information in the image header of each image and used to combine all the exposures into a single well-aligned and drizzled mosaic.

Figures 2.6 and 2.7 show the offset fit solution results from running **tweakreg** on two images of NGC 104, obtained using WFC3/UVIS with F606W, observed in the same orbit. The fit solution for matching objects in both images were highly accurate to an RMS of about 0.03 pixels in X and Y, and showed random residual vectors across the field.

More in-depth information about astrometry and positional uncertainties are available in Chapter 3 and Appendix A.

Figure 2.6: Fit Solution RMS for two WFC3/UVIS Images Taken in the Same Orbit

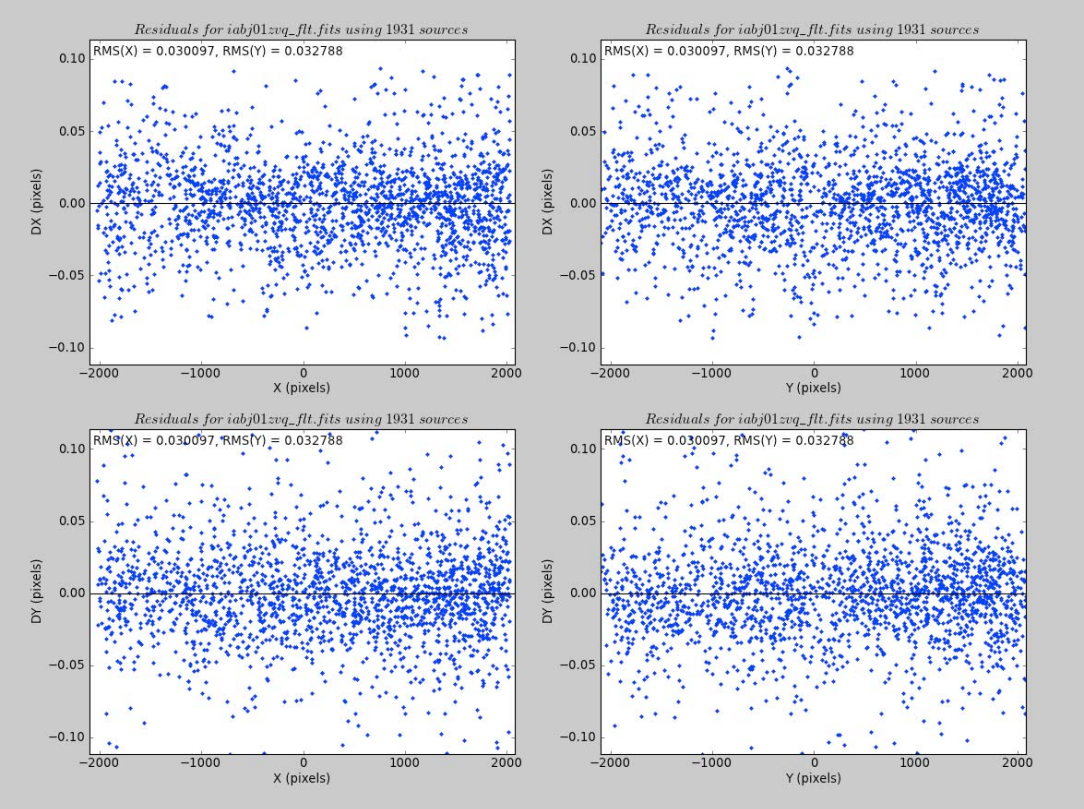
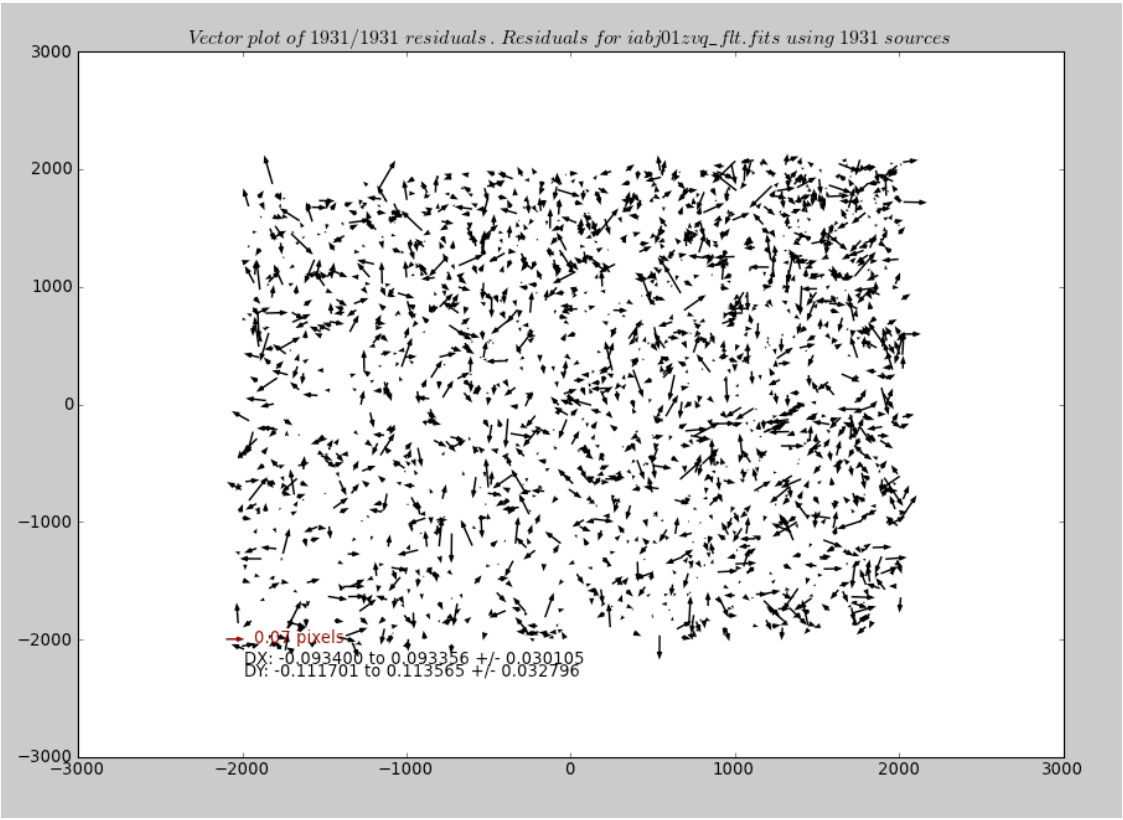


Figure 2.7: Vector of Shift Residuals for Two Images Taken in the Same Orbit



CHAPTER 3:

Astrometric Information in the Header

In this chapter…

3.1 Introduction

A major enhancement in AstroDrizzle software is its handling of astrometric information. In the pipeline, AstroDrizzle uses the Simple Image Polynomial (SIP) convention (Shupe, et. al, 2005). Pipeline image headers contain keywords, some that are new, that describe two types of distortion corrections: linear distortion corrections that are incorporated into the *CD matrix*, and non-linear distortion corrections expressed as simple image polynomial (SIP) quadratic functions. For the latter, polynomial distortion coefficients are stored in the `IDCTAB` reference file. These are the only two types of geometric distortion corrections required for WFC3, STIS, and NICMOS data.

A third class of distortions, called residual or non-polynomial distortions, cannot be expressed as functions. Corrections for non-polynomial distortions come in the form of look-up tables that are stored in the image as FITS extensions. Therefore, it is no longer necessary to use the DGEOFILE reference file in AstroDrizzle processing, as was done in running MultiDrizzle. There are also several new header keywords associated with these look-up tables.

WFPC2 data is only available from the *static Archive*, where it was processed by MultiDrizzle. Users who wish to reprocess that data using AstroDrizzle only require the relevant distortion correction files for their images so that AstroDrizzle can use

them to directly update the images. (Please see the example in Chapter 7 for more information.)

All three distortion correction operations for ACS, and the linear and polynomial distortion corrections for WFC3, are performed in the pipeline by AstroDrizzle and associated tasks. This chapter describes the new distortion correction features in more detail.

3.2 How Distortions are Represented in AstroDrizzle

3.2.1 Design of the AstroDrizzle Distortion Representation

The best information about instrument aperture locations and orientations, pixel scales, and the image distortion correction model is contained in a HST reference file called the Instrument Distortion Correction reference table (named in the FITS header keyword `IDCTAB`).

Astrometric calculations by MultiDrizzle use distortion correction information in the `IDCTAB` and `DGEOFILE` reference files, as well as information in the image header about telescope orientation, spacecraft velocity vector for velocity aberration correction, and date of observation to determine time-dependent skew in ACS. Even though SIP keywords were present in image headers from the previous HST Archive pipeline, actual astrometric calculations were done using the `IDCTAB` and `DGEOFILE` reference files.

With AstroDrizzle in the calibration pipeline, the `IDCTAB` reference file is now only used in the HST Archive pipeline to create and populate SIP header keywords in images.

Non-polynomial distortion correction information, previously supplied by the `DGEOFILE` reference file, are now contained in two new reference files given by the image header keywords `NPOLFILE` (Non-POLynomial distortion correction) and `D2IMFILE` (Detector to IMage distortion correction). The `NPOLFILE` reference file contains tabular information about residual distortion corrections for some instrument modes. The `D2IMFILE` reference file contains column (or row) width corrections for some instrument chips. During pipeline processing of images that require non-polynomial distortion corrections, the relevant information is obtained from these reference files and stored as new FITS extensions in the image data. A more detailed description is provided later in this section.

Three major forms of distortion corrections are implemented in AstroDrizzle.

1. Linear distortion corrections–scale, rotation, and time-dependent skew–are incorporated into the CD matrix.
2. Non-linear distortion corrections expressed as simple image polynomial (SIP) quadratic functions.
3. Optional additional non-polynomial distortion corrections.

- Non-optical detector variations (i.e., column width variations).
- Residual distortion corrections not described by the SIP polynomial functions.

The implementation of distortion corrections to an image works in the following way:

1. Apply the detector to image correction (DET2IM) calibration step to input pixel values.
2. Apply the SIP coefficients to DET2IM-corrected pixel values.
3. Apply the non-polynomial look-up table corrections to DET2IM-corrected pixel values.
4. Add the results of the SIP and look-up table corrections.
5. Apply the WCS transformation in the CD matrix to the summed results to get intermediate world coordinates.
6. Add the Right Ascension and Declination position at the reference pixel, CRVAL1 and CRVAL2 keyword values, to the transformed positions to get the positions on the tangent plate.
7. Apply the inverse projection from the tangent plane to the celestial sphere to get the true world coordinates.

This transformation can be conceptualized as:

$$(u,v) = DET2IM(x,y)$$

where column corrections applied to initial coordinates (*x*,y) using the DET2IM convention produces the coordinates (*u*,*v)*.

In the OTFR pipeline, this correction is only done for ACS/WFC data using the detector correction table stored in FITS image extension type D2IMARR. Other ACS detectors, and all WFC3 detectors do not require the D2IMARR correction, therefore,

$u = x$ and $v = y$.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u - CRPIX1 \\ v - CRPIX2 \end{bmatrix}$$

where u',v' are the DET2IM-corrected coordinates relative to CRPIX1,CRPIX2.

$$\begin{bmatrix} \alpha \\ \delta \end{bmatrix} = \left(\begin{bmatrix} CRVAL1 \\ CRVAL2 \end{bmatrix} + P^{-1} \left[\begin{bmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_2 \end{bmatrix} \begin{bmatrix} u' + f(u',v') + LT_x(x',y') \\ v' + g(u',v') + LT_y(x',y') \end{bmatrix} \right] \right)$$

where,

- (α,δ) is the source position on the celestial sphere
- CRVAL1, CRVAL2 are the Right Ascension and Declination position of the reference pixel

- `CD1_1`, `CD1_2`, `CD2_1`, `CD2_2` are CD matrix keyword values describing linear distortions: plate scale, rotation, and skew of the image
- LT_x, LT_y are the residual distortion corrections in the look-up reference tables that are appended to the image as FITS extensions (using the Paper IV look-up table convention).
- P^{-1} is the de-projection of the tangent plane back onto the celestial sphere
- *f(u',v')* and *g(u',v')* represent the polynomial distortion correction specified as

$$f(u', v') = \sum_{p+q=2}^{AORDER} A_{pq} u'^{p} v'^{q}$$

$$g(u', v') = \sum_{p+q=2}^{BORDER} B_{pq} u'^{p} v'^{q}$$

The type of tangent plane projection used in the creation of the distortion correction polynomial coefficients is described by the `CTYPE1` and `CTYPE2` header keywords. For AstroDrizzled data, their values are `RA---TAN-SIP` and `DEC--TAN-SIP`, respectively.

3.2.2 World Coordinate System

Definition

Astrometric information in FITS images (in the form of World Coordinate System or WCS) is stored in image headers using a standard set of keywords:

- `CRVAL1`: right ascension (α) at a reference pixel
- `CRVAL2`: declination (δ) at a reference pixel
- `CRPIX1`: the *x* location of the image reference pixel
- `CRPIX2`: the *y* location of the image reference pixel
- `CTYPE1`[1]: the coordinate type for the first axis, value is `RA---TAN-SIP`
- `CTYPE2`: the coordinate type for the second axis, value is `DEC--TAN-SIP`

The CD matrix is defined by the partial derivatives of the world coordinates with respect to the pixel coordinates as evaluated at the reference pixel. Its keywords are:

- `CD1_1` is the partial[2] of first axis coordinate w.r.t. *x*
- `CD1_2` is the partial of first axis coordinate w.r.t. *y*

1. Type of tangent plane projection used for distortion correction polynomial coefficients

2. The "partial" refers to the change in coordinate value (R.A. or Dec.) along an axis (*x* or *y*), at the reference pixel.

- CD2_1 is the partial of second axis coordinate w.r.t. x
- CD2_2 is the partial of second axis coordinate w.r.t. y

Computation

These keyword values get computed using the operations:

$$\begin{bmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_2 \end{bmatrix} = scale \times \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} A_{10} & A_{11} \\ B_{10} & B_{11} \end{bmatrix}$$

The plate scale, *scale*, represents the average plate scale (in decimal degrees) for the reference pixel after removing all distortion.

The value of θ originates from the PA_V3 keyword; this is the angle eastward from North to the telescope's V3 axis. However, the CD matrix describes a tangent plane projection, and PA_V3 describes the orientation of the telescope at the center of its field of view. Therefore, the orientation at the reference position for the image needs to be computed by projecting the PA_V3 orientation onto the detector coordinate system.

The orientation of the Y axis at the center of the instrument's field of view is then computed relative to the -V3 axis to obtain the value of the PA_APER keyword. This step takes into account the orientation of the detector's axes relative to the V2-V3 coordinate system. Take for example, the 45° degree rotation of the WFPC2 field of view relative to the V2-V3 axes; the orientation for each chip's reference position, θ, gets computed by projecting the tangent plane for the full field of view with an orientation of PA_APER onto each chip's reference point. The orientation of this tangent plane at each chip's reference point is recorded in the ORIENTAT keyword.

A_{10}, A_{11}, B_{10}, B_{11} refer to the linear terms of the distortion model as defined in the polynomial stored in the IDC table (IDCTAB) reference file. These terms represent the skew, change in orientation, and plate scale in X and Y at the reference pixel of the image only. More information about geometric distortion and SIP keywords is available in Section 3.3.

The right ascension (R.A.) and declination (Dec.) of the image target is stored in the image header keywords RA_TARG and DEC_TARG. These values are, however, the same for all images in a dithered set since the target itself is not changing. However, the R.A. and Dec. values (CRVAL1, CRVAL2) at the image reference pixel (CRPIX1, CRPIX2) will always be unique for each image. Therefore, the offset between the images (x and y shift) can be determined by retrieving the values of the associated reference pixel keywords.

Correcting for Velocity Aberration

The velocity aberration for each exposure gets reported as the VAFACTOR keyword value, computed as a fractional scale change induced by the velocity of the telescope during the observation. This keyword has values near 1 and represent the change in plate scale for the image. The CD matrix simply gets multiplied by this term to correctly represent the actual pixel scale on the sky for this specific exposure.

Application of the velocity aberration is performed when the WCS gets recomputed by AstroDrizzle to include all the distortion terms–this is done by setting a parameter to execute the **updatewcs** task in **astrodrizzle**. More detailed information about velocity aberration can be found in Appendix A.

Updating the WCS Header Information

World Coordinate System Information contained in the header of each image is as accurate as the guide star catalog information used for pointing the telescope, as well as the level of focal plane alignment errors, which evolve over time. Typical accuracy of the Guide Star Catalog 1 (GSC1) was about 1 arcsecond. GSC1 was the operational catalog until mid-2006 (cycle 14/15 boundary). As a result of this catalog accuracy, combined with focal plane evolution seen during this era, observations taken of the same field at different orientations with different guide stars could see offsets of a couple of arcseconds. From cycle 15 onwards, GSC2, with positional errors of ~0.25 arcseconds (1 σ) has been the operational catalog. A more stable focal plane, with calibration errors kept below this level has meant that since this time, offsets of the same target at different rolls and guide stars have typically been ~0.3 to 0.5 arcseconds.

Users may choose to calculate offsets due to different visits to align their images more accurately, then incorporate those shifts into WCS information in the image headers. This facilitates proper translation of pixel to sky coordinates for science analysis of the images, based on the WCS of a reference image, and stores the offset information for each image in the header for future alignment reference.

It's also important to update WCS keywords in the image header based on distortion correction solutions (such as geometric distortion and velocity aberration) applied to the data during drizzling. The full distortion solution including velocity aberration correction is applied by the AstroDrizzle code-set using of the **makewcs** task. Primary keywords affected are the CD matrix keywords; namely, `CD1_1`, `CD1_2`, `CD2_1`, and `CD2_2`. These keywords not only contain the plate scale and orientation at a reference pixel, but also linear terms of all distortion corrections. Higher order terms of the distortion model are reported in the remainder of the SIP keywords. Users who wish to apply their own distortion corrections should be careful to update these keywords.

3.2.3 The Simple Image Polynomial (SIP) Convention

In using the SIP convention (Shupe et al., 2005), pixel coordinates are transformed to sky coordinates using the CD matrix as specified in the header of the image. The CD matrix includes linear terms of the distortion model: skew, rotation, and scaling. Non-linear terms, defined as polynomial functions *f(i,j)* and *g(i,j)*, are applied using the following transformation:

$$\begin{bmatrix} \beta \\ \Upsilon \end{bmatrix} = \begin{bmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_2 \end{bmatrix} \begin{bmatrix} i + f(i,j) \\ j + g(i,j)) \end{bmatrix}$$

where,

- β and Υ are intermediate world coordinates represented in degrees with its origin at (`CRVAL1`,`CRVAL2`) on the sky

- CD1_1, CD1_2, CD2_1, CD2_2 are CD matrix keyword values describing linear distortions applied to plate scale, rotation, and skew of the image
- *f* and *g* are simple image polynomial (SIP) functions
- *i* and *j* are image pixels relative to a chosen origin (CRPIX1,CRPIX2) on the detector

Coefficients for the non-linear polynomial terms $i^p j^q$ are recorded in the image header using the keyword naming convention A_*p*_*q* and B_*p*_*q*. The polynomial functions for use in the transformation can, therefore, be expressed as:

$$f(i,j) = \sum_{p,q} A\_p\_q i^p j^q \quad \text{where} \quad (p+q) \le A\_\text{ORDER}$$

$$g(i,j) = \sum_{p,q} B\_p\_q i^p j^q \quad \text{where} \quad (p+q) \le B\_\text{ORDER}$$

where *A_ORDER* and *B_ORDER* are keywords reporting the order of the polynomial used for the model.

In the astrometric expression shown above, all linear components–scale, rotation, and skew–are incorporated into the CD matrix. Therefore, the simple image polynomials (SIP) have no terms lower than the quadratic; this is a change from the previous representation in HST data where linear terms were included in the distortion polynomial. In other words, simple image polynomials (SIP) represent deviations from a standard tangent plane described by CRVAL1, CRVAL2, and the CD matrix.

3.2.4 Optional Non-Polynomial Distortions

Image extensions containing non-polynomial distortion corrections are created in the pipeline by AstroDrizzle, and are included in the data that's delivered from the HST Archive to users. These tables were created using information extracted from the NPOLFILE (Non-POLynomial) and D2IMFILE (Detector to IMage) calibration reference files.

At present, WFC3 data does not require non-polynomial distortion corrections. The CD matrix and SIP provide distortion corrections that are good to 0.1 pixels (with a root mean square error significantly smaller than that value).

In some cases, however, distortion corrections using SIP and the CD matrix can be improved with additional non-polynomial distortion corrections. For example, a low order (4th order) SIP distortion solution for ACS/WFC is only good to a couple of tenths of a pixel and requires additional distortion terms for a more accurate geometric distortion correction. For ACS detectors, Anderson, et al. (Calibration Workshop 2002 and ACS ISR 2004-15), represented these as non-polynomial corrections in the form of look-up tables.

For some instruments, there are small offsets in the physical positions of columns or rows in chips. ACS WFC CCD's have column offsets due to the placement of the lithograph used in creating the chips. (There's no offsets in the other ACS detectors

and in WFC3. For information about other instruments, please refer to the *MultiDrizzle Handbook*.)

For MultiDrizzle, non-polynomial corrections (residual distortion corrections and detector column or row offset corrections) were represented by DGEOFILE (differential geometry) reference images: each chip had one full-frame image representing higher order offsets in *x*, and another full-frame image representing higher order offsets in *y*. These non-polynomial corrections were applied to ACS's WFC, HRC, and SBC data.

Instead of using DGEOFILE reference images, AstroDrizzle returns to the look-up table approach, implementing distortion corrections in the following stages:

1. The chip position is corrected for detector errors (currently only applied to ACS/WFC images in the pipeline). Those corrections are represented here as *DET2IM*[3]

$$(x', y') = DET2IM(x, y)$$

2. The position of the coordinates are determined relative to the fiducial point of the field (CRPIX1, CRPIX2):

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} x' - CRPIX1 \\ y' - CRPIX2 \end{bmatrix}$$

3. These relative positions are then used in the SIP equation, along with another look-up table, *LT*, containing corrections for optical distortions that cannot be easily represented by a polynomial[4]. With these changes the basic SIP equation becomes:

$$\begin{bmatrix} \beta \\ \Upsilon \end{bmatrix} = \begin{bmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_2 \end{bmatrix} \begin{bmatrix} u' + f(u', v') + LT_x(x', y') \\ v' + g(u', v') + LT_y(x', y') \end{bmatrix}$$

All look-up tables used in AstroDrizzle follow the proposed Paper IV FITS convention. This enables the storage of look-up tables in image extensions needed for computing tabular interpolations.

Note:

The definition of the look-up tables used here has the potential to create a small offset between the position stated in the header keywords CRVAL1, CRVAL2 and the actual sky positions at CRPIX1, CRPIX2. For ACS WFC, this offset is of order one

3. The information is obtained, during pipeline processing, from the D2IMFILE (detector-to-image look-up) reference file, and stored as a FITS extension in the image file.

4. These non-polynomial corrections were obtained from the NPOLFILE (Non-POLynomial distortion correction) reference file during pipeline processing, and stored in the image as FITS extensions.

to two hundredths of a pixel. Finally, it's worth noting that the SIP convention makes the reference pixel CRPIX1, CRPIX2 a special location on the chip; it is the reference position of the distortion solution. If one defines the look-up tables to have zero shift at that same reference position, then this offset disappears. Unfortunately, it cannot be implemented for the ACS without producing a small relative offset of the WFC chips.

The NPOLFILE Reference File: the Newer, Smaller DGEOFILE

MultiDrizzle uses the DGEOFILE reference files for applying non-polynomial distortion corrections. There are two full-chip size image extensions in the DGEOFILE reference file for each instrument detector chip; one extension contains residuals of the polynomial fit in *x*, and the other contains residuals of the polynomial fit in *y*. For example, the DGEOFILE reference file for ACS/WFC has four extensions, each of them with dimensions 4096 x 2048 pixels: extension 1 contains residuals of the polynomial fit in *x* chip 2; extension 2 has residuals of the polynomial fit in *y* for chip 2; extension 3 contains residuals of the polynomial fit in *x* for chip 1; extension 4 contains residuals of the polynomial fit in *y* for chip1.

As outlined in Anderson, 2002, residuals of the polynomial fit were originally modeled by sampling every 64 pixels in the WFC, creating a 65 × 33 table for each chip. For the HRC, it was every 16 pixels sampled and modeled to create a 65 × 65 table. The value of a residual distortion at any point on the chip is determined by interpolation. These tables were used to create a full-chip distortion "image," the DGEOFILE reference image, for each camera.

Because of the coordinate transformations, and the many steps involved in creating DGEOFILE files, it was not possible to start with the original tables. Therefore, a new version of the distortion tables were created for use by AstroDrizzle, done by subtracting the column width corrections from the DGEOFILE reference file, then sampling each full chip size DGEOFILE reference file extension to re-create the look-up tables. Tests have showed a near one-to-one match with the original images.

A Note About Distortion Corrections in NPOLFILE Reference Files

In MultiDrizzle, non-polynomial distortion corrections were applied after implementing distortion corrections described by polynomial functions (that have coefficient values stored in the IDCTAB reference file). As a result, non-polynomial distortion corrections in the DGEOFILE reference file also included linear distortion corrections.

For AstroDrizzle, in accord with the SIP convention, those linear distortion corrections applied to scale, rotation, and skew are now incorporated in the CD matrix; this portion of the transformation from distortion-corrected coordinates to sky coordinates is performed in the final step. Therefore, residual distortion tables in the NPOLFILE reference file, that were extracted from the DGEOFILE reference file as described earlier, must be corrected with the inverse of the CD matrix so that linear distortion corrections are not applied twice. This is done in the pipeline when the **stwcs.updatewcs** task performs the correction for each extension of the NPOLFILE reference file (that contains the tabular form of the residual distortions).

Interpretation of NPOLFILE reference files

Look-up tables containing residual distortion corrections for an image are derived from the NPOLFILE reference files and saved as separate FITS image extensions of type WCSDVARR in the science image.

In a later processing step, software that performs coordinate transformations will use bilinear interpolation to calculate the value of the distortion at each pixel location in the image array.

To fully map the image array to the distortion array, the standard WCS header keywords CRPIX*j*, CRVAL*j* and CDELT*j* are used, where *j* denotes the axis. The mapping follows the transformation

$$p_j\_im = CDELT_j\_dist \times (p_j\_dist - CRPIX_j\_dist) + CRVAL_j\_dist$$

where

- *im* refers to the image array
- *dist* is the distortion array
- *p* is the pixel

In this mapping of the two arrays,

- $CRPIX_j\_dist$ is chosen as $NAXIS_j\_dist/2$ (axis dimension in pixels for each chip)
- $CRVAL_j\_dist$ corresponds to positions at $CRPIX_j\_im$
- $CDELT_j\_dist$ is the spacing interval between pixels in the image array that map to consecutive pixels in the distortion array

In general $\texttt{CDELT}_j\_dist$ can have a non-integer value, but cannot be zero. However, if the distortion array is obtained as a sub-image of a larger array, having a non-integer step size can produce undesirable results during interpolation.

A Note About Look-up Tables in the WCSDVARR Extension

It is essential that the look-up tables in the image extensions meet two restrictions:

- Every point in the corrected image is mapped to only one point in the uncorrected image (injective)
- Every point in the corrected image corresponds to at least one point on the uncorrected image (surjective)

In other words, a look-up table acts as a function to describe a one-to-one relationship between each point in the distortion-corrected image with a unique counterpart point in the uncorrected image, such that both images are invertible by their look-up table.

This one-to-one (non-extrapolation) requirement can have implications on the geometry of the distortion array. If the distortion array is obtained as a subimage or subsample of a larger array, it is important that the edges of the distortion array coincide with the edges of the image.

The D2IMFILE Reference File

The other element of the non-polynomial distortion correction is the fixed column width correction (for ACS/WFC). It's applied at the very start of the distortion correction process so that column width-corrected coordinates are used as input to the polynomial and non-polynomial distortion corrections. The adopted implementation is based on the FITS Paper IV look-up table convention.
This transformation can be thought of, conceptually, as:

$$(u, v) = DET2IM(x, y)$$

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u - CRPIX1 \\ v - CRPIX2 \end{pmatrix}$$

where,
u,v are the initial coordinates x,y with the column correction applied (the *DET2IM* calibration step). At this time, the correction is only applied to ACS/WFC data. For detectors that do not need this correction, $u=x$ and $v=y$.
u',v' are the *DET2IM*-corrected coordinates relative to `CRPIX1`, `CRPIX2`.

For ACS/WFC, a new reference file, given by the header keyword `D2IMFILE`, was created. It contains a one-dimensional array of column width corrections. In the pipeline, this correction table gets attached to the image as a FITS extension of type `D2IMARR`. Each element in the `D2IMARR` array specifies the correction (in units of pixels) for every pixel in the column of both science image extensions.

This kind of detector-to-image correction could be extended to an arbitrary number of chips and extensions for other instruments and detectors. The extension headers would contain keywords about the implementation of the corrections. The keyword `AXISCORR`, for instance, indicates if the distortion corrections are applied to columns or rows. For ACS/WFC, the `CDELT1` keyword was set to "1" which indicated that no distortion correction interpolations were done.

3.3 Distortion Information in Pipeline Calibrated Images

3.3.1 New Image Structure

ACS and WFC3 images retrieved from the HST Archive are processed via the 'On-The-Fly-Reprocessing' (OTFR) system that calibrates images using the best-available calibration file, including latest-available distortion correction models.

Calibrated science images from the HST Archive, with the suffix `flt.fits`, have undergone standard image reductions including flat fielding, but have not been corrected for geometric distortion. After the data calibration step in the pipeline, `flt.fits` images are processed by AstroDrizzle to produce drizzled images that are corrected for geometric distortion. If several images are part of an association, the

images are drizzle-combined using pre-defined **astrodrizzle** settings stored in the MDRIZTAB reference file.

As described in the previous section, WCS information in flt.fits images are updated by AstroDrizzle in the pipeline to include the full distortion correction model:

- scale, rotation, and time-dependent skew linear corrections are incorporated in the CD matrix.
- Distortion correction polynomial function orders and coefficients from the IDCTAB are recorded as keyword values in the image header.
- Optionally, any residual distortion corrections and detector corrections from the NPOLFILE and D2IMFILE reference files are appended to the science image as new FITS extensions of type WCSDVARR and D2IMARR, respectively.

Note: even though the IDCTAB and DGEOFILE reference files are no longer needed for post-pipeline reprocessing, these keywords have been retained in the new flt.fits files for users who prefer to process their images using MultiDrizzle. The new flt.fits files have been verified to be compatible with MultiDrizzle.

The figure below shows the new structure of an ACS/WFC image, generated by the IRAF command **catfits**:

```
--> catfits jb7310ymq_flt.fits<br>
 EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT


 0     jb7310ymq_flt jb7310ymq_flt.fits                       16
 1       IMAGE       SCI                   1     4096x2048    -32
 2       IMAGE       ERR                   1     4096x2048    -32
 3       IMAGE       DQ                    1     4096x2048    16
 4       IMAGE       SCI                   2     4096x2048    -32
 5       IMAGE       ERR                   2     4096x2048    -32
 6       IMAGE       DQ                    2     4096x2048    16
 7       IMAGE       D2IMARR               1     4096         -32
 8       IMAGE       WCSDVARR              1     65x33        -32
 9       IMAGE       WCSDVARR              2     65x33        -32
 10      IMAGE       WCSDVARR              3     65x33        -32
 11      IMAGE       WCSDVARR              4     65x33        -32
```

3.3.2 New Keywords

Naming a Polynomial Distortion Solution

Polynomial distortion models for all HST images, which are used as the primary source of distortion information, are presented in the image headers using the SIP convention.

Some instrument modes, however, require additional distortion corrections. During pipeline processing, distortion correction information is obtained from reference files and stored in images as FITS extensions. For data processed in the OTFR pipeline, only ACS images require residual optical distortion corrections, stored in the image files as FITS extensions of type `WCSDVAR`. For ACS/WFC a correction for detector column width is also required, stored in image files as a FITS extension of type `D2IMARR`.

The name for a specific polynomial distortion model is recorded in the primary header of each image in a new keyword called `SIPNAME`. In the pipeline, the default distortion model is named after the image's rootname and its `IDCTAB` reference file. A name can also be assigned to this keyword by the user during post-pipeline processing. A value of 'N/A' or a blank string indicates that no SIP model was provided or applied. A value of *UNKNOWN* means that there's a SIP model but no record of the model's origin.

A unique description of the full distortion model is summarized in a new keyword called `DISTNAME`, in the primary header of the image file. The value for this keyword is a string comprised of the names of all distortion model components used for the image, described by the keywords `SIPNAME`, `NPOLFILE`, and `D2IMFILE`. A value of *UNKNOWN* is used if a distortion model was applied but the keyword values for `SIPNAME`, `NPOLFILE`, and `D2IMFILE` are not provided.

In the example below, **hedit** is used to query the `SIPNAME` and `DISTNAME` for an ACS/WFC image. The `NPOLFILE` and `D2IMFILE` keyword values are also obtained to show the naming nomenclature for the `DISTNAME` keyword value:

```
--> hedit j9cd01kpq_flt[0] SIPNAME,DISTNAME,NPOLFILE,D2IMFILE .
j9cd01kpq_flt[0],SIPNAME = j9cd01kpq_u7n18502j
j9cd01kpq_flt[0],DISTNAME = j9cd01kpq_u7n18502j-v971826gj-v971826mj
j9cd01kpq_flt[0],NPOLFILE = jref$v971826gj_npl.fits
j9cd01kpq_flt[0],D2IMFILE = jref$v971826mj_d2i.fits
```

Even though there are several extensions appended to this FITS file, the sum total of them does not exceed 100 kB (ACS/WFC images require 97,200 bytes), making this a space-efficient way to manage all distortion and WCS information.

This effort to include distortion information in science images requires the use of multiple FITS conventions (proposed and improved) to support the full range of calibration distortion models used for HST data. The approved SIP Convention is used for describing polynomial terms of the distortion correction, while the proposed Paper IV FITS convention is used for `NPOLFILE` and `D2IMFILE` distortion corrections.

SIP Convention Keywords

IDCTAB reference files are used, in the pipeline, to populate science header keyword values that describe polynomial models as coefficients. A prime example is the implementation of SIP (a registered FITS convention) in the STScI_Python package **stwcs** (which relies on the FITS standard C package **wcstools**). The reliance on published or proposed FITS standards allows these updated HST headers to be understood by other standard astronomy-related tools such as the image display software ***ds9***.

The keywords used for the SIP standard are shown in the table below.

Table 3.1: Standard SIP Keywords Keyword

KEYWORD	DEFINITION
CTYPE1, CTYPE2	The type of tangent plane projection used in the creation of the distortion.
CD1_1, CD1_2, CD2_1, CD2_2	Linear terms of distortion: scale, rotation, and skew. These are CD matrix keywords
A_ORDER, B_ORDER	Linear terms of distortion: scale, rotation, and skew. A_ORDER, B_ORDER Order for distortion polynomials, along axis 1 and axis 2 respectively.
A_*p*_*q*, B_*p*_*q*	High order coefficients for X-axis and Y-axis, respectively. For A_*p*_*q*, $(p + q) \leq$ A_ORDER ; for B_*p*_*q*, $(p + q) \leq$ B_ORDER
A_DMAX, B_DMAX	Maximum correction, in units of pixels, along axis 1 and axis 2, respectively. [Optional keywords]
SIPREF*i*	Origin of distortion model along axis *i*. [Optional keywords]
SIPSCL*i*	Scale term for axis *i* [Optional keywords]

Excerpts from an ACS/WFC image header illustrate how distortion correction-related keywords, including standard SIP keywords, are presented:

```
World Coordinate System and Related Parameters
WCSAXES =                    2 / number of World Coordinate System axes
CRPIX1  =                512.0 / x-coordinate of reference pixel
CRPIX2  =                512.0 / y-coordinate of reference pixel
CRVAL1  =       343.7452251651 / first axis value at reference pixel
CRVAL2  =       -13.7827815967 / second axis value at reference pixel
CTYPE1  = 'RA---TAN-SIP'       / the coordinate type for the first axis
CTYPE2  = 'DEC--TAN-SIP'       / the coordinate type for the second axis
CD1_1   = 8.83065918237360E-07 / partial of first axis coordinate w.r.t. x
CD1_2   = 6.89997778057832E-06 / partial of first axis coordinate w.r.t. y
CD2_1   = 7.84941986149176E-06 / partial of second axis coordinate w.r.t. x
CD2_2   = -7.6694270297401E-08 / partial of second axis coordinate w.r.t. y
LTV1    =        0.0000000E+00 / offset in X to subsection start
LTV2    =        0.0000000E+00 / offset in Y to subsection start
VAFACTOR=   1.000066938799E+00 / velocity aberration plate scale factor
WCSNAME = 'IDC_q692007bj'
SIPNAME = 'j94f05bgq_q692007bj'
```

Additional information:

- LTV1, LTV2, *x*, and *y* position, respectively, of the first science pixel in the science array
- WCSNAME and SIPNAME, while not standard SIP keywords, are new keywords to support the use of the SIP convention, with WCSNAME following FITS WCS Paper I standards.

```
New SIP-related Keywords (see Table 3.1 for definitions)
A_3_1   = -1.8489059133749E-12
A_3_0   = 1.16159212298711E-10
B_3_0   = 4.45076703973379E-10
B_3_1   = -1.4260311711311E-12
B_1_2   = -6.9177461674173E-10
B_1_3   = -5.7069041987336E-13
B_1_1   = -2.6506257298218E-06
B_2_1   = 3.29559403960939E-10
B_2_0   = 1.96232468750349E-06
B_2_2   = 9.99713632234838E-13
A_4_0   = 2.46309332730701E-13
A_ORDER =                 4
B_0_4   = -1.1021179642905E-12
B_0_3   = 5.00419914431113E-10
B_0_2   = 1.14514599215996E-05
B_ORDER =                 4
B_4_0   = 3.92892145036559E-13
A_1_1   = 8.92967949471260E-06
A_1_3   = 1.26960536157938E-13
A_1_2   = 3.85428302555575E-10
A_0_4   = 3.97292355746197E-14
A_0_2   = -7.0355048953902E-07
A_0_3   = -8.7006311150326E-12
A_2_2   = -1.4442609282225E-12
A_2_0   = -3.2953191490160E-06
A_2_1    = 2.11696756823604E-11
SIPNAME = 'j94f05bgq_q692007bj'
```

```
 / Original linear distortion terms from reference file
OCX10   = -3.9113030652515E-05
OCX11   =  0.02828506007790565
OCY10   =  0.02483976073563099
OCY11   = 0.002909119008108974
IDCSCALE=              0.025
IDCTHETA=                0.0
IDCXREF =              512.0
IDCYREF =              512.0
IDCV2REF=   206.3240051269531
IDCV3REF=    472.2200012207031
```

Additional information:

- OCX10, OCX11, OCY10, OCY11 are linear distortion terms without image scale, directly from the distortion model in the IDCTAB reference file.
- IDCSCALE is the pixel scale from the IDCTAB reference file.
- IDCTHETA is the orientation of the detector's y-axis relative to the V3 axis, as derived from the IDCTAB reference file.
- IDCV2REF, IDCV3REF are the reference pixel's V2 and V3 positions, respectively, as derived from the IDCTAB reference file.
- IDCXREF, IDCYREF are the reference pixel location in x and y as specified in the IDCTAB.

```
 / Time-dependent distortion keywords for ACS (link to ISR 2007-08)
TDDALPHA=  0.1676283367556471
TDDBETA = -0.05320944558521569
```

The SIP convention retains use of the current definition of the CD matrix where linear terms of the distortion model are folded in with the orientation, scale, and time-dependent skew at the reference point for each chip to provide the best linear approximation of the distortion. The SIP convention gets interpreted by applying higher-order coefficients *A_p_q*, *B_p_q* to the input pixel positions, then by applying the CD matrix and adding the CRVAL position to get the final world coordinates.

The SIP convention was created from the original form of the FITS Paper IV standards, but the Paper IV proposal since changed to use a different set of keywords and conventions. All of the FITS WCS Papers, including the draft for Paper IV, can be accessed through the HEASARC FITS WCS webpage.

Paper IV Proposal

The current Paper IV conventions provide a mechanism for specifying either a look-up table or polynomial model for the distortion of each axis. The standard states, in Section 2.1 of the Paper IV draft (as of November 2011):

> Note that the prior distortion functions ... operate on pixel coordinates (i.e., p rather than p− r), and that the independent variables of the distortion functions are the uncorrected pixel or intermediate pixel coordinates. That is, for example, we do not allow the possibility of $q_3' = q_3 + \delta_{q3}(q_1', q_2')$

The keywords used for describing these corrections use the syntax given in Table 2 of the draft Paper IV. For our purposes, the keywords of interest are those related to look-up tables, namely,

Table 3.2: Excerpt of Table 2 from Draft Paper IV with Keywords Related to Look-up Tables

Keyword	Type	Section (in Paper IV)	Use	Comments
CPDISja	string	2.4.1	Distortion code	Prior distortion function type
DPja	record	24.2	Distortion parameters	Parameter for a prior distortion function, for use in the image head.

This syntax only provides the option to specify one correction at a time for each axis of the image. It precludes the use of this convention to specify both a look-up table and a polynomial model at the same time for the same axis. It does not state what should be done if the polynomial has been specified using a different convention, for example, the SIP convention. Therefore, SIP and Paper IV should not be seen as mutually exclusive. In fact, they will work together rather naturally since the SIP and Paper IV conventions both assume that the corrections will work on the input pixel and add to the output frame.

The only transformation that needs to be performed in this implementation is the adaptation of the `DGEOFILE` for use with the SIP coefficients, as performed by **makewcs** when the FITS header originally first gets updated in the pipeline. Corrections derived from the `DGEOFILE`, as well as corrections described by the SIP coefficients can then be added together as it was done in the original distortion calibration code. This has been documented in the paper describing the implementation of the distortion models in the draft of Paper IV.

The following header excerpt illustrates how the keywords get defined for an ACS/WFC image with an NPOLFILE reference image specified in the image header.

```
CPERR1=           0.058       / Maximum error of dgeo correction for axis 1
CPDIS1  = 'Lookup  '           / Prior distortion funcion type
DP1     = 'EXTVER: 1.0'        / Version number of WCSDVARR extension containing
DP1     = 'NAXES: 2.0'         / Number of independent variables in distortion f
DP1     = 'AXIS.1: 1.0'        / Axis number of the jth independent variable in
DP1     = 'AXIS.2: 2.0'        / Axis number of the jth independent variable in
DP1     = 'Scale1: 1.0'          / Scale factor (binning) in axis 1
DP1     = 'Scale2: 1.0'          / Scale factor (binning) in axis 2
DP1     = 'Offset1: 0.0'       / Offset in axis 1
DP1     = 'Offset2: 0.0'       / Offset in axis 2
CPERR2=                0.072   / Maximum error of dgeo correction for axis 2
CPDIS2  = 'Lookup  '           / Prior distortion function type
DP2     = 'EXTVER: 2.0'        / Version number of WCSDVARR extension containing
DP2     = 'NAXES: 2.0'         / Number of independent variables in distortion f
DP2     = 'AXIS.1: 1.0'        / Axis number of the jth independent variable in
DP2     = 'AXIS.2: 2.0'        / Axis number of the jth independent variable in
DP2     = 'Scale1: 1.0'          / Scale factor (binning) in axis 1
DP2     = 'Scale2: 1.0'          / Scale factor (binning) in axis 2
DP2     = 'Offset1: 0.0'       / Offset in axis 1
DP2     = 'Offset2: 0.0'       / Offset in axis 2
NPOLEXT = 'wfc_64_f606w_npl.fits'
```

The DP<*n*>.EXTVER keywords point to the WCSDVARR extension associated with the correction, with one extension for each dimension of the image chip. In other words, there are two WCSDVARR extensions associated with this single chip, one containing the X corrections the other one containing the Y corrections.

Detector to Image Correction

The fixed column width correction, required only for ACS/WFC is applied at the very start of the distortion correction process. Its applied to flt.fits pixel positions, and the output positions are then used for computing the polynomial and other non-polynomial distortion corrections. The adopted implementation for describing this detector-to-image correction in the header, and how to apply it to the coordinates, is based on the Paper IV Look-up Table convention. It's assumed that the detector to image correction is the same for all chips, so only one look-up table needs to be specified and appended as a new FITS extension.

For ACS/WFC, the correction is a one-dimensional image extension of type D2IMARR. Each element in the row represents the correction for every pixel in the column of the science extension.

The following new keywords for this correction have been added to the science image's header.

Table 3.3: Standard Detector Distortion Correction Keywords

KEYWORD	DEFINITION
D2IMFILE	Name of reference file to be used for creating the look-up table. The task updatewcs creates an image extension of type D2IMARR, and populates it with column distortion information from this reference file.
AXISCORR	Axis to which the *DET2IM* correction is applied. (1 - 'X' Axis, 2- 'Y' axis)
D2IMEXT	Name of reference file which was last used to create the look-up table. 'D2IMEXT' stores the name of the reference file used by updatewcs to create a D2IMARR extension. If 'D2IMEXT' is present in the 'SCI' extension header and is different from the current value of D2IMFILe in the primary header, the correction array in D2IMARR is updated.
D2IMERR	Maximum value of the correction (optional)

For example:

```
 --> hedit j9cd01kpq_flt.fits[sci,2] d2imfile,axiscorr,d2imext,d2imerr .
j9cd01kpq_flt.fits[sci,2],D2IMFILE = jref$v971826mj_d2i.fits
j9cd01kpq_flt.fits[sci,2],AXISCORR = 1
j9cd01kpq_flt.fits[sci,2],D2IMEXT =  jref$v971826mj_d2i.fits
j9cd01kpq_flt.fits[sci,2],D2IMERR = 0.002770500956103206
```

These new keywords are added to the science image primary header when the keyword D2IMFILE has been assigned a reference file. Information in this reference file is used to create a look-up table containing column width corrections for both chips, stored in the D2IMARR image extension.

3.3.3 Final Products

AstroDrizzle's primary product is a multi-extension FITS file with the suffix drz.fits.

The first extension contains the science (SCI) image which is corrected for distortion and, if applicable, dither-combined or mosaiced. The drizzled SCI image extension is typically in units of electrons per second which is the default for ACS and WFC3 images. (A user can choose to have the output in either electrons or electrons per second.) All image pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image that is both photometrically and astrometricly accurate for point sources and extended sources. The dimensions of

the output image are computed automatically by AstroDrizzle, and the default output plate scale value is given by the header keyword `IDCSCALE` (its value obtained from the `IDCTAB` reference file during pipeline processing). These **astrodrizzle** parameters and others may be changed by the user during post-pipeline reprocessing to best suit the actual data and user's needs–this will be covered in a later section of this handbook.

The WCS information for the science image of the drizzle product no longer contains any keywords related to distortion, as those models were removed from each input prior to being included in this combined image. Only the basic sky transformation gets written out to the drizzle science image header to reflect the uniform pixel scale and orientation of the image. The resulting header only contains the basic `CD*`, `CRVAL`, `CRPIX`, and `CTYPE` keywords to describe the transformation from each pixel position to a sky position, with a basic RA-TAN/DEC-TAN projection being provided in the header. This WCS solution also gets a default label, as the `WCSNAME` keyword, of `DRZWCS` to reflect the fact that it no longer contains any distortion model from any of the input images. Future versions of this code may allow each user to define the value to be written out as the `WCSNAME` keyword to be anything they want, but for now, this single default value will be provided in all AstroDrizzle products, especially those generated during pipeline processing.

The secondary outputs, available as image extension types `WHT` and `CTX`, are the the weight and context images, respectively. The weight image gives the relative weight of the output pixels, and can be considered an effective exposure time map. The context extension contains a map of the output images and a record of which image contributed to each pixel.

The WFC3/IR pipeline is based on the NICMOS pipeline, and produces a calibrated `flt.fits` file using the same up-the-ramp fitting procedure which rejects cosmic rays. The units of the WFC3/IR `flt.fits` file are electrons per second, rather than data numbers per second, to be more consistent with the calibrated WFC3/UVIS flt.fits files, which are in units of electrons.

ACS and WFC3 users should note that calibrated `flt.fits` files from the pipeline contain additional bits in the DQ mask that identify pixels flagged as cosmic rays by AstroDrizzle processing in the pipeline. The default value for these bits is 4096, and if AstroDrizzle is re-run off-line on those `flt.fits` files, it will be able to use these flags and proceed directly to the final drizzle step without the need to re-do cosmic ray masking, unless required; however, users should first verify that the images are properly aligned before proceeding to the final AstroDrizzle step. Alternatively, if the cosmic ray rejection needs to be improved, then cosmic rays identified in the pipeline can be ignored by setting the ***resetbits*** parameter to 4096 when re-running **astrodrizzle**.

3.4 The Headerlet

3.4.1 Introduction

A headerlet is a compact FITS file representing a single WCS solution for a single exposure that has been aligned to a particular image or astrometric catalog, complete with all distortion information. It could be utilized as a convenient WCS information package that a user could e-mail to a collaborator working on the same image, so it can be incorporated into the collaborator's copy of the image. This avoids the need to send that entire image containing the new WCS to the collaborator.

Headerlets serve two primary purposes:

1. They contain a summary of all WCS information for a single exposure.
2. They can be used to update the WCS information in additional copies of the same image.

The default suffix for headerlet files, as used by all DrizzlePac tasks, is `hlet.fits`. For example, the name of a headerlet written out by **tweakreg** for the WFC3 image `iabf01ckq_flt.fits` for the WCS solution named `TWEAK` would be named `iabf01ckq_tweak_hlet.fits`.

Why Headerlets are Useful

A lot of effort can be expended in determining the alignment between an image and an astrometric field or another image. It requires some expertise with source-finding algorithms, computing offset fit solutions, and a knowledge of FITS WCS standards. That expertise can now be embodied in headerlets created using the **tweakreg** task, which was written to simplify image alignment and provide updated WCS information for aligned images. This information, stored in the headerlets, can then be easily applied to the same images being studied by other users, saving them the work needed to align the images to the same reference image or catalog.

Types of Information Contained in Headerlets

AstroDrizzle requires that all WCS and distortion information be present in the input images in order to align them. This can include multiple components, some that are time-dependent, and others that are exposure- or orbit-dependent. The **updatewcs** task, used by AstroDrizzle in the pipeline (and also available to users), reads in base models from the reference files; it computes components such as time- and orbit-dependent WCS information that are unique to an exposure, and populates the relevant WCS keywords in the image header. For example, ACS and WFC3 data have a velocity aberration keyword which reports the specific aberration for that exposure. That information gets applied to the distortion model as an additional plate scale correction. As a result, the distortion solution for one image will never be applicable to any other exposure, even if they are taken back-to-back.

When aligning a set of images from different visits, image WCS offsets between visits due to guide star coordinate errors need to be removed. This involves revising

the image WCSs such that they align with one image from the set that acts as the reference image. Another option is to align the image WCSs to an external astrometric catalog. Therefore, updating the WCS, as done by **tweakreg**, accounts not only for the full distortion model (as mentioned in the previous paragraph), it can also correct WCS offsets due to different guide stars.

3.4.2 WCS Information in Archival Images

For active instruments[5] (specifically, ACS and WFC3), when a user submits a data request to the HST Archive, images are processed with the latest image calibrations, including the latest available distortion models, by the On-The-Fly Reprocessing (OTFR) system before being delivered to the user. Calibrated image files, that have the suffix `flt.fits`, (or `flc.fits` for CTE-corrected ACS images), serve as inputs to AstroDrizzle where distortions are corrected and the images are drizzle-combined into a single drizzled image (with suffix `drz.fits`, or `drc.fits` from `flc.fits` input files).

WCS information in `flt.fits` images gets updated when AstroDrizzle is run in the pipeline. The full distortion model, including the full polynomial solution from the `IDCTAB` reference file, and all the corrections formerly combined into the `DGEOFILE` reference image, are now stored directly in the images in the form of keyword values and FITS extensions. The FITS Conventions report by N. Dencheva, et al. contains a full description of the conventions for coordinate transformations used to describe all these components in a FITS file.

As described in Section 3.3.2, image headers now contain the following keywords and image extensions to fully describe the WCS with distortion corrections.

- Linear WCS (including CD matrix) keywords: `CRPIX*`, `CRVAL*`, CD*, and `CTYPE*`
- SIP coefficients keywords: `A_`*p*`_`*q* and `B_`*p*`_`*q*[6], `A_ORDER`, `B_ORDER`, `OCX10`, `OCX11`, `OCY10`, and `OCY11`
- [7]Residual distortion corrections reference file: if the `NPOLFILE` header keyword specifies the name of a residual distortions reference file for an image, `CPDIS*` and `DP*` keywords point to image extensions of type `WCSDVARR` (Paper IV convention)

5. Retired instruments are available in Static Archives. Please refer to the MAST website for additional information. WFPC2 and NICMOS data can be reprocessed using DrizzlePac tasks.

6. For A_*p*_*q*, $(p+q) \leq$ `A_ORDER`; for B_*p*_*q*,, $(p+q) \leq$ `B_ORDER`

7. Residual or non-polynomial optical distortion corrections, contained in the `NPOLFILE` reference file and stored as image extensions of type `WCSDVARR`, are currently only implemented for ACS images. The column width corrections, from the `D2IMFILE` reference file and stored in image extension type `D2IMARR`, are only used for ACS/WFC images. Please check the DrizzlePac website for possible updates for other instruments. Note that WFPC2 images, currently in the static Archive, also requires the row width correction, but this is implemented directly by AstroDrizzle.

  - [7]WCSDVARR extensions: two image extensions per chip are created, each with look-up tables containing the corrections from the NPOLFILE (non-polynomial) reference file, where each extension corresponds to an image axis (*X* correction or *Y* correction)

- [7]Column correction reference file: if the D2IMFILE header keyword specifies a reference file for column width corrections, the D2IMEXT and D2IMERR keywords point to a file extension of type D2IMVARR.
  - [7]D2IMVARR extension: an extension with a look-up table containing chip column corrections from the D2IMFILE reference file.

Each science header will have its own set of these keywords, as well as extensions that will be kept together as part of the full WCS solution.

Only one set of distortion coefficients and look-up tables, stored in the SIP header keywords and from the NPOLFILE reference file, respectively, can be included in the science header. This single distortion model is used by all alternate WCS's in the image header that are stored as sets of WCS keywords using the FITS Paper I Multiple WCS Standard.

3.4.3 Storing Multiple WCSs in an image

When making updates to any WCS solution, the software will, by default, save previous WCS solutions in case it is needed for later use. This results in multiple alternate WCS solutions, based on the same distortion model, being present in the image header at one time. Therefore, each image has the potential to align to several different astrometric solutions, depending on which alternate WCS solution the user wants to select. One potential reason for having multiple headerlets in an image would be to provide WCSs that are aligned to different astrometric catalogs. That would allow a user to pick a WCS that matches a catalog that he or she wishes to use. These layers of multiple WCS solutions can be somewhat complex. Code in the STWCS package, used by DrizzlePac tasks, was designed to simplify the management of these different WCSs, making them more easily accessible via headerlets and other **drizzlepac** task parameters.

Table 3.4: Extensions in the Typical ACS/WFC Exposure after Running tweakreg

EXT #	FITS NAME	FILE NAME	EXTVE	DIMENSION	BITPIX
0	j8hw27c4q_flt	j8hw27c4q_flt.fits			16
1	IMAGE	SCI	1	4096x2048	-32
2	IMAGE	ERR	1	4096x2048	-32
3	IMAGE	DQ	1	4096x2048	16
4	IMAGE	SCI	2	4096x2048	-32
5	IMAGE	ERR	2	4096x2048	-32
6	IMAGE	DQ	2	4096x2048	16
7	IMAGE	D2IMARR	1	4096	-32
8	IMAGE	WCSDVARR	1	64x32	-32
9	IMAGE	WCSDVARR	2	64x32	-32
10	IMAGE	WCSDVARR	3	64x32	-32
11	IMAGE	WCSDVARR	4	64x32	-32
12	BINTABLE	WCSCORR		18Fx10R	

Even though there are a large number of extensions appended to this FITS file, the sum total of all of these new extensions comes to approximately 100kB for ACS/WFC images (the example in Table 3.4, for instance, is just 86,400 bytes), making them a space efficient means of managing all of the distortion and WCS information.

WCSCORR Image Extension for Storing Multiple WCSs

Table 3.4 shows an entry that has yet to be mentioned: `WCSCORR`. This is a FITS extension in the form of a binary table that was first created when running **tweakreg** with ***updatehdr=True*** and updated every time **tweakreg** is run with ***updatehdr=True*** to keep a record of all updates made to WCS keywords. Each row of the `WCSCORR` table corresponds to a single update to a single chip's set of WCS keywords. Each set of keywords defines what needs to be included in a headerlet when it gets written as a WCS solution for a single exposure.

The columns of the table include, at minimum, these keywords associated with a specific WCS:

WCS_ID	Unique descriptive ID for this particular solution
EXTVER	Chip to which this solution applies
WCS_key	FITS Paper I Alternate WCS ID
HDRNAME	Label for headerlet solution
SIPNAME	Name of polynomial distortion solution
NPOLNAME	Name of non-polynomial distortion
D2IMNAME	Name of detector-to-image distortion
CRVAL1	New value of CRVAL1
CRVAL2	New value of CRVAL2
CRPIX1	New value of CRPIX1
CRPIX2	New value of CRPIX2
CD1_1	New value of CD1_1
CD1_2	New value of CD1_2
CD2_1	New value of CD2_1
CD2_2	New value of CD2_2
CTYPE1	Name of coordinate for axis 1
CTYPE2	Name of coordinate for axis 2
ORIENTAT	New value of ORIENTAT (derived from new CD)
PA_V3	New value of PA_V3 (required for updatewcs)
RMS_RA	RMS in RA for fit (arcseconds)
RMS_Dec	RMS in Dec for fit (arcseconds)
NMatch	Number of sources used in fit
Catalog	Name of astrometric catalog used for fit
Descrip	Description of fit or sources used in fit

Syntax for Alternate WCSs

Details of how alternate WCSs get defined and used in FITS headers can be found online in FITS WCS Paper I. This capability is crucial to the success of the DrizzlePac software, as this standard allows multiple alignment solutions for an image to be specified in the image header at one time.

The default WCS solution is referred to as the *primary WCS* in documentation for DrizzlePac and STWCS tasks as it serves as the primary or default WCS transformation that is applied to the pixel positions.

Each alternate WCS solution is identified by a single letter, referred to in the standard and this documentation as a *wcskey*, with values from `A` to `Z`. The primary WCS, however, is assigned a wcskey of " " (a string comprised of a single blank). All keywords that make up each alternate WCS solution has a wcskey value appended to the end of the keyword name. For example, the alternate WCS keywords with the wcskey of "`A`" would have keywords `CRVAL1A`, `CRVAL2A`, `CRPIX1A`, `CRPIX2A`, and so on.

In addition, each WCS solution, including the primary WCS, gets labeled using the `WCSNAME*` keyword with the WCS's key appended to the end of the keyword. For example, the alternate WCS associated with the wcskey of "`A`" would have the keyword named `WCSNAMEA`.

The wcskey of "`O`" has been reserved by DrizzlePac and STWCS packages for use in archiving the original WCS solution generated in the Archive. It will always be `WCSNAMEO="OPUS"`. The software makes it nearly impossible to overwrite it since `WCSNAMEO` provides a means of recovering the original WCS solution in case updates to the images's WCS keywords introduces errors that can only be corrected by starting from scratch. The **updatewcs** task uses the alternate "`O`" WCS to recompute the original WCS solution assuming the distortion reference files can still be accessed, as it assumes that none of the WCS or distortion information in the headers are accurate.

3.4.4 Headerlet Structure

Headerlets are generated using the headerlet tasks in the **stwcs.wcsutil.headerlet** module (more about this later in the section) or by running the **tweakreg** task with ***updatehdr=True***. (Headerlets may also be created by other FITS standard software not covered in this document, as long as they follow established headeret definitions.)

When appended to a science FITS image, the headerlet is attached as a FITS extension with a non-standard extension type. This non-standard extension type will not cause problems with FITS readers, however, when using PyFITS, it will return the headerlet's full FITS format with access to all the headerlet's extensions[8] when the STWCS package has been installed. For details on how to access a headerlet's information when it is attached to an image, see "Accessing the Headerlet Extension using PyFITS" on page 59.

On the other hand, the STWCS package used by DrizzlePac tasks can use a headerlet to define a WCS for an image to perform coordinate transformations (complete with distortion correction) without even requiring the original science image and all its data.

Headerlet Format

A headerlet is a multi-extension FITS file containing a single full WCS solution for a single exposure. The full set of extensions in a headerlet is determined by the distortion model used in the image, as well as the number of chips read out for each image. A PyFITS listing of extensions (see "Accessing the Headerlet Extension using

8. FITS-compliant files can be a multi-extension FITS file, or it could be an extension to a multi-extension FITS file.

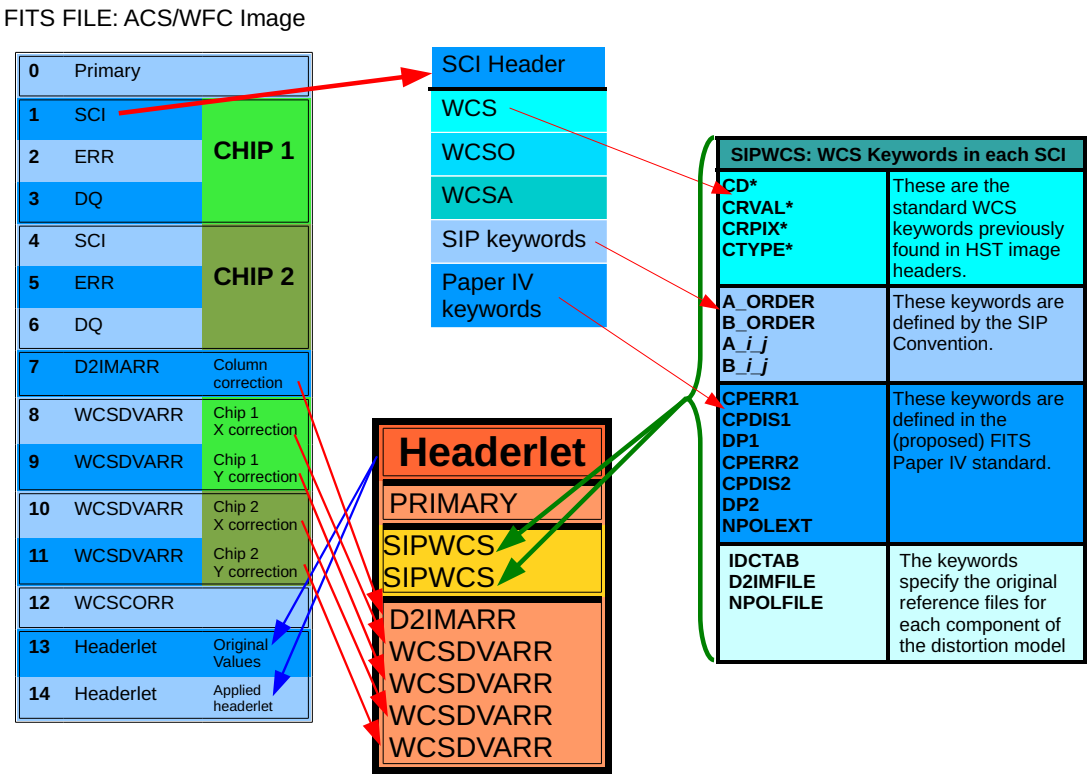
PyFITS" on page 59) in a headerlet, created from an ACS/WFC image, is shown in Table 3.5. ACS images rely on non-polynomial (NPOL) and detector column width (D2IM) corrections in the form of look-up table extensions; these get included as extensions to the headerlet in order to maintain a complete set of distortion model information. The full set of keywords included in the headerlet and from where they are derived can be seen in Figure 3.1

Table 3.5: Headerlet FITS Extensions for an ACS/WFC Headerlet

EXT #	FITS NAME	FILE NAME	EXTVE	DIMENSION	BITPI
0	j8hw27c4q	j8hw27c4q_hdr.fits			16
1	IMAGE	SIPWCS	1		8
2	IMAGE	SIPWCS	2		8
3	IMAGE	WCSDVARR	1	65x33	-32
4	IMAGE	WCSDVARR	2	65x33	16
5	IMAGE	WCSDVARR	3	65x33	-32
6	IMAGE	WCSDVARR	4	65x33	-32
7	IMAGE	D2IMARR	1	4096	-32

Note that a headerlet derived from a full-frame WFC3/UVIS image would only contain a `PRIMARY` header and two `SIPWCS` extensions (one for each SCI extension) as WFC3/UVIS does not currently use `NPOLFILE` or `D2IMFILE` reference files as part of their distortion model.

Figure 3.1: Relationship Between an ACS/WFC Image's FITS File, Science Array Header, and Headerlet



This figure shows the keywords that are included in a headerlet, the extensions included in a headerlet, and how a headerlet appears as a new extension when it gets appended to the original ACS/WFC file.

Headerlet's PRIMARY Header

The `PRIMARY` header of a headerlet will only contain those keywords necessary for identifying the science exposure name, distortion model information, and title for the WCS solution. The primary header must have four required keywords:

- `HDRNAME`, a unique name for the headerlet
- `DESTIM`, target image filename (the `ROOTNAME` keyword of the original archive filename)
- `STWCSVER`, version of **stwcs** used to create the WCS of the original image
- `PYWCSVER`, version of PyWCS used to create the WCS of the original image

These keywords are used for determining whether a headerlet can be applied to a given exposure and how it needs to be applied. Additional keywords provide more information about the solution itself, how it was derived, and by whom, through use of the following keywords:

- `AUTHOR`, name of person who created the headerlet
- `DESCRIP`, short description of the headerlet solution

- RMS_RA, RMS in R.A. at the reference pixel of the WCS stored in the headerlet solution, if updated from the Archive's default WCS
- RMS_DEC, RMS in Dec. at the reference pixel of the WCS stored in the headerlet solution, if updated from the Archive's default WCS
- NMATCH, number of sources used in the new solution fit, if updated from the Archive's default WCS
- CATALOG, astrometric catalog used for headerlet solution
- COMMENT, long description of how the headerlet solution was derived, if updated from Archive's default WCS

These keywords allow the headerlet to retain enough information about how the new solution was generated so that a user could determine if it can be applied to his or her copy of the image.

SIPWCS: A New WCS FITS Extension

All WCS-related keywords from the image's SCI headers, including all keywords referring to NPOL and D2IM extensions, are used to create SIPWCS headerlet extensions. The number of SIPWCS extensions directly correlate to the number of SCI extensions in the original science image. For instance, an ACS/WFC or WFC3/UVS image would have [SIPWCS,1] derived from [SCI,1] and [SIPWCS,2] derived from [SCI,2].

Keywords in SIPWCS extensions can then be used to overwrite SCI header keywords when a headerlet gets used to update a science image. The SIPWCS extension therefore serves not only as a record of the specific WCS solution derived for an image, but also as the source of values for replacing the image SCI headers WCS solution when desired. Keywords recording alignment information in the SIPWCS headesr also provide a record of how much of an offset there is between this solution and the default OPUS-generated solution.

3.4.5 Working with Headerlets

A set of tasks have been written, in the **stwcs.wcsutil.headerlet**, to support interactive management of headerlets covering operations from creating headerlets to applying headerlets, and even deleting headerlets applied to an image. The full set of interactive tasks with TEAL interfaces are:

apply_headerlet	apply a headerlet to a file
archive_headerlet	save a WCS solution as a headerlet extension and write it out as a headerlet FITS file
attach_headerlet	attach a headerlet as an extension to a file
delete_headerlet	delete a headerlet extension from a file
extract_headerlet	write out a headerlet extension as a separate FITS file
headerlet_summary	print a summary of all headerlet extensions in a file
restore_headerlet	replace current WCS solution with the WCS solution from a headerlet extension
write_headerlet	save a WCS solution as a separate headerlet FITS file

All these tasks are part of the STWCS package distributed along with the DrizzlePac package in the STScI_Python public release. This section describes how to use the most commonly-needed of these tasks to create headerlet FITS files and then apply them to science images. The full descriptions of all the remaining tasks goes beyond the scope of this handbook at this time (but please check the DrizzlePac website for updates). Each task, though, includes a detailed description of the task and its parameters in the help file which can be accessed through the TEAL GUI using the "Help" button.

Creating a Headerlet

The **tweakreg** task can generate headerlets for users when updating the input images with the fit that aligns those images to a reference image or catalog. Interactively, a headerlet can be created from any image using the **write_headerlet** task that will create a headerlet, write out the headerlet to a separate headerlet FITS file, and then, optionally, attach it as an extension to a science image (if it has not already been saved as an extension). The parameters for this task are:

Parameter	Default	Description
filename		Name(s) of science file(s) from which headerlets will be created and written. • String input formats supported include use of wild-cards, IRAF-style `"@"-files` (given as `@<filename>`) and comma-separated list of names. • An input filename will be expanded as necessary to interpret any environmental variables included in the filename. This allows the user to use something like ***`datadir$*flt.fits`*** as input, assuming "`datadir`" has been defined in the user's environment.
hdrname		Unique name for the headerlet, stored in the `HDRNAME` keyword, that will serve as the label for this headerlet.
output	None	Filename, or rootname, of the output headerlet FITS file to be written out by this task. If the string does not contain "`.fits`", it will create a filename starting with the science filename and ending with "`_hlet.fits`". A value of ***`mosaic1`***, for example, would result in the filename `ib3m23d1q_mosaic1_hlet.fits` when used on the image `ib3m23d1q_flt.fits`. If ***None*** is specified, a default filename based on the input filename will be generated for the headerlet FITS filename
sciext	SCI	Name (`EXTNAME`) of the extension that contains the WCS to be saved.
wcsname	None	Name of the WCS to be used to create the headerlet. It will return without doing anything when a blank string is given as input.
wcskey	None	Alternate WCS key for the WCS used to create the headerlet: values from ***A*** to ***Z***, or **" "** (blank string), or ***PRIMARY***. If a blank string or ***PRIMARY*** are specified, then it will create the headerlet from the primary WCS.
destim	None	The value of this parameter gets written out as the `DESTIM` keyword in the headerlet to denote that this headerlet can only be applied to that specific image. If set to ***None***, the `ROOTNAME` keyword value from the science image header is used.
sipname	None	Name of unique file which should be read to obtain the polynomial distortion coefficients. For a value of ***None***, the code looks for the keyword `SIPNAME` in the science header. • If not found, for HST data, it defaults to `IDCTAB`. • If there is no SIP model the value is `NOMODEL`. • If there is a SIP model but no `SIPNAME`, it is set to `UNKNOWN`.

Parameter	Default	Description
npolfile	None	Name of a unique file where non-polynomial distortions are stored. If the value ***None*** is given, the value gets determined by the code using: • The NPOLFILE keyword in the science header. • If NPOLFILE was not found and there is no *NPOL* model, it is set to NOMODEL. • If the *NPOL* model exists, it is set to UNKNOWN.
d2imfile	None	Name of a unique file where the detector to image correction was stored. If a value of ***None*** is given, the code get the value using: • The D2IMFILE keyword in the science header. • If D2IMFILE is not found and there is no *D2IM* correction, it is set to NOMODEL. • If the *D2IM* correction exists, but D2IMFILE is missing from science header, it is set to UNKNOWN.
author	None	Name of user who created the headerlet, used as the value for the AUTHOR keyword in the headerlet's PRIMARY header.
descrip	None	Short description of the solution provided by the headerlet, added as the value to the DESCRIP keyword in the headerlet's PRIMARY header.
history	None	Long (possibly multi-line) description of the solution provided by the headerlet. These comments will be added as HISTORY cards to the headerlet's PRIMARY header. If a file-name is specified, it will format and attach all text from that file as the history.
attach	True	Specify whether or not to attach this headerlet as a new extension. It will verify that no other headerlet extension has been created with the same ***hdrname*** value.
clobber	False	If the output headerlet file already exists, this parameter specifies whether or not to over-write that file
logging	False	This parameter enables logging of the operations to a file.

Basically, a headerlet can be created from a science image header and recorded with user-specified history and descriptions. This task can also be called directly using Python with the syntax:

```
>>> from stwcs.wcsutil import headerlet
   >>> headerlet.write_headerlet(filename, hdrname, output=None, sciext='SCI',
                        wcsname=None, wcskey=None, destim=None,
                        sipname=None, npolfile=None, d2imfile=None,
                        author=None, descrip=None, history=None,
                        nmatch=None, catalog=None,
                        attach=True, clobber=False, logging=False)
```

All the parameters used in the Python call are identical to those described for interactive use through the TEAL GUI.

Applying a Headerlet

Updating an image retrieved from the HST Archive with a headerlet, for example, requires the use of the task **apply_headerlet**. The full set of parameters used to run this task are:

Parameter	Default	Description
filename		File name of science observation whose WCS solution will be updated by the headerlet.
hdrlet		The filename of the headerlet to be applied to the science image.
attach	True	If set to ***True*** (default), it will append the headerlet to the FITS science file as a new extension after updating the science header with the WCS solution in the headerlet.
primary	True	This parameter specifies whether or not to replace the `PRIMARY` WCS with the WCS from headerlet.
archive	True	If set to ***True*** (default), then before updating the image's WCS, it will create a headerlet from the WCS solution already present in the science image header and add it as an extension to the science file. This allows the previous solutions to be backed up in the file in a way that allows the user to restore it as needed.
force	False	If set to ***True***, this will cause the headerlet to replace the current `PRIMARY` WCS even if it has a different distortion model. Applying a headerlet with one distortion model to an image which has a different distortion model can sometimes introduce errors in the coordinate transformations.
wcskey		Key value (***A*** to ***Z***, except O) to be used when writing the headerlet's WCS solution to the science header as an alternate WCS. If ***None***, the next available key will be used.
wcsname		Name to be assigned to the headerlet's WCS solution when writing it out to the science header as an alternate WCS. By default, it will use the `WCSNAME` keyword value from the headerlet itself. Headerlets require the use of the `WCSNAME` keyword, but this allows the user to change it as desired when applying it to a science file.
logging	False	This parameter enables logging of the tasks operations to a file.

The **apply_headerlet** task does not simply drop a new WCS solution into an image, but takes care to remember the previous solution in case it needs to be restored. The full set of operations performed by this task (some optional) include:

1. Create a headerlet from the original WCS solution in the science image (this step can be turned off).
2. Copy all WCS information from the science image to an alternate WCS using FITS Paper I standards.
3. Copy the WCS solution from the headerlet to the science observation.
4. Update the `WCSCORR` table with the linear distortion corrections WCS keyword values and name of the SIP solution (based on the name of the reference files) for each `SIPWCS` extension from the headerlet, along with the keyword values from the `PRIMARY` header of the headerlet.
5. Append the new headerlet to the science image as a single new extension (optional)

This process assumes that when an image header is updated with a headerlet, the new solution from the headerlet will become the prime WCS while providing the option to simply add the headerlets solution as an alternate WCS instead. The updated image can then be aligned to other images based on the original WCS or any headerlet WCS solution applied to that image.

The primary difficulty in applying a headerlet comes from insuring that the distortion model of the headerlet matches the distortion model described in the science image to be updated. Headerlets rely on the `DISTNAME` keyword to provide the description of the distortion models in the headerlet and in the science image. If the science image has a different distortion model than the one specified in the headerlet, the original distortion model from the science image gets moved into a headerlet and appended to the file as a new headerlet extension, then all the distortion information gets replaced with the new distortion information (keywords and distortion extensions) from the headerlet. The user can override this synchronization of models, if they absolutely feel the need, but this process insures that the WCS specified in the headerlet gets applied to the science image exactly as it was determined when creating the new solution for the headerlet. For most users, this will never be a concern unless they have the need to either redefine the distortion models based on their own calibrations or need to update data they retrieved with old distortion models with the latest calibrations.

This task can also be called directly using Python to apply the headerlet from a file to the `PRIMARY` WCS of a science file with this syntax:

```
>>> from stwcs.wcsutil import headerlet # only if it hasn't already been called
>>> headerlet.apply_headerlet_as_primary(filename,hdrlet,attach=True,archive=True,
force=False,logging=False)
```

Alternatively, to apply the headerlet as an alternate WCS solution in the science image, the following syntax can be used:

```
>>>headerlet.apply_headerlet_as_alternate(filename,hdrlet,attach=True,wcskey=None
, wcsname=None,logging=False)
```

All the parameters for both functions correspond to parameters already described for use with the TEAL GUI. Additional information about using Python syntax to run headerlet tasks are described in the next section.

3.4.6 Python Syntax of the Headerlet Tasks

The material covered in this section requires some experience working with Python.

This section describes the currently implemented Python syntax for working with headerlets supported by the **stwcs.wcsutil.headerlet** module. First, there's a potentially confusing point that should be cleared up: a headerlet, as implemented, is simply a FITS file containing multiple extensions that contain all of the parameters necessary to reproduce the WCS solution in the science image from which it was created.

When a headerlet is applied to an image, a copy of the original WCS information is appended to the image's HDU (Header Data Unit) list as a special extension HDU called a *Headerlet HDU*. A Headerlet HDU consists of a simple header describing the headerlet, and has as its data, the headerlet file itself, which may be compressed. A Headerlet HDU has an `XTENSION` value of `HDRLET`. Common FITS readers (such as **fv**) will see this extension as an image extension. PyFITS in conjunction with STWCS, on the other hand, allows the user to actually work with that data array of the headerlet HDU as if it were a separate headerlet file, complete with extensions, as described below. This makes it easy for users to verify and work with the contents of each headerlet HDU separately.

The Python syntax, on the other hand, works primarily with the *Headerlet object;* a Python class that contains all the WCS keyword information and distortion information, as well as built-in methods (akin to functions) for applying, writing out, or performing other operations on this object. The rest of this discussion will help document how to use the Headerlet object in Python tasks or scripts as needed.

Software Requirements

Working with headerlets only requires **stwcs** and all the tasks defined in it (such as those reported for use with TEAL when the commands `import stwcs` or `reload(stwcs)` are issued). This package comes as part of the standard STScI Python package and can be used independently of **drizzlepac**. A full description of the GUI interfaces for all the headerlet tasks from this package can be found in Section 4.4.5.

Getting Headerlet Function Names

The full set of functions available for use with headerlets can be found in the **stwcs.wcsutil.headerlet** module which contains functions that can be called to perform some predefined set of operations from start to end, while the Headerlet class provides basic functionality used by all the functions in this module and by the TEAL GUI.

The full list of functions can be obtained using Python's own introspection functionality:

```
>>> from stwcs.wcsutil import headerlet
>>> dir(headerlet)
```

Under *ipython*, a user can also use the `help()` function to get more specific help for any function listed in the module, such as:

```
>>> help(headerlet.create_headerlet)
```

Creating a Headerlet Object

The headerlet module in **stwcs** provides the *createHeaderlet()* function to simply create a Headerlet object from an input science image. This function actually gets called by the **write_headerlet** task and *write_headerlet* function from this package, both of which have already been described. The *write_headerlet()* function creates the headerlet from a file, then automatically writes out the headerlet to a file without actually returning the Headerlet object for continued use. As a result, the following describes how to create a Headerlet object for use in Python tasks or scripts.

The call for creating a Headerlet object uses the *createHeaderlet()* function with the following syntax:

```
>>> from stwcs.wcsutil import headerlet
>>> hdrlet = headerlet.create_headerlet(filename,sciext='SCI',hdrname=None,
destim=None,wcskey=" ",wcsname=None,sipname=None,npolfile=None,d2imfile=None,
author=None,descrip=None,history=None,nmatch=None,catalog=None,logging=False,
logmode='w')
```

All the parameters used by the *create_headerlet()* function correspond to already described parameters used by the **write_headerlet** task, with one exception. The ***logmode*** parameter allows the user to specify whether or not to overwrite ("w") or append ("a") to the log when writing out processing comments to a log file with ***logging=True***.

An example of how to use this function to create a Headerlet object from the image ACS/WFC `j94f05bgq_flt.fits` and label the Headerlet with the name "VERSION1" would be:

```
>>> from stwcs.wcsutil import headerlet
>>> hdrlet = headerlet.createHeaderlet('j94f05bgq_flt.fits', 'VERSION1')
>>> type(hdrlet)
 <class 'stwcs.wcsutil.headerlet.Headerlet'>
>>> hdrlet.info()
 Filename: (No file associated with this HDUList)
 No.    Name         Type      Cards   Dimensions   Format
 0     PRIMARY     PrimaryHDU    12     ()
 1     SIPWCS      ImageHDU     111     ()
 2     SIPWCS      ImageHDU     110     ()
 3     WCSDVARR    ImageHDU      15     (65, 33)    float32
 4     WCSDVARR    ImageHDU      15     (65, 33)    float32
 5     WCSDVARR    ImageHDU      15     (65, 33)    float32
 6     WCSDVARR    ImageHDU      15     (65, 33)    float32
 7     D2IMARR     ImageHDU      12     (4096,)     float32
```

Here, the Headerlet object is similar to a normal PyFITS *HDUList object*. *createHeaderlet()* can be given either the path to a file, or an open HDUList, as its first argument.

Applying a Headerlet Object

What gets done to a Headerlet object? The main purpose of a Headerlet object is to apply its WCS solution to the header of another FITS file. This can be done using the *Headerlet.apply()* method:

```
>>> hdrlet.apply('some_other_image.fits')
```

Or use the *applyHeaderlet()* convenience function that takes an existing headerlet file path or object as its first argument. The remaining arguments are the same as those used when calling *Headerlet.apply()*. As with *createHeaderlet()*, both of these can take a file path, or opened HDUList objects, as arguments.

Accessing the Headerlet Extension using PyFITS

When a headerlet is applied to an image, an additional headerlet containing that image's original WCS solution is automatically created, and is appended to the file's HDU list as a Headerlet HDU. However, this behavior can be disabled by setting the *createheaderlet* keyword argument to *False* in either *Headerlet.apply()* or *applyHeaderlet()*.

When opening a file that contains `Headerlet HDU` extensions, it will normally appear in PyFits as follows:

```
>>> import pyfits
 >>> hdul = pyfits.open('94f05bgq_flt_with_hlet.fits')
 >>> hdul.info()
 Filename: j94f05bgq_flt_with_hlet.fits
 No.    Name         Type        Cards   Dimensions    Format
 0      PRIMARY    PrimaryHDU     248    ()            int16
 1      SCI        ImageHDU       286    (4096, 2048)  float32
 2      ERR        ImageHDU        76    (4096, 2048)  float32
 3      DQ         ImageHDU        66    (4096, 2048)  int16
 4      SCI        ImageHDU       282    (4096, 2048)  float32
 5      ERR        ImageHDU        74    (4096, 2048)  float32
 6      DQ         ImageHDU        66    (4096, 2048)  int16
 7     WCSCORR    BinTableHDU     56    10R x 23C     [40A,I,1A,D,D,D,D,D,D,D,24A,
                                                        24A,D,D,D,D,D,D,D,J,40A]
 8      WCSDVARR   ImageHDU        15    (65, 33)      float32
 9      WCSDVARR   ImageHDU        15    (65, 33)      float32
 10     WCSDVARR   ImageHDU        15    (65, 33)      float32
 11     WCSDVARR   ImageHDU        15    (65, 33)      float32
 12     D2IMARR    ImageHDU        12    (4096,)       float32
 13     HDRLET     HeaderletHDU    13
 14     HDRLET     HeaderletHDU    13
```

The names of the separate headerlet extensions, reported as HeaderletHDUs, are both `HDRLET`, but its type shows up as `HeaderletHDU` when STWCS has been installed. The headers can be read, and although the data can be read, the user would have to know what to do with it, as the data has been converted to a tar file containing the headerlet FITS object. The structure of this HDU is the same as that proposed for the FITS extension type proposed by NRAO on:

`http://listmgr.cv.nrao.edu/pipermail/fitsbits/2002-April/thread.html.`

If the **stwcs.wcsutil.headerlet** module is imported, though, PyFITS will recognize these extensions as special Headerlet HDUs and allow viewing of the data in the

Headerlet. The summary of the FITS extensions for an image with a Headerlet HDU can be accessed using the following Python commands:

```
>>> import stwcs.wcsutil.headerlet
>>> # Note that it is necessary to reopen the file
>>> hdul = pyfits.open('j94f05bgq_flt_with_hlet.fits')
>>> hdul.info()
 Filename: j94f05bgq_flt_with_hlet.fits
 No.    Name         Type        Cards Dimensions    Format
 0     PRIMARY     PrimaryHDU     248  ()            int16
 1     SCI         ImageHDU       286  (4096, 2048)  float32
 2     ERR         ImageHDU        76  (4096, 2048)  float32
 3     DQ          ImageHDU        66  (4096, 2048)  int16
 4     SCI         ImageHDU       282  (4096, 2048)  float32
 5     ERR         ImageHDU        74  (4096, 2048)  float32
 6     DQ          ImageHDU        66  (4096, 2048)  int16
 7    WCSCORR     BinTableHDU     56  10R x 23C     [40A,I,1A D,D,D,D,D,D,D,D,24A,
                                                     24A,D,D,D,D,D,D,D,J,40A]
 8     WCSDVARR    ImageHDU        15  (65, 33)      float32
 9     WCSDVARR    ImageHDU        15  (65, 33)      float32
 10    WCSDVARR    ImageHDU        15  (65, 33)      float32
 11    WCSDVARR    ImageHDU        15  (65, 33)      float32
 12    D2IMARR     ImageHDU        12  (4096,)       float32
 13    HDRLET      HeaderletHDU    13
 14    HDRLET      HeaderletHDU    13

>>> hdul['HDRLET', 1].header
 XTENSION= 'HDRLET  '            / Headerlet extension
 BITPIX  =                    8 / array data type
 NAXIS   =                    1 / number of array dimensions
 NAXIS1  =               102400 / Axis length
 PCOUNT  =                    0 / number of parameters
 GCOUNT  =                    1 / number of groups
 EXTNAME = 'HDRLET  '            / name of the headerlet extension
 HDRNAME = 'j94f05bgq_orig'      / Headerlet name
 DATE    = '2011-04-13T12:14:42' / Date FITS file was generated
 SIPNAME = 'IDC_qbu1641sj'       / SIP distortion model name
 NPOLFILE= '/grp/hst/acs/lucas/new-npl/qbu16424j_npl.fits' / Non-polynomial corre
 D2IMFILE= '/grp/hst/acs/lucas/new-npl/wfc_ref68col_d2i.fits' / Column correction
 COMPRESS=                    F / Uses gzip compression
```

HeaderletHDU objects are similar to other HDU objects in PyFITS. However, they have a special .headerlet attribute that returns the actual FITS formatted headerlet contained in the HDU data as a Headerlet object as if the user were reading it directly

from a standalone headerlet FITS file with a `_hlet.fits` filename. The .headerlet object can be read using the following Python syntax:

```
>>> hdrlet = hdul['HDRLET', 1].headerlet
 >>> hdrlet.info()
 Filename: (No file associated with this HDUList)
 No.    Name         Type        Cards  Dimensions    Format
 0      PRIMARY      PrimaryHDU     12  ()            uint8
 1      SIPWCS       ImageHDU      111  ()            uint8
 2      SIPWCS       ImageHDU      110  ()            uint8
 3      WCSDVARR     ImageHDU       15  (65, 33)      float32
 4      WCSDVARR     ImageHDU       15  (65, 33)      float32
 5      WCSDVARR     ImageHDU       15  (65, 33)      float32
 6      WCSDVARR     ImageHDU       15  (65, 33)      float32
 7      D2IMARR      ImageHDU       12  (4096,)       float32
```

This is useful for viewing the contents of the headerlets attached to a file without having to write out the Headerlet to its own separate FITS file, especially if this image was updated by someone else.

Other Headerlet Extension Functions

The presence of a `HeaderletHDU` (headerlet extension) in a science image's FITS file indicates that at one point the science image's WCS was updated using the information from that headerlet. In fact, many updates may have been applied to a science image's WCS making it possible that multiple headerlet extensions may be present in the science image's FITS file. The remaining functions provided by the **stwcs.wcsutil.headerlet** module allows a user to work with these multiple headerlet solutions to perform such operations as deleting extraneous headerlet extensions, write out the headerlet extension as a standalone FITS file, or even use a headerlet extension to update the WCS of the science image.

These additional capabilities can be broken down into two separate sets: use of the Headerlet object and use of headerlet FITS files. Once a Headerlet object has been created in memory, either from a .headerlet attribute of a HeaderletHDU as illustrated in the previous example or from reading in a headerlet FITS file, it can be used to perform several operations as defined by the following methods of the Headerlet class.

The functions provided by the headerlet module for working with headerlet FITS files covers all the same basic operations supported by the Headerlet class and its methods while using headerlet FITS files as direct inputs for each function. Unfortunately, providing full details on all the available operations and how to perform those operations go beyond the scope of this handbook at this time. Full details can be obtained from the documentation included with the software itself.

```
apply_as_alternate : function apply_as_alternate(self, fobj, attach=True, wcskey=None, wcsname=None)
    This method allows a Headerlet object to be added to the list of alternate WCS solutions for the science data in the FITS file.
 apply_as_primary : function apply_as_primary(self, fobj, attach=True, archive=True, force=False)
    This method allows a Headerlet object to replace the primary(default) WCS of the science data in the FITS file.
 attach_to_file   : function attach_to_file(self, fobj, archive=False)
    This method allows a Headerlet object to be added as a new HeaderletHDU extension in a science image's FITS file.
 build_distname   : function build_distname(self, dest)
    This method creates the value for the DISTNAME keyword which gets used to identify what distortion model has been included in the headerlet.
 hverify          : function hverify(self)
    This method verifies that the Headerlet object has all the necessary information in the PRIMARY header to be considered a valid headerlet.
 info             : function info(self, columns=None, pad=2, maxwidth=None, output=None, clobber=True, quiet=False)
    A summary of the key distortion and WCS related keywords from the Headerlet will be reported in a table format.
 summary          : function summary(self, columns=None)
    A summary of the key distortion and WCS related keywords from the Headerlet will be reported as a Python dictionary.
 tofile           : function tofile(self, fname, destim=None, hdrname=None, clobber=False)
    A Headerlet object can be written out to it's own FITS file using this method.
 verify_dest      : function verify_dest(self, dest)
   Use this method to verify whether or not this Headerlet can be applied to the destination science image based on the DESTNAME (or ROOTNAME) keywords.
 verify_hdrname   : function verify_hdrname(self, dest)
    Use this method to verify whether or not the destination science image already has a headerlet applied to it with the same HDRNAME label.
 verify_model     : function verify_model(self, dest)
    Use this method to verify whether or not this Headerlet can be applied to the destination science image by checking that the distortion model in
the Headerlet matches the distortion model in the science image.
```

CHAPTER 4:

DrizzlePac Package

In this chapter…

4.1 DrizzlePac: An Overview

DrizzlePac is a package of tasks used primarily for post-pipeline drizzling of images. **astrodrizzle** is its flagship task, and the rest of DrizzlePac's tasks support more complex drizzling operations.

AstroDrizzle replaces MultiDrizzle in the pipeline and for post-pipeline image drizzling. It uses the same algorithmic base as MultiDrizzle with a modified set of core routines that have been recoded in C and Python.

AstroDrizzle also introduces a conceptual change in how drizzling is handled, from moving pixels according to shifts to moving them according to the astrometry encoded in the WCS. This chapter provides a description of the software interface to AstroDrizzle and the other tasks in the DrizzlePac package.

In a nutshell, AstroDrizzle processes a set of pipeline flat-field calibrated files (`flt.fits`) as follows:

- Each input image is aligned to a reference image based on WCS information in the image headers.
- Using this astrometric information, and the distortion information in the header, the images are separately drizzled onto undistorted output images in a common reference frame.
- These distortion-free output images are combined to create a median image.
- The median image is then blotted, or reverse-drizzled, back to the frame of each input `flt.fits` image.

- By comparing each `flt.fits` image with its blotted counterpart median image, the software locates bad pixels in each of the original `flt.fits` frames and creates bad pixel masks. These bad pixels are typically cosmic rays or faulty pixels in the detector.
- In the final step, all images are drizzled, using the mask files, onto a single output image. This produces an image that's corrected for geometric distortion and largely free of cosmic rays and detector artifacts.

In the pipeline, AstroDrizzle task parameters are given default values selected to cover a wide range of data. These parameter values are stored in a reference file named by the `MDRIZTAB` image header keyword. However, these values may not be well suited for some images. In such cases, users may elect to reprocess the images to obtain better quality results.

In this section, as well as Chapters 6 and 7, information will be provided to help users select the best possible task parameter values suited for reprocessing their data. This includes combining images spread over several visits at different roll angles–this type of data is almost always taken using different guide stars, and due to positional uncertainties in the Guide Star Catalog, the coordinate frames of each visit could be significantly misaligned as much as 0.5 arcseconds. A DrizzlePac task called **tweakreg** will serve as a useful tool, in most imaging cases, for aligning all multi-visit images to a common WCS. There are also tools for converting pixel positions to sky coordinates, and vice versa, as well as translating pixel positions for an image before and after geometric distortion corrections.

Software Requirements

DrizzlePac has been designed and implemented as a package in Python. AstroDrizzle and other DrizzlePac tasks maintain their own command-line interface in the Python environment. Tasks in DrizzlePac can also be run using the TEAL GUI, an interface that resembles the "epar" GUI in PyRAF. TEAL is part of another Python-based package called **stsci.tools,** and is invoked by typing **epar** followed by the taskname in PyRAF. These packages have been designed to work together without requiring any manual settings of the environment by the user.

DrizzlePac requires at least Python 2.6, PyFITS v3.0 and Numpy v1.6 to be installed for operation with the Python interfaces. In addition, PyRAF v1.10 or greater is required for use of the GUI interface. Additional information is available at the STScI Python webpage and the DrizzlePac webpage.

4.2 AstroDrizzle: The New Drizzle Workhorse

Note for Experienced MultiDrizzle Users: astrodrizzle parameters are essentially the same as those in multidrizzle. There are, however, three new parameters: stepsize, resetbits, and num_cores. Please refer to their descriptions in Table 4.1

AstroDrizzle uses flat-field calibrated images (`flt.fits`, `c0m.fits` for WFPC2, `flc.fits` for CTE-corrected ACS images) as input files. There are two main components to the process: (1) create mask files for bad pixels and cosmic rays; (2) drizzle-combine the input images using the mask files, while applying geometric distortion corrections, to create a "clean" distortion-free combined image.

There are three components to mask files used by AstroDrizzle. Information about bad pixels flagged in the input images data quality array are taken into account. A "static pixel mask" is created by identifying pixels with abnormally low values in each image.

The third component is cosmic ray masks, one for each input `flt.fits` image. Creating cosmic ray masks are a major part of AstroDrizzle, so they are briefly described below (with more detailed information available later in this section):

- Each input `flt.fits` image is drizzled to create an undistorted copy in the output frame, where the WCS information in the single-drizzled image header is used to align it with respect to a reference image.
- The aligned single-drizzled images are combined to create a "clean" median image that approximates the appearance of a clean distortion-free combined image.
- The "clean" median image is "blotted" (or reverse-drizzled) back to the distorted image frames of each input `flt.fits` image.
- By comparing each `flt.fits` file with its counterpart median image, cosmic rays and other bad pixels can be identified and stored in a mask file.

These mask files are used in the final step where each input image is drizzle-combined to create a clean distortion-free output image. Depending on the number of images and type of observations, users can elect to adjust parameters to improve the quality of the final drizzle-combined image.

Note Regarding WFC3/IR Images

For IR images, the steps "Create Median Image," "Blot back the Median Image," and "Remove Cosmic Rays with Deriv, Driz-CR" in **astrodrizzle** may be turned off since cosmic rays are flagged in **calwfc3** as part of the "up-the-ramp fitting." However, it may be useful to run those steps (using a different bit flag, like 8192, for "cosmic rays" found during AstroDrizzle processing) to flag additional detector

artifacts not present in the data quality arrays of the calibrated images. Note that it is very important to subtract the sky prior to drizzing the final image, or the science array will be compromised by increased noise.

Detailed Description of the AstroDrizzle Steps

4.2.1 Set-up and Initialize astrodrizzle Parameters

Several initialization steps are performed by AstroDrizzle to set some preferences on how the code will execute and to insure the input data files have the correct format. These include specifying the input files, providing a name and format for the output image, and resetting cosmic ray flags in the data quality arrays of input images. In pipeline processing, AstroDrizzle is instructed to use specific task parameter values stored in the `MDRIZTAB` reference file, and the WCS is always updated to account for distortion corrections.

Parameter Details

Initialization parameters in **astrodrizzle** are listed below. Text in Arial font are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.1: Parameters for astrodrizzle: "Initialization"

Parameter	Default	Description
input	*flt.fits	Input files (name, suffix, or @list) • To drizzle a single image or association, specify the filename. • For several input images, provide a wildcard followed by a suffix (e.g., ***`*flt.fits`***) or a comma separated list of filenames. • For processing specific images in a directory, provide the list of images in a file with prefix "@" (e.g., `@filelist`). • If a user specifies "***final_wht_type=IVM***", the input file list can have two columns, one for the image and the second for the user-generated inverse variance maps (IVM) file If no IVM files are specified, **astrodrizzle** will automatically generate it for each input image.
output	final	`Rootname for output drizzled products.` The default rootname is '***final***.' • If ***build=False***, these individual data products are created: science image (e.g.,`final_drz_sci.fits`), weight image (e.g., `final_drz_wht.fits`), and if ***context=True***, the context image (e.g., `final_drz_ctx.fits`). • If ***build=True*** one image with three extensions will be created (e.g., ***final_drz.fits***). The first extension is the drizzled science image, the second is the weight image, and if ***context=True***, the third extension is the context image. If ***context=False***, the third extension is empty.
Button in the Teal window, **"Update from MDRIZTAB"**	n/a	Clicking on this button tells **astrodrizzle** to populate its parameters using values that were pre-determined and stored in the reference file named in the input image header keyword `MDRIZTAB`. In the pipeline, AstroDrizzle uses this reference file to populate parameters for different types of association.
runfile	astrodrizzle.log	File for logging the script commands. As **astrodrizzle** runs, comments about its progress are displayed on the window. This file captures those comments, making it a useful resource for tracking down problems.

Table 4.1: Parameters for astrodrizzle: "Initialization" (Continued)

Parameter	Default	Description
updatewcs	False	Update the WCS keywords? **updatewcs**, a task in the **stwcs** package, updates the values of basic WCS keywords and SIP convention keywords that hold astrometric information about an image. Use of this parameter requires the availability of the `IDCTAB` distortion reference file. After a file has been updated in the pipeline with this step, the `IDCTAB` distortion reference file is no longer needed when running AstroDrizzle for reprocessing data, as was the case for MultiDrizzle. In most instances, the default setting ***False*** should be used since **updatewcs** was already run for images retrieved after AstroDrizzle was installed in the pipeline. (For post-pipeline combining of images from multiple visits, WCS information can be updated using **tweakreg**.)
wcskey	""	WCS version to use in processing This value, a string, is the label of a WCS solution to be used when doing post-pipeline AstroDrizzle processing. Every time a WCS gets recorded as a new solution in an image, it is assigned a label stored in the keyword `WCSNAME*`. If this parameter is used, all input images are required to have the same WCS key value. (These WCS updates are done following FITS Paper I alternate WCS standards.)
proc_unit	native	Units used during processing By default, AstroDrizzle uses units specified by the `BUNIT` keyword in `flt.fits` and `flc.fits` headers. For ACS and WFC3, units are specified in electrons. For WFPC2, unit are in DN.
coeffs	True	Use header-based distortion coefficients? The default, ***True***, tells AstroDrizzle to use distortion keyword values stored in the header.
context	True	`Create context image during final drizzle?` The context image uses each bit (power of 2) in a 32-bit integer to record what input image contributed to the final value of the output pixel. For instance, the bits are set as follows: image 1/chip 1 = 1, image 1/chip 2 = 2, image 2/chip 1 = 4, image 2/chip 2 = 8, image 3/chip 1 = 16, and image 3/chip 2 =32, and so on. Each pixel in the context image is an additive combination of some of these values, depending on which images were used. If the user combines three UVIS images, a pixel with contributions from image1/chip1, image2/chip2, and image3/chip3 as the sum of the integer values 1, 4, and 16 respectively, for a total value of 21. Should more than 32 chips be combined into a final output image, this context array will be "expanded" by creating another complete 32-bit integer array of the same size and layer, on top of the original to create a three-dimensional array with 2 layers. The new layer would then be used to keep track of the next 32 chips, with additional layers being created for further sets of 32 chips as needed.
group	""	Single extension or group to be combined/cleaned By default, all detector extensions in a FITSs file are processed. To drizzle only one chip, specify the group number for that image. For example, drizzling only chip 2 in ACS/WFC could be specified as ***1*** or ***sci,1***.
build	False	Create multi-extension output file for final drizzle? • If ***build=False*** three separate data products are created: science image (e.g.,`rootname_drz_sci.fits`), weight image (e.g., `rootname_drz_wht.fits`), and context image (e.g., `rootname_drz_ctx.fits`). • If ***build=True*** one image with three extensions, a multi-extension FITS file, will be created (e.g., `rootname_drz.fits`). The first extension is the drizzled science image, the second is the weight image, and the third extension is the context image.
staticfile	""	Name of (optional) input static bad-pixel mask This is a user-created static file for bad pixels.
crbit	4096	`Bit value for CR ident. in DQ array` When AstroDrizzle combines several images, whether its associated images in the pipeline or during post-pipeline processing, cosmic rays identified during processing are flagged in the input `flt.fits` data quality (DQ) extensions. In the HST pipeline, the DQ bit value used for cosmic rays is 4096.

Table 4.1: Parameters for astrodrizzle: "Initialization" (Continued)

Parameter	Default	Description
stepsize[1]	10	Step size for drizzle coordinate computation This default value of 10 means that geometric distortion corrections are directly performed for every 10th pixel in *x* and *y*. For pixels between pairs of pixels with directly-calculated distortion corrections, interpolation between the pair is used to determine their distortion corrections values.
resetbits[1]	4096	Bit values to reset in all input DQ arrays In the pipeline, AstroDrizzle updates the `flt.fits` data quality (DQ) arrays to flag cosmic rays found when combining associated flt.fits images. During reprocessing, these cosmic ray bits should be reset by the user to 0 (good pixels), so cosmic rays can be re-identified using user-specified **astrodrizzle** parameter values. Re-identification of cosmic rays is also needed for multi-visit image combination due to misaligned WCSs (caused by different guide star pairs used for each visit) that require realignment using **tweakreg**. This parameter can be used to reset any or all bit values found in the input images' data quality arrays.
num_cores[1]	None	Max CPU cores to use (n < than 2 disables, None = auto-decide) This specifies the number of CPU cores to use during processing. Any value less than two will disable parallel processing.

1. New parameters in **astrodrizzle**.

Table 4.2: Parameters for astrodrizzle: "State of Input Files"

Parameter	Default	Description
preserve	True	Copy input files to archive directory, if not already archived? When set to the default value, ***True***, **astrodrizzle** creates a subdirectory called `OrIg_files` to store a copy of the input `flt.fits` files before they're modified by AstroDrizzle. If `OrIg_files` already exists, the files in it are not overwritten.
overwrite	False	Copy input files into archive, overwriting if required? If the directory `OrIg_files` already exists, this parameter allows users to overwrite the directory contents with input data for the current **astrodrizzle** run.
clean	False	Delete temporary files after completion? Intermediate files created by AstroDrizzle are automatically removed after processing by setting this parameter to ***True***. For trouble-shooting purposes, setting this parameter to ***False*** could allow users to track down processing problems, or the files could be useful to new users who would like to inspect the intermediate products.
restore	True	Copy input files from archive directory for processing? This is a useful option if **astrodrizzle** is re-run from scratch in the same directory using original input files in directory `OrIg_files`.

4.2.2 Create a Static Mask Containing Permanent Bad Pixels

A detector-based static mask is created to flag bad pixels in the input images. First, severely negative or low value pixels are identified in each image to create individual bad pixel masks. For each detector in the list of input images[1], the individual masks are combined to create a master-mask file for use in the final drizzle step.

1. AstroDrizzle input images can be from more than one instrument, such as cases where ACS/WFC and WFC3/UVIS images are being combined.

If the AstroDrizzle static mask creation step is turned on, a static mask is created that identifies pixels with abnormally low values which could be due to over-subtraction of bad pixel values when applying the dark image correction in routine calibrations, or from post-pipeline over-subtraction of high sky levels. The clipping level for identifying bad pixels is an integer multiple of the RMS below the image's mode. That number is given by the **astrodrizzle** parameter ***static_sig***.

How the Static Mask is Used in Subsequent astrodrizzle Steps

The bad pixel mask used in the single-drizzle step in AstroDrizzle contains the static mask created as described in the previous paragraph, combined with any user supplied static mask as specified in the ***staticfile*** parameter, along with a mask of bad pixels described in the input image's data quality arrays as flagged by the parameter ***driz_sep_bits***.

The bad pixel mask used in the final drizzle-combining step in AstroDrizzle contains all bad pixels from all input images as used in the single-drizzle step, as well as the mask of pixels identified as cosmic rays as determined from running the ***driz_cr*** step. This allows AstroDrizzle to ignore all recognizable bad pixels during processing, including the bad pixels generated as a result of detector sensitivity issues (as identified in the static mask or data quality mask), as well as all user-identified bad pixels (through the ***staticfile*** mask).

Users should consider the details of their science image and decide if creating this mask is appropriate for the resulting science. For instance, if the image field is very crowded, or if it contains mostly nebulous or extended objects, the statistics used to create the static mask could be heavily skewed resulting in a mask that flags sources as bad pixels.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.3: Parameters for astrodrizzle: "Static Mask" Creation

Parameter	Default	Description
static	True	Create static bad pixel mask from the data?
static_sig	4	Sigma*rms below mode to clip for static mask Abnormally low pixel values that deviate by more than a value of ***static_sig*** sigma below the image median are flagged as bad pixels in the static pixel mask. These are generally caused by bad pixel oversubtraction in the dark image during calibration.

4.2.3 Perform Sky Subtraction

In the pipeline, AstroDrizzle sky computation is based on the statistical distribution of pixel values in an input image. It is performed by iterative sigma-clipping, starting with the full range of pixel intensity values, to calculate the standard deviation. By default, pixel values deviating from the median value (specified by parameter ***skystat***) by over four sigma are rejected, and this operation is repeated for a total of five iterations. The median value of the final distribution is used as the sky value, stored in

the header keyword `MDRIZSKY` in the original `flt.fits` image (*the original input images themselves are NOT sky-subtracted at this stage*).

The default method provides good results for a wide range of datasets. However, the sky is occasionally slightly overestimated, and such cases can be improved in post-pipeline AstroDrizzle processing.

If the sky in a field is strongly affected by bright sources, the primary purpose of a background measurement is to ensure that there are no additional varying offset levels in the exposures prior to combination. Here, the varying "sky" levels may be due to any number of external sources (i.e., bright earth limb), but in a field of view dominated by bright source, there would not be enough sky pixels beyond the source to accurately determine the true sky. This leads to the possibility that sky variations from one exposure to the next could ultimately make the true background difficult or potentially impossible to determine.

Flat-field calibrated images (`flt.fits`, `c0m.fits` for WFPC2, `flc.fits` for CTE-corrected ACS images) delivered to users from the Archive are *not* sky-subtracted. However, the sky values calculated by AstroDrizzle in the pipeline are held in the group header keyword `MDRIZSKY`. The computed sky value is subtracted from a copy of the input image; these sky-subtracted image copies are later used in the final image combination step.

For cameras with multiple detectors (four in WFPC2, two each in ACS/WFC and WFC3/UVIS), sky values in an image are measured separately for each detector. The lowest measured detector sky value is adopted for all detectors for that exposure. This is based on the premise that the pixel intensity distribution will be higher in one or more detectors with extended or bright objects, thereby overestimating their sky value. In other words, the sky value in the detector least affected by bright or extended objects would provide a more realistic sky determination.

In the pipeline, sky subtraction is performed for broad-band data, and turned off for narrow-band data and UV observations that are "dark" (i.e., those that have no geocoronal emission in the filter bandpass). This is done for the following reasons:

- The sky background through such filters is much lower than through optical broad-band filters
- These observations are often extended diffuse emission-line targets with flux that is much higher than the background. In such situations, the automated sky calculations by AstroDrizzle will lead to the introduction of errors larger that the background value. However, if the user is able to determine an accurate background level for such images, from parts of the image with measurable sky, then a user-specified sky value may be used to propagate those values directly to AstroDrizzle.

Sky subtraction is generally recommended for optimal flagging and removal of cosmic rays, when the sky background is more than a few electrons. However, for some science applications where the sky should not be removed, the final drizzle step can be performed with no sky subtraction (***skysub*** set to ***False*** in **astrodrizzle**). If sky subtraction is turned off, then the ***final_pixfrac*** parameter should also be set to ***1***; otherwise, variations in sky between images will add noise to the data.

Methodology

Steps for determining sky value for each image:

1. For each chip in the input image, obtain an estimate of the sky background by computing the clipped statistics (set by the parameters ***skyclip***, ***skystat***, ***skylsigma***, and ***skyusigma***).
2. The lowest sky value relative to area on the sky (in units of electrons/square-arcsecond) among all the chips is then adopted for all chips in the image. That value is then rescaled to the plate scale of each chip to become the sky value for that chip.
3. The science header of each input image is updated with this value.
4. Sky subtraction is not applied to the `flt.fits` images. Instead, the sky value is subtracted from the chip, on-the-fly, during the process of drizzling the image.

The default estimate of the sky background relies on computing the clipped median for each input chip. Offline processing by the user, however, could use a number of statistical operations to independently estimate the sky background, including the mode, mean and median.

A user-determined sky value, already subtracted from the input image, can be entered as a new keyword in the input file header–this keyword name is given by the **astrodrizzle *skyuser*** parameter. This user-supplied value will then get used, instead of AstroDrizzle's computed sky value, when the sky-subtraction step is run.

However, if the sky subtraction step is turned off, AstroDrizzle will still use the sky value recorded in the `MDRIZSKY` keyword when performing single-image drizzling and cosmic ray identification, as it provides the only indication of the background sky level needed for the statistical computations used to identify cosmic rays.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.4: Parameters for astrodrizzle: "Sky Subtraction"

Parameter	Default	Description
skysub	True	Perform sky subtraction?
skywidth	0.3	Bin width for sampling sky statistics (in sigma) Bin width, in sigma (standard deviation), used to sample the distribution of pixel flux values in order to compute the sky background statistics.
skystat	median	Sky correction statistics parameter Statistical method for determining the sky value from the image pixel values. Valid options are: • ***median*** • ***mode*** • ***mean***
skylower	None	Lower limit of usable data for sky
skyupper	None	Upper limit of usable data for sky
skyclip	5	Number of clipping iterations Number of clipping iterations to use when computing the sky value.
skylsigma	4.0	Lower side clipping factor (in sigma) Lower clipping limit, in sigma, used when computing the sky value.
skyusigma	4.0	Upper side clipping factor (in sigma) Upper clipping limit, in sigma, used when computing the sky value.
skyuser	""	KEYWORD indicating a sky subtraction value if done by user Name of header keyword which records the sky value already subtracted from the image by the user.
skyfile	""	Name of file with user-computed sky values Name of file containing user-computed sky values for use with each input image. This ASCII file should only contain two columns: (1) image filename in column 1; (2) sky value in column 2. The sky value should be provided in units that match the units of the input image. For multichip images, the same value will be applied to all chips.

4.2.4 Create Separately Drizzled and Registered Images

Each input image is drizzled to create an undistorted copy in the output frame[2]. These images are registered with respect to a reference image (by default, the first image in the input list) using the WCS information in their headers, and will be used in the next three steps to create cosmic ray masks for each input image.

These single-drizzled images are generated using the full WCS information provided by each of the input images. Any WCS offsets will show up as misaligned sources in the single-drizzled images. In post-pipeline processing, these images can be used for refining the image registration for each of the input images, if the user decides to use image registration software other than the **tweakreg** task.

The default values in this step will define the output frame that will ideally include all input pixels from all the WCS-registered single-drizzled input images. But like the final drizzle step, the output frame may be redefined at this step using different parameter settings

2. Two output files are created: a single-drizzled science image and a single-drizzled weight image, both in a simple FITS format.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.5: Parameters for astrodrizzle: "Drizzle Separate Images"

Parameter	Default	Description
driz_separate	True	Drizzle onto separate output images? This parameter specifies whether or not to drizzle each input image onto separate output images. The separate output images will be corrected for geometric distortion and aligned to the reference image based on their WCS. These images are used to create the median image, needed for the cosmic ray rejection step.
driz_sep_kernel	turbo	Shape of kernel function The default ***turbo*** kernel function is only used in the "drizzle separate images" step, not in the final drizzling step. It specifies the form of the kernel function used to distribute flux onto the separate output images. Several options are available: • ***square***: original classic drizzling kernel which maps each corner of the input pixel (scaled in size by the ***driz_sep_pixfrac*** parameter) to its position in the output image. • ***point***: a point so each input pixel can only contribute to the single pixel that is closest to the output position. It's equivalent to the limit *pixfrac- ->0* and is very fast. • ***gaussian***: this is a circular gaussian with FWHM equal to the value of *pixfrac*, measured in input pixels. • ***turbo***: this is similar to the square kernel, but the box is always the same shape and size on the output grid, and always aligned with the *X* and *Y* axes. This results in a significant increase in speed. • ***tophat***: the kernel is a circular "top hat" shape of width *pixfrac*. Flux from the input pixel gets distributed only to the output pixels within *pixfrac/2* of the output position. • ***lanczos3***: a Lanczos-style kernel extending 3 pixels from the center. The Lanczos kernel is a damped bounded form of the ***sinc*** interpolator and is very effective for resampling single images when drizzing is in the native scale and *pixfrac=1*. It leads to less resolution loss than the other kernels, and also less correlated noise in outputs. It is however much slower. *It should never be used for pixfrac != 1.0 and is not recommended for scale != native scale*. NOTE: The default value for this parameter is **turbo** since it is much faster than **square**, and it is quite satisfactory for the purposes of generating the median image.
driz_sep_wt_scl	exptime	Weighting factor for input data image If set to ***exptime***, the scaling value will be set equal to the exposure time found in the image header. This is the default recommended behavior. It is also possible to give ***driz_sep_wt_scl=expsq*** for weighting by the square of exposure time, which is optimal for read noise-dominated images. A final option would be to specify a specific numeric scaling factor (floating point value) that should be applied as-is when drizzling the images.
driz_sep_pixfrac	1.0	Linear size of drop in input pixels Fraction by which the input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, or *drop size* of a pixel in units of the input pixel size. (If *pixfrac* is set to less than 0.001, the kernel will be reset to point for more efficient processing.) For the "drizzle separate images" step, a default value of 1.0 is recommended in order to ensure that each output drizzled image is fully populated with pixels from the input image.
driz_sep_fillval	None	Value to be assigned to undefined output points Value assigned to output pixels that did not receive flux from any input pixels during drizzling. The default setting is ***None***. If the weight in both the input and output images for a given pixel are zero, the output pixel will be set to the value it would have had if the input had a non-zero weight. Otherwise, if a numerical value is provided (for example, 999), then these pixels will be set to that value.

Table 4.5: Parameters for astrodrizzle: "Drizzle Separate Images" (Continued)

Parameter	Default	Description
driz_sep_bits	0	Integer mask bit values considered good This parameter holds data quality (DQ) bit values of anomalous pixels in the `flt.fits` images that should be considered as good. All pixels flagged as good (bit value 0), and those with flags specified in this parameter, are used to create the single drizzle science image and define its associated weight images. If there are several flags that should be considered good, either the sum of their values or a comma-separated list of values can be entered.DQ flag values for different instruments are located in their Data Handbooks.

Table 4.6: Parameters for astrodrizzle: "Custom WCS for Separate Outputs"

Parameter	Default	Description
driz_sep_wcs	False	Define custom WCS for separate output images?
driz_sep_refimage	""	Reference image from which to obtain a WCS
driz_sep_rot	None	Position Angle of drizzled image's Y-axis w.r.t. North (degrees) Position angle of output image's *Y*-axis relative to North. A value of ***0.0*** orients the drizzled image with North at the top of the drizzled image. The default, ***INDEF***, specifies that the images will not be rotated, but drizzled in the camera's default orientation (where the drizzled image's *x* and *y* axes corresponding approximately to the detector axes). The ***INDEF*** option conserves disk space, making it the optimal choice for single drizzled images used in the intermediate step of creating a median image.
driz_sep_scale	None	Absolute size of output pixels in arcsec/pixel Linear pixel size in arcseconds/pixel for the separately drizzled images (that will be used in creating the median image for cosmic ray rejection). The default value of ***INDEF*** specifies that the undistorted pixel scale of the first input image will be used as the pixel scale for all the output images.
driz_sep_outnx	None	Size of separate output frame's *X*-axis (pixels) *S*ize, in pixels, of the *X* axis for the separately drizzled output images. If no value is specified, it will use the smallest size that can accommodate the full dithered field.
driz_sep_outny	None	Size of separate output frame's *Y*-axis (pixels) Size, in pixels, of the *Y* axis for the separately drizzled output images. If no value is specified, it will use the smallest size that can accommodate the full dithered field.
driz_sep_ra	None	Right Ascension output frame center in decimal degrees
driz_sep_dec	None	Declination output frame center in decimal degrees

4.2.5 Create a Median Image

Single-drizzled WCS-aligned images created from the previous step are combined to create a median image with statistical rejection of bad pixels from the image stack. It serves as an approximation of a combined distortion-free "clean" image with most cosmic rays and hot pixels removed. This median combination gets performed section-by-section from the input single-drizzled images. Each section corresponds to a contiguous set of lines from each image taking up no more than 1 Mb in memory, so that combining 10 input images would only require 10 Mb of memory for this step.

This median image will then be used in the next two steps for creating cosmic ray and bad pixel masks.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.7: Parameters for astrodrizzle: Create Median Image

Parameter	Default	Description
median	True	Create a median image? The median image, created using single-drizzled images from the previous step, will be used as a clean distortion-free "truth" image for cosmic ray rejection.
median_newmasks	True	Create new masks when doing the median? These masks are generated from weight files created in the previous step (drizzling each input image to create a single drizzled image, as well as a weight image based on the `flt.fits` DQ array). The bad pixels flagged in those mask files will be excluded when calculating the median. Generally this step should be set to ***True***, unless the user has reasons to want to include bad pixels in the median.
combine_maskpt	0.3	Percentage of weight image value below which it is flagged as a bad pixel Pixel values less than this value in the individual weight files will be flagged as a bad pixel in the median image's weight image.
combine_type	minmed	Type of combine operation This parameter allows the user to choose method for creating the median image. Valid options are: ***median***, ***mean***, ***minmed***, ***imedian***, ***imean***, ***and iminmed***. • The ***mean*** and ***median*** options set the calculation type when running *numcombine*, a numpy[1] method for median-combining arrays to create the median image. • The ***minmed*** option will produce an image that is generally the same as the ***median***, except in cases where the median is significantly higher than the minimum good pixel value. In this case, ***minmed*** will choose the minimum value. The sigma thresholds for this decision is provided by the ***combine_nsigma*** parameter. However, when a large number of images are being combined with this method, and many of the values for a pixel exceed the ***combine_nsigma*** threshold, ***minmed*** will pick the minimum value, biasing the pixel in that median image towards a low value. ***minmed*** is highly recommended for three images, and is good for four to six images, but should be avoided for ten or more images. • A value of ***median*** is the recommended method for a large number of images, and works equally well as ***minmed*** down to approximately four images. However, the user should set the ***combine_nhigh*** parameter to a value of ***1*** when using ***median*** with four images, and consider raising this parameter's value for larger numbers of images. When the number of values being considered for calculating a median value is even, the two inner values are averaged. This could bias the median value. Therefore, the user may want to have an odd number for the total number of images minus ***combine_nhigh***. • The options starting with ***i***, such as ***imedian***, works just like the normal median operation except when dealing with a pixel where all the values are flagged as "bad." In this case, the ***i*** functions to return the last pixel in the stack as if it were good. This will, for instance, prevent saturated pixels in the image from leaving holes in the middle of the stars.
combine_nsigma	4,3	Significance for accepting minimum instead of median When ***combine_type*** is ***minmed***, these are the sigma values used for accepting minimum values instead of median values. For the two values specified, the first value is used in the initial choice between selecting the median and minimum value for a pixel. Pixels adjacent to those flagged as cosmic rays are more prone to being affected by cosmic rays as well. This gets accounted for by specifying a second value for this parameter. This second value specifies a more stringent criteria for identifying a pixel as a cosmic ray and will be used to reject additional pixels around those identified in the first step. If only one value is specified, then it is used in both steps.

Table 4.7: Parameters for astrodrizzle: Create Median Image (Continued)

Parameter	Default	Description
combine_nlow	0	Number of low pixels to reject When ***combine_type*** is set to ***median***, this parameter sets the number of low value pixels to automatically reject during image combination.
combine_nhigh	0	Number of high pixels to reject When ***combine_type*** is set to ***median***, this parameter sets the number of high value pixels to automatically reject during image combination.
combine_lthresh	None	Lower threshold for clipping input pixel values Sets the lower threshold for clipping input pixel values from the generated median image which are more than this number of sigma below the median for each pixel. This floating-point value, when specified, gets passed directly to the **imcombine** function used by AstroDrizzle for use in creating the median image. The default setting of ***None*** specifies that no thresholds be used to clip the values used to create the median image.
combine_hthresh	None	Upper threshold for clipping input pixel values Sets the upper threshold in sigma for clipping input pixel values from the median image which are more than this number of sigma above the median value for each pixel. This floating-point value, when specified, gets passed directly to the **imcombine** function used by AstroDrizzle for use in creating the median image.The default setting of None specifies that no thresholds be used to clip the values used to create the median image.
combine_grow	1	Radius (pixels) for neighbor rejection Width in pixels for additional pixels to be rejected, in an image with a rejected pixel from one of the rejection algorithms.

1. NumPy is a Python extension that provides support for large, multi-dimensional arrays and matrices, including a large library of mathematical functions for operating on arrays.

4.2.6 Blot Median Image

The *blot* software takes a distortion-corrected image and applies (not removes) the full distortion model to recreate the original distorted input image. In other words, a drizzled image is "reverse-drizzled" to recreate the original distorted image.

In the previous step, a median image was created by combining the distortion-corrected single-drizzled images to generate an initial guess for the cosmic ray-cleaned combined output image. In this step, the median image is "blotted" to create clean versions of each input image at each of their respective dither locations.

This is done so that in the next step, these blotted images will be directly compared to their counterpart original distorted input images for detection of bad pixels, hot pixels, and cosmic rays to create bad pixel masks.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.8: Parameters for Astrodrizzle: "Blot Back the Median Image"

Parameter	Default	Description
blot	True	`Blot the median back to the input frame?` Perform the blot operation on the median image. The output will be median-smoothed images that match each input chip location. These images will be used in the cosmic ray rejection step.
blot_interp	poly5	Interpolant (nearest,linear,poly3,poly5,sinc) Type of interpolation function to use when blotting drizzled images back to their original WCS. Valid options are: • ***nearest***: Nearest neighbor • ***linear***: Bilinear interpolation in *x* and *y* • ***poly3***: Third order interior polynomial in *x* and *y* • ***poly5***: Fifth order interior polynomial in *x* and *y* • ***sinc***: Sinc interpolation; accurate but slow The ***poly5*** interpolation method has been chosen as the default because it is relatively fast and accurate compared to the other options. If ***sinc*** interpolation has been selected, then it will use the value of the parameter ***blot_sinscl*** to specify the size of the ***sinc*** interpolation kernel.
blot_sinscl	1.0	Scale for sinc interpolation kernel Size of the ***sinc*** interpolation kernel in pixels.
blot_addsky	True	Add sky using MDRIZSKY value from header? Restore a sky value (subtracted from the input image during the drizzle step) using the `MDRIZSKY` value from the header. If True, the ***blot_skyval*** parameter is ignored. This allows the output blot image to have the same background sky level as its counterpart original input image.
blot_skyval	0.0	Custom sky value to be added to blot image A single user-specified custom sky value to be added back to all the blotted images. This is only used if ***blot_addsky*** is set to ***False***.

4.2.7 Create a Cosmic Ray Mask for Each Image

In this step, the software compares each blotted image with its counterpart original `flt.fits` image to detect spurious pixels such as cosmic rays and hot pixels. The spurious pixels are flagged in cosmic ray masks that will be used in the final drizzle-combine step. Spurious pixels are identified by comparing the original input `flt.fits` image and its corresponding cleaned blot image, and blot derivative image. (A derivative image provides a measure of how sharp the edges of sources are in the image.)

The process of identifying cosmic rays and other bad pixels requires the following operations:

- Take the spatial derivative of each blotted image from the previous step. This derivative image is used to estimate the degree to which the value of the blotted estimate has been distorted by errors in the image offset (computed from the WCS of each input image), and the blurring effect of taking the median.

- Compare each original image with the corresponding blotted image. Where the difference is larger than what would be expected from noise statistics or an error in the shift, the suspect pixel is masked. The statistical limit gets set by the user through the first term of the ***driz_cr_snr*** parameter.
- Repeat the previous step on pixels adjacent to pixels already masked, using a more stringent comparison criterion specified by the second term of the ***driz_cr_snr*** parameter.

The *deriv* algorithm uses the blotted median image to compute the absolute value of the difference between each pixel and its four surrounding neighbors; for each pixel, the largest of these four values is then used by the *driz_cr* algorithm to flag cosmic rays and other blemishes, such as satellite trails. Where the difference is larger than can be explained by noise statistics, the flattening effect of taking the median, or an error in the shift (the latter two effects are estimated using the image derivative), the suspect pixel is masked.Cosmic rays are flagged using the following rule:

```
|data_image - blotted_image| > scale x deriv + SNR x noise
```

where *scale* is defined as the multiplicative factor applied to the derivative, *deriv.*

This expression is used to determine if the difference between the data image and the blotted image is large enough to require masking. *noise* is calculated using a combination of the detector read noise and the poisson noise of the blotted median image, plus the sky background.

The user must specify two cut-off signal-to-noise (SNR) values for determining whether a pixel should be masked: the first for detecting the primary cosmic ray, and the second for masking lower-level bad pixels adjacent to those found in the first pass.

Since cosmic rays often extend across several pixels, the adjacent pixels make use of a slightly lower SNR threshold. If desired, a third-pass cosmic ray rejection can be carried out by "growing" the cosmic rays via the ***driz_cr_grow*** parameter.

When ***driz_cr_corr*** is set to ***yes***, the task will create both a cosmic ray mask image (suffix `sci?_crmask.fits` where "?" is the extension number) and a clean version of the original input images (suffix `crclean.fits`), where flagged pixels are replaced by pixels from the blotted median. The cosmic ray masks are multiplied by the static pixel mask from the first step and masks created from pixels flagged as "bad" in the `flt.fits` DQ array, to create a final mask for each image. The optional parameter ***crbit*** allows the user to assign an alternate flag value to cosmic rays, and this flag will be written to the DQ array of each input image.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.9: Parameters for astrodrizzle: "Remove Cosmic Rays with deriv, driz_cr"

Parameter	Default	Description
driz_cr	True	Perform CR rejection with driz_cr? Perform cosmic ray detection? If set to ***True***, it will detect cosmic rays and create cosmic ray masks using the ***driz_cr*** algorithm.
driz_cr_corr	False	Create CR cleaned _cor file and a _crmask file? When set to ***True***, two images per input image are created: a cosmic ray-cleaned input image with suffix `_cor.fits`, generated directly from the input image, and a corresponding file with suffix `_crmask.fits` that documents the pixels detected affected by cosmic rays.
driz_cr_snr	3.5 3.0	Driz_cr.SNR parameter These values specify the signal-to-noise ratios used by this step for detecting cosmic rays.
driz_cr_grow	1	Driz_cr_grow parameter Radius (in pixels) around each detected cosmic ray to use more stringent detection criteria for additional cosmic ray detection.
driz_cr_ctegrow	0	Driz_cr_ctegrow parameter Length (in pixels) of CTE tail to mask in drizzled output.
driz_cr_scale	1.2 0.7	Driz_cr.scale parameter Scaling factor applied to the derivative in this step when detecting cosmic rays.

4.2.8 Create a Final Distortion-Free Combined Image

In the final step, the original input images are drizzle-combined, using the DQ, static, and cosmic ray masks to exclude bad pixels, hot pixels, and cosmic rays from the final image computation. The resulting drizzle-combined image is a registered, cosmic ray-cleaned, distortion-free, photometrically flat science image with associated weight and context images. By default, the output image will be written out as a single multi-extension FITS file, but the user could have them written out as separate simple FITS images.

The output frame, just like the single drizzle step, can be redefined using specific parameter settings; otherwise, default values will be used to include all the input pixels from all the WCS-registered input images.

Parameter Details

Text in Arial are supplemental comments to the descriptions in the **astrodrizzle** configuration file. Additional information is available in the **astrodrizzle** help file.

Table 4.10: Parameters for astrodrizzle: "Drizzle Final Combined Image"

Parameter	Default	Description
driz_combine	True	Perform final drizzle image combination? If ***True***, AstroDrizzle combines the input images, while applying the static and cosmic ray masks, to create a final cleaned, distortion-corrected image.

final_wht_type	EXP	Type of weighting for final drizzle
		Specify the type of weighting for the drizzle-combined science product. The options are:
		• ***EXP***: the final image is weighted by the input image exposure times. This option is provided as the default since it produces reliable weighting for all types of data, including older instruments (eg., WFPC2) where more sophisticated options may not be available. This weighting is a good approximation in the regime where the sky noise dominates.
		• ***ERR***: for ACS and STIS datasets, this option is generally recommended as the most accurate type of weighting for producing the final drizzle-combined image. The final drizzled image is weighted according to the inverse variance of each pixel in the input images, calculated from the data error array extension in each calibrated input image (`flt.fits` or `flc.fits`). This error array encapsulates all the noise sources in each exposure such as read noise, dark current, and sky background, as well as Poisson noise from the sources which includes a dependence upon exposure time as the Poisson noise increases with increasing counts. This option should not be used for WFPC2 because the `ERR` array is not produced during calibration.
		• ***IVM***: for this option, the user can either supply his or her own inverse-variance weighting map, or let AstroDrizzle automatically generate one during the final drizzle step. This may be necessary for specific purposes, for example, to create a drizzled weight file for software such as SExtractor that expects a weight image containing all background noise sources (sky level, read noise, dark current, etc.) but not the Poisson noise from the objects themselves.
		  • User-generated inverse variance images are specified at **astrodrizzle**'s input parameter using an ASCII file containing a list of input calibrated images (one per line), with a second column containing the name of the inverse variant map (`IVM`) file corresponding to each calibrated exposure. Each `IVM` file must have the same file format as the input file. If the calibrated images are multi-extension FITS files, then the `IVM` extension must be defined in the `EXTNAME` keyword as `IVM` with a separate `IVM` extension for each `SCI` extension in the science image.
		  • If no `IVM` files are specified on input, AstroDrizzle will rely on the flat-field reference file and computed dark value from the image header to automatically generate an `IVM` file specific to each exposure. This requires that the `PFLTFILE` (flat-field reference file) be accessible during processing.

final_kernel	square	Shape of kernel function Shape of the kernel used by Drizzle in the final image combination. The supported choices are: • ***square***: original classic drizzling kernel • ***point***: this kernel is a point so each input pixel can only contribute to the single pixel which is closest to the output position. It is equivalent to the limit pixfrac `-->` 0 and is very fast. • ***gaussian***: this kernel is a circular gaussian with FWHM equal to the value of pixfrac, measured in input pixels. • ***turbo***: this is similar to the square kernel but the box is always the same shape and size on the output grid and always aligned with the *X* and *Y* axes. This results in a significant speed increase in some cases. • ***tophat***: the kernel is a circular "top hat" shape of width *pixfrac*. In effect, only output pixels within *pixfrac/2* of the output position are affected. • ***lanczos3***: a Lanczos-style kernel extending three pixels from the center. The Lanczos kernel is a damped, bounded form of the sinc interpolator and is very effective for resampling single images when *scale* and *pixfrac* are set to 1. It leads to less resolution loss than the other kernels, and also less correlated noise in outputs. It is, however, much slower. It should never be used for *pixfrac != 1* and is not recommended for *scale != 1*. The default value for this parameter is ***square***.

Table 4.10: Parameters for astrodrizzle: "Drizzle Final Combined Image" (Continued)

Parameter	Default	Description
final_wt_scl	exptime	Weighting factor for input data image For the default value, ***exptime***, the scaling value is set equal to the exposure time found in the image header. It Is also possible to set this value to ***expsq*** for weighting by the square of exposure time which is optimal for read noise-dominated images.
final_pixfrac	1.0	Linear size of drop in input pixels Fraction by which the input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, or *drop size* of a pixel in units of the input pixel size. If the *pixfrac* is set to less than 0.001, the ***final_kernel*** parameter is reset to point for more efficient processing. If more than a few images are being combined, values smaller than 1 (e.g. 0.7 or 0.8) can be specified, which result in a slightly sharper output image. The ***final_pixfrac*** value has to be small enough to avoid degrading the final drizzle-combined image, but large enough that when all images are "dropped" onto the final frame, coverage of the output frame is fairly uniform. In general, ***final_pixfrac*** should be slightly larger than the final output scale to allow some "spillover" to adjacent pixels. This will help avoid "holes" in the final product when a given pixel has been flagged as "bad" in several frames. As a rule of thumb, statistics performed on the drizzled weight image in the region of interest should yield an RMS value (standard deviation) that is less than 20% of the median (midpoint) value. This threshold is a balance between the benefits of improving the image resolution at the expense of increasing noise in the background.
final_fillval	None	Value to be assigned to undefined output points Value assigned to output pixels that have zero weight or did not receive flux from any input pixels during drizzling. If the default value ***None*** is used and if the weight in both the input and output images for a given pixel are zero, the output pixel will be set to the value it would have had if the input had a non-zero weight. Otherwise, if a numerical value is provided (e.g. 999), then these pixels will be set to that value.
final_bits	0	Integer mask bit values considered good Integer sum of all the DQ bit values from the input image's DQ array that should be considered "good" when building the final weighting image. (If the parameter ***resetbits*** was not set, the bit flag for cosmic rays detected in a previous AstroDrizzle run, 4096 for ACS and WFPC2 data, can be set here.)
final_units	cps	Units for final drizzle image (counts or cps) This parameter determines the units of the final drizzle-combined image, and can either be ***counts*** or ***cps*** (count rate). The designated units are passed through to drizzle in the final drizzle step.

Table 4.11: Parameters for astrodrizzle: "Custom WCS For Final Output"

Parameter	Default	Description
final_wcs	False	Define custom WCS for final output image?
final_refimage	""	Reference image from which to obtain a WCS
final_rot	None	Position Angle of drizzled image's *Y*-axis w.r.t. North (degrees) The default of ***0.0*** (***None***) would orient the final output image so north is at the top of the image. A value of ***INDEF*** would specify that the images will not be rotated, but will instead be drizzled in the default orientation for the camera, with the *x* and *y* axes of the drizzled image corresponding approximately to the detector axes.

Parameter	Default	Description
final_scale	None	Absolute size of output pixels in arcsec/pixel Linear size of the output pixels in arcseconds/pixel for the final combined product. The default value of ***None*** specifies that the undistorted pixel scale for the first input image will be used as the pixel scale for the final output image.
final_outnx	None	Size of FINAL output frame *X*-axis (pixels) Size of the *X* axis of the final drizzled image (in pixels). The default, ***None***, will use the smallest size that can accommodate the full dithered field.
final_outny	None	Size of FINAL output frame *Y*-axis (pixels) Size of the *Y* axis of the final drizzled image (in pixels). The default, ***None***, will use the smallest size that can accommodate the full dithered field.
final_ra	None	Right Ascension output frame center in decimal degrees
final_dec	None	Declination output frame center in decimal degrees

4.2.9 Overriding Instrument-Specific Parameters

It is possible to override information in the image headers by setting these parameters directly so AstroDrizzle can work with data generated or modified by the user rather than working with data that came directly from the HST Archive.

Parameter Details

Table 4.12: Parameters for astrodrizzle: "Instrument Parameters"

Parameter	Default	Description
gain	""	Detector gain
gnkeyword	""	Detector gain keyword in header
rdnoise	""	Detector read noise
rnkeyword	""	Detector read noise keyword in header
exptime	""	Exposure time
expkeyword	""	Exposure time keyword in header

4.2.10 astrodrizzle Syntax in PyRAF and Python

A general overview of **drizzlepac** is available in Section 4.4. The examples below (following the "`-->`" PyRAF prompt) illustrate how to call **astrodrizzle** from PyRAF and Python.

Before **astrodrizzle** can be run, the **drizzlepac** package must be loaded. This only needs to be done once per session. The command is executed either in PyRAF or Python.

```
--> import drizzlepac
```

If the package needs to be reloaded,

```
--> reload(drizzlepac)
```

To run **astrodrizzle** using the TEAL GUI in PyRAF,

```
--> epar astrodrizzle
```

To run **astrodrizzle** as a command-line in PyRAF or Python (the parameter settings are for illustrative purposes):

```
--> from drizzlepac import astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits', output='final', wcskey='TWEAK', \
       driz_sep_bits='64,32', final_wcs=True, final_scale=0.05, final_rot=0)
```

The command above can also be run in PyRAF or Python with the parameter values written to a configuration file (suffix "`.cfg`") that is read by **astrodrizzle**,

```
--> from drizzlepac import astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits', configobj='myparam.cfg')
```

The help file can be accessed either from the TEAL GUI or at the command-line as such:

```
--> from drizzlepac import astrodrizzle
--> astrodrizzle.help()
```

To write the help contents to an ASCII file,

```
--> astrodrizzle.help(file='astrodriz_help.txt')
```

4.2.11 astrodrizzle Parameters

Listed below, as a refresher for experienced MultiDrizzle users, are the parameters for **astrodrizzle**. The values assigned to the parameters are task defaults that have been chosen to cover a wide range, but not all, data types.

While the parameters below are familiar, the format of that file is something new. It's called a configuration file, an ASCII file containing a list of parameters and their values that can be loaded into **astrodrizzle.** Any **drizzlepac** tasks run from the TEAL GUI can accept configuration files to set the value of task parameters, and parameter settings can be saved to uniquely-named configuration files.

Table 4.13: astrodrizzle Configuration File with Task Default Values

```
 _task_name_ = astrodrizzle#
input = *flt.fits# Input files (name, suffix, or @list)
output = ""# Rootname for output drizzled products
runfile = astrodrizzle.log# File for logging the script commands
updatewcs = False# Update the WCS keywords?
wcskey = ""# WCS version to use in processing
proc_unit = native# Units used during processing
coeffs = True# Use header-based distortion coefficients?
context = True# Create context image during final drizzle?
group = ""# Single extension or group to be combined/cleaned
build = False# Create multi-extension output file for final drizzle?
crbit = 4096# Bit value for CR ident. in DQ array
stepsize = 10# Step size for drizzle coordinate computation
resetbits = 4096# Bit values to reset in all input DQ arrays
num_cores = None# Max CPU cores to use (n<2 disables, None = auto-decide)
[STATE OF INPUT FILES]
preserve = True# Copy input files to archive directory, if not already archived?
overwrite = False# Copy input files into archive, overwriting if required?
clean = False# Delete temporary files after completion?
restore = False# Copy input files FROM archive directory for processing?
[STEP 1: STATIC MASK]
static = True# Create static bad-pixel mask from the data?
static_sig = 4.0# "Sigma*rms below mode to clip for static mask"
```

Table 4.13: astrodrizzle Configuration File with Task Default Values (Continued)

```
[STEP 2: SKY SUBTRACTION]
skysub = True# "Perform sky subtraction?"
skywidth = 0.3# "Bin width for sampling sky statistics (in sigma)"
skystat = median# "Sky correction statistics parameter"
skylower = None# "Lower limit of usable data for sky"
skyupper = None# "Upper limit of usable data for sky"
skyclip = 5# "Number of clipping iterations"
skylsigma = 4.0# Lower side clipping factor (in sigma)
skyusigma = 4.0# Upper side clipping factor (in sigma)
skyuser = ""# KEYWORD indicating a sky subtraction value if done by user.
skyfile = ""# Name of file with user-computed sky values.
[STEP 3: DRIZZLE SEPARATE IMAGES]
driz_separate = True# "Drizzle onto separate output images?"
driz_sep_kernel = turbo# Shape of kernel function
driz_sep_wt_scl = exptime# "Weighting factor for input data image"
driz_sep_pixfrac = 1.0# Linear size of drop in input pixels
driz_sep_fillval = None# Value to be assigned to undefined output points
driz_sep_bits = 0# Integer mask bit values considered good
```

Table 4.13: astrodrizzle Configuration File with Task Default Values (Continued)

```
[STEP 3a: CUSTOM WCS FOR SEPARATE OUTPUTS]
driz_sep_wcs = False# "Define custom WCS for separate output images?"
driz_sep_refimage = ""# Reference image from which to obtain a WCS
driz_sep_rot = None# "Position Angle of drizzled image's Y-axis w.r.t. North (degrees)"
driz_sep_scale = None# Absolute size of output pixels in arcsec/pixel
driz_sep_outnx = None# Size of separate output frame's X-axis (pixels)
driz_sep_outny = None# Size of separate output frame's Y-axis (pixels)
driz_sep_ra = None# right ascension output frame center in decimal degrees
driz_sep_dec = None# declination output frame center in decimal degrees
[STEP 4: CREATE MEDIAN IMAGE]
median = True# "Create a median image?"
median_newmasks = True# "Create new masks when doing the median?"
combine_maskpt = 0.3# "Percentage of weight image value below which it is flagged as a bad
pixel."
combine_type = minmed# "Type of combine operation"
combine_nsigma = 4 3# "Significance for accepting minimum instead of median"
combine_nlow = 0# "minmax: Number of low pixels to reject"
combine_nhigh = 0# "minmax: Number of high pixels to reject"
combine_lthresh = None# Lower threshold for clipping input pixel values
combine_hthresh = None# "Upper threshold for clipping input pixel values"
combine_grow = 1# Radius (pixels) for neighbor rejection
```

Table 4.13: astrodrizzle Configuration File with Task Default Values (Continued)

```
[STEP 5: BLOT BACK THE MEDIAN IMAGE]
blot = True# "Blot the median back to the input frame?"
blot_interp = poly5# Interpolant (nearest,linear,poly3,poly5,sinc)
blot_sinscl = 1.0# Scale for sinc interpolation kernel
blot_addsky = True# "Add sky using MDRIZSKY value from header?"
blot_skyval = 0.0# Custom sky value to be added to blot image
["STEP 6: REMOVE COSMIC RAYS WITH DERIV, DRIZ_CR"]
driz_cr = True# Perform CR rejection with deriv and driz_cr?
driz_cr_corr = False# "Create CR cleaned _cor file and a _crmask file?"
driz_cr_snr = 3.5 3.0# "Driz_cr.SNR parameter"
driz_cr_grow = 1# Driz_cr_grow parameter
driz_cr_ctegrow = 0# Driz_cr_ctegrow parameter
driz_cr_scale = 1.2 0.7# Driz_cr.scale parameter
[STEP 7: DRIZZLE FINAL COMBINED IMAGE]
driz_combine = True# "Perform final drizzle image combination?"
final_wht_type = EXP# Type of weighting for final drizzle
final_kernel = square# Shape of kernel function
final_wt_scl = exptime# Weighting factor for input data image
final_pixfrac = 1.0# Linear size of drop in input pixels
final_fillval = None# "Value to be assigned to undefined output points"
final_bits = 0# Integer mask bit values considered good
final_units = cps# Units for final drizzle image (counts or cps)
```

Table 4.13: astrodrizzle Configuration File with Task Default Values (Continued)

```
[STEP 7a: CUSTOM WCS FOR FINAL OUTPUT]
final_wcs = False# "Define custom WCS for final output image?"
final_refimage = ""# Reference image from which to obtain a WCS
final_rot = None# "Position Angle of drizzled image's Y-axis w.r.t. North (degrees)"
final_scale = None# Absolute size of output pixels in arcsec/pixel
final_outnx = None# "Size of FINAL output frame X-axis (pixels)"
final_outny = None# Size of FINAL output frame Y-axis (pixels)
final_ra = None# right ascension output frame center in decimal degrees
final_dec = None# declination output frame center in decimal degrees
[INSTRUMENT PARAMETERS]
gain = ""#
gnkeyword = ""#
rdnoise = ""#
rnkeyword = ""#
exptime = ""#
expkeyword = ""#
```

4.2.12 A Note about Photometry and Weights in AstroDrizzle

AstroDrizzle combines data using weights, a practice that may be new to some users who routinely add images together. Summing images works if the sky is stable, or if the objects' Poisson noise (not read noise) is the primary source of noise in the image; this is essentially equivalent to setting ***final_wht_type*** to ***EXP*** (exposure time weighting) in **astrodrizzle**.

However, for much of the work done with HST, sources of interest are faint compared to the sky. Images are not dominated by Poisson noise, but rather by read noise and sky noise. In this case, the inverse variance map (`IVM`) is a good weighting option because it weights each pixel by the sky, dark noise, and read noise, adjusting appropriately for the value of the flat field.

But then, why not do a full calculation of the noise in each pixel in each image instead? This can be done by setting ***final_wht_type*** to ***ERR*** in **astrodrizzle**. This weight is what statisticians call the "minimum variance estimator." It has the smallest statistical error of all choices of weights. However, it is not an "unbiased minimum variance estimator." If a pixel in an image has, by chance, a few lesser counts than one would expect on average in a particular pixel, then calculating the noise for that pixel based on the number of counts will underestimate the noise and thus overestimate the significance of that pixel. The inverse problem occurs if the pixel happens to be a bit brighter than average. As a result, using this weight produces a small bias (usually no more than one to two percent in HST images). Sources in the resulting final image will be slightly fainter than they really are.

In general, photometric software available to most users does not take advantage of the final weight map. Some users may wish to write their own code to use the information in the weights. In this case, there are several choices:

- If exposure time weighting (***EXP***) was used, the final image could be multiplied by the weight image to get the number of source counts in each pixel to estimate photon noise.
- For ***IVM*** weighting, and when there's no concern for very bright sources in the field, the final `IVM` output weight map could be used.
- If the ***ERR*** array was used, this already provides a very good estimate of the noise (but not the bias).

For users who want an accurate weight map for use in photometry, a good approach may be to first run **astrodrizzle** using ***final_wht_type*** set to either ***EXP*** (when bright sources are most important) or ***IVM*** (if the faint sources are the primary objects of interest). This creates the final science image but not the final weight map. To create a final weight map, **astrodrizzle** needs to be run again, using the same weighting scheme as in the first drizzle; however, the original weight maps (`drz_wht.fits` images) should be drizzled rather than the images. This will give an output "image" that fully estimates the error at each pixel. At present, there's no easy way for the user to do this. Perhaps this could be done by swapping in the `flt.fits` `ERR` array for the science (`SCI`) array, then running **astrodrizzle** on the modified images. If there is

demand for this approach, it may be implemented as a feature in a future version of AstroDrizzle.

4.2.13 AstroDrizzle Memory Usage

The AstroDrizzle software will estimate the memory requirements for processing a particular set of data during the "Initialization Step" and print a summary to the screen.

The reported requirements include the size of the final output image in pixels, the number of cores to be used, and an estimate of the amount of memory required for processing. AstroDrizzle will estimate the maximum amount of memory required during processing based on the size of the input files, the number of cores to be used, and the size of the final output products. This information can be used by the user to determine whether or not their processing run will require more memory or disk space than installed on their system, allowing them to interrupt the processing as soon as possible using `Ctrl-C` so that input parameters can be reset as needed.

AstroDrizzle requires approximately the same processing time as MultiDrizzle. However, to improve speed a new **astrodrizzle** parameter called ***num_cores*** will enable parallel processing using multiple cores, greatly reducing processing time. This has been enabled for the ***driz_separate*** and ***driz_cr*** steps. Memory usage will be a prime concern for those who rely on this parallel processing feature as each core will require a separate copy of the output array to be stored in memory along with having each input image in memory. For example, processing four WFC3/UVIS images to generate a 5000 x 5000 mosaic on a quad-core system will require over 1.3Gb of memory during the single drizzle step. Users with little RAM are, therefore, strongly advised to run with ***num_cores=1***.

An example of the processing time is shown in Figure 4.1 for three WFC3/UVIS images that were combined using **astrodrizzle** with default parameters. The total processing time using 6 cores was only 3.6 minutes. When num_cores was set to 1, the actual memory usage was lower, but the total processing time increased to 6.3 minutes, compared to the run time for MultiDrizzle (5.5 minutes) for the same dataset using one core.

Figure 4.1: Processing Time for Three WFC3/UVIS Images using One and Six Cores

```
Timing: Mdrizz
   Step                Elapsed time
1  Initialize:         38.3311240673 sec.
2  Build static mask:  18.2397210598 sec.
3  Subtract sky:       24.6707360744 sec.
4  Drizzle separate:   48.0001471043 sec.
5  Median/sum/ave.:    47.7558379173 sec.
6  Blot:               44.1800470352 sec.
7  DrizCR:             34.1886100769 sec.
8  Final drizzle:      74.3060610294 sec.
   Total               329.672284365sec. = 5.5 minutes

Timing: Adrizz (1 core)
Step                          Elapsed time
--------------------          --------------------
          Initialization         21.6343 sec.
            Static Mask          5.7392 sec.
           Subtract Sky          13.2276 sec.
        Separate Drizzle         165.3693 sec.
          Create Median          32.1671 sec.
                   Blot          36.2507 sec.
                Driz_CR          28.2185 sec.
           Final Drizzle         76.9802 sec.
    ====================          ====================
                  Total          379.5868 sec.  = 6.3 minutes

Timing: AD (6 out of 24 cores)
   --------------------          --------------------
         Initialization          15.0582 sec.
            Static Mask          3.7108 sec.
           Subtract Sky          13.3153 sec.
       Separate Drizzle          21.4353 sec.
          Create Median          32.5870 sec.
                   Blot          36.6024 sec.
                Driz_CR          18.6143 sec.
           Final Drizzle         74.1446 sec.
   ====================          ====================
                  Total          215.4680 sec.=  3.6 minutes
```

4.3 AstroDrizzle in the Pipeline

Only ACS and WFC3 data are currently being processed using On-the-Fly Calibration (OTFR) with AstroDrizzle in the pipeline. Active imaging instruments that are AstroDrizzle-compatible are WFC3, ACS, and STIS. Drizzled NICMOS and WFPC2 data are stored in a Static Archive as MultiDrizzle products along with their flat-field calibrated files (flt.fits and c0m.fits, respectively). However, they are also compatible with AstroDrizzle for anyone who wishes to reprocess those images.

In the pipeline, data undergoes standard calibration (to produce flat-field calibrated files (`flt.fits`, and `flc.fits` for CTE-corrected ACS data). AstroDrizzle processing starts with the name of a single exposure, or the name of an association file containing the names of several exposures to be combined. A single exposure will generate an output file with the same rootname but with the suffix `drz.fits`. When an association file is provided as input to AstroDrizzle, all `flt.fits` images belonging to that association are drizzle-combined to create a product with the same rootname (with suffix `drz.fits`) as the association file. The AstroDrizzle products from the Archive, listed below with their respective file suffix, are:

- `drz.fits`: These are the default drizzle multi-extension FITS products, created using `flt.fits` input files. This type of drizzled image has extension types `SCI` (for the science image extension), `WHT` (the weight image extension), and `CTX` (context image extension) in the final combined image
- `drc.fits`: These drizzled multi-extension FITS products are generated from ACS CTE-corrected calibrated data with the suffix `flc.fits`.
- `.log`: AstroDrizzle generates an ASCII file containing a log of all messages reported during processing; this file gets used as the basis for the final products trailer file during pipeline processing.

As part of pipeline processing, distortion information are integrated into the calibrated data (`flt.fits` and `flc.fits` for CTE-corrected ACS/WFC) from a set of reference files, and WCS information is then recomputed using this distortion information.

AstroDrizzle parameter settings for pipeline use are defined for different observing configurations in a reference table named by the header keyword `MDRIZTAB`[3]. Drizzle-combined data from the pipeline should generally be regarded as quick-look

3. Default values in the **astrodrizzle** task serve as a starting point for reprocessing data; they're not the same as those in the `MDRIZTAB` reference file.

products, and users are strongly encouraged to reprocess their images using AstroDrizzle to see if the quality of data can be improved using different parameter values. Information on how to determine the quality of Archive drizzled products is available in Chapter 8.

How does AstroDrizzle know which images to combine in the pipeline? The association file contains images that were taken as a dither pattern, using the `POS TARG` special requirement, or repeated exposures in the form of sub-exposures or `CR-SPLIT` observations. These observations types, taken in the same visit with the same guide star pairs, almost always have highly accurate offset values in the image header that could be used to align the images. (Exceptions, of course, could be due to a loss of lock on a guide star midway through the visit or other telescope pointing anomalies. So it is always useful to check the image quality for signs of anomalies.)

MDRIZTAB Reference File

The `MDRIZTAB` reference table contains AstroDrizzle task parameters optimized for a wide range of observations. AstroDrizzle uses this table to match the best parameter values with the type of observations being processed. Each instrument detector has its own `MDRIZTAB` reference table; in it, each row has AstroDrizzle task parameter settings optimized for the filters used and three ranges of input image numbers per association (one image, 2 to 6 images, and more than 6 images).

To understand the processing that takes place in the pipeline, it can be helpful to inspect the `MDRIZTAB` reference file; it can be identified by the image header keyword `MDRIZTAB`. This reference file specifies the values for most AstroDrizzle task parameters, and uses software default values for the rest of the parameters.

Table 4.16 shows an example of the `MDRIZTAB` contents for four types of observations. But first, outlined below are two summary lists of **astrodrizzle** parameters; the first list shows parameters with values that are the same for all observations processed in the pipeline, and therefore, those parameters do not need to be in the `MDRIZTAB` file. The second list shows parameter values that may have different values based on the type of observations.

Table 4.14: astrodrizzle Parameters With the Same Values During Pipeline Processing for All Observations

mdriztab	Set to **True** so **astrodrizzle** in the pipeline always uses the `MDRIZTAB` reference file for populating its parameters
stepsize	Set to **10**; distortion corrections are applied directly to every 10th pixel, and those in-between are interpolated. Tests have found that the interpolation values are highly accurate for a step size of 10.
output	Set to blank since input files rootnames are used for naming the output product
updatewcs	Set to **True** to always update the headers with the latest distortion information
resetbits	Set to **4096** so that all pixels flagged as cosmic rays by calibration software that was run before AstroDrizzle processing are treated as good pixels.
preserve:	Set to **False**. This parameter makes backups of the input files—it is useful for reprocessing but not needed in the pipeline.

Table 4.15: astrodrizzle Parameters that Appear in the `MDRIZTAB` Table

General parameters:	mdriztab, refimage, runfile, coeffs, context, clean, group, ra, dec, build, shiftfile
Keyword definition parameters	gain, gnkeyword, readnoise, rnkeyword, exptime, expkeyword, crbitval
Static Mask parameters:	static, staticfile, static_sig
Sky Subtraction parameters	subsky, skywidth, skystat, skylower, skyupper, skyclip, skylsigma, skyusigma
Separate Drizzle parameters:	driz_separate, driz_sep_outnx, driz_sep_outny, driz_sep_kernel, driz_sep_scale, driz_sep_pixfrac, driz_sep_rot, driz_sep_fillval, driz_sep_bits
Median Step parameters:	median, median_newmasks, combine_type, combine_nsigma, combine_nlow, combine_nhigh, combine_lthresh, combine_hthresh, combine_grow
Blot Step parameters:	blot, blot_interp, blot_sinscl
Driz_CR Step parameters	driz_cr, driz_cr_corr,'driz_cr_snr, driz_cr_scale, driz_combine'
Final Drizzle parameters	driz_combine, final_outnx, final_outny, final_kernel, final_scale, final_pixfrac, final_rot, final_fillval, final_bits

These parameter values depend on two primary conditions: the number of input exposures to be combined and the filters used for the exposures.

Many parameters use the task default values. Some parameter values are determined by specific image characteristics. For instance, sky subtraction is turned off for narrow-band and ramp filters because the image sky background is usually low. Associations with more than six images are drizzled to a finer scale and "pixfrac" to increase image resolution (more about that in Chapter 6).

For ACS data, the MDRIZTAB reference file specifies "final_bits=96" so that astrodrizzle in the pipeline treats any pixel flagged with bits 32 and 64 in the data quality array of the flat-field calibrated (flt.fits) files as good pixels. These flags are set by calacs and correspond to CTE tails of hot pixels in superdark DQ arrays and warm pixels in superdark DQ arrays.

Table 4.16 shows `MDRIZTAB` reference file contents for four different types of WFC observations. Four rows corresponding to those observation types were

extracted from the MDRIZTAB table, and transposed for easier viewing. Here, the spectral element ***ANY*** refers to all wide-band filters.

Table 4.16: Excerpt of MDRIZTAB for Four Types of Observing Scenarios

Parameter	Units	Wide band, one image	Wide band, two images	Wide band, six image	F658N + CLEAR2L, two images
numimages	none	1	2	6	2
mdriztab	none	yes	yes	yes	yes
refimage	none	“”	“”	“”	“”
runfile	none	“”	“”	“”	“”
coeffs	none	header	header	header	header
context	none	yes	yes	yes	yes
clean	none	yes	yes	yes	yes
group	none	“”	“”	“”	“”
ra	degrees	INDEF	INDEF	INDEF	INDEF
dec	degrees	INDEF	INDEF	INDEF	INDEF
build	none	yes	yes	yes	yes
gain	none	“”	“”	“”	“”
gnkeyword	none	***1***	***1***	***1***	***1***
readnoise	none	“”	“”	“”	“”
rnkeyword	none	***2***	***2***	***2***	***2***
exptime	seconds	INDEF	INDEF	INDEF	INDEF
expkeyword	none	EXPTIME	EXPTIME	EXPTIME	EXPTIME
crbitval	none	4096	4096	4096	4096
shiftfile	none	“”	“”	“”	“”
static	none	no	yes	yes	yes
staticfile	none	“”	“”	“”	“”
static_sig	“”	4.	4.	4.	4.
subsky	none	yes	yes	yes	no
skywidth	none	0.1	0.1	0.1	0.1
skystat	none	median	median	median	median
skylower	electrons	-100.	-100.	-100.	-100.
skyupper	electrons	INDEF	INDEF	INDEF	INDEF

Table 4.16: Excerpt of MDRIZTAB for Four Types of Observing Scenarios (Continued)

skyclip	none	5	5	5	5
skylsigma	none	4.	4.	4.	4.
skyusigma	none	4.	4.	4.	4.
driz_separate	none	no	yes	yes	yes
driz_sep_outnx	pixels	INDEF	INDEF	INDEF	INDEF
driz_sep_outny	pixels	INDEF	INDEF	INDEF	INDEF
driz_sep_kernel	none	turbo	turbo	turbo	turbo
driz_sep_scale	none	INDEF	INDEF	INDEF	INDEF
driz_sep_pixfrac	none	1.	1.	1.	1.
driz_sep_rot	degrees	INDEF	INDEF	INDEF	INDEF
driz_sep_fillval	none	INDEF	INDEF	INDEF	INDEF
driz_sep_bits	none	96	96	96	96
median	none	no	yes	yes	yes
median_newmasks	none	no	yes	yes	yes
combine_type	none	minmed	minmed	median	minmed
combine_nsigma	none	“4 3”	“4 3”	“4 3”	“4 3”
combine_nlow	none	0	0	0	0
combine_nhigh	none	1	1	1	1
combine_lthresh	none	INDEF	INDEF	INDEF	INDEF
combine_hthresh	none	INDEF	INDEF	INDEF	INDEF
combine_grow	pixels	1	1	1	1
blot	none	no	yes	yes	yes
driz_cr	none	no	yes	yes	yes
driz_cr_corr	none	no	no	no	no
driz_cr_snr	none	“3.5 3.0”	“3.5 3.0”	“3.5 3.0”	“3.5 3.0”
driz_cr_scale	none	“1.2 0.7”	“1.2 0.7”	“1.2 0.7”	“1.2 0.7”
driz_combine	none	yes	yes	yes	yes
final_outnx	pixels	INDEF	INDEF	INDEF	INDEF
final_outny	pixels	INDEF	INDEF	INDEF	INDEF
final_kernel	none	square	square	square	square
final_kernel	none	square	square	square	square
final_scale	arcseconds	INDEF	INDEF	INDEF	INDEF

Table 4.16: Excerpt of MDRIZTAB for Four Types of Observing Scenarios (Continued)

final_pixfrac	none	1.	1.	0.8	1.
final_rot	degrees	INDEF	INDEF	INDEF	INDEF
final_fillval	none	INDEF	INDEF	INDEF	INDEF
final_bits	none	96	96	96	96
blot_interp	""	poly5	poly5	poly5	poly5
blot_sinscl	pixels	1.	1.	1.	1.

1. value is "ATODGNA,ATODGNB,ATODGNC,ATODGND"
2. value is "READNSEA,READNSEB,READNSEC,READNSED"

4.4 The DrizzlePac Package

drizzlepac is a suite of tasks that allow users to align HST images, combine them, and perform coordinate transformations on source positions.

AstroDrizzle uses image WCS information to align the images. For the most part, this works well for images taken in the same orbit. Images in one visit over several orbits usually align well because the same guide star pair is used, but they should be carefully inspected to check for very small offsets due to guide star re-aquisition anomalies.

Images from different visits, however, cannot be aligned based on WCS alone because guide star catalog positions have uncertainties as high as 0.3 to 0.5 arcseconds. Therefore, the **tweakreg** task was developed to fine-tune the alignments, using point sources common to each image to determine *x* and **y** offsets, and rotations. Other tasks in **drizzlepac** complement **astrodrizzle** and **tweakreg**, providing functions that would be useful, such as coordinate transformations. All these tasks rely on the same distortion information required by AstroDrizzle, and also depend on other tasks from the **stwcs** and **fitsblender** packages (described later) to perform many of their operations.

Detailed information is available in the help files for these task. Help can be accessed either from the TEAL GUI or by Python commands. These interfaces are documented in Chapter 5.

4.4.1 DrizzlePac tasks

A brief description of the tasks in **drizzlepac**:

astrodrizzle	Primary task for combining images, removing cosmic rays, and removing distortion
tweakreg	Computes offsets in WCS between images, and a reference image or reference frame
imagefindpars	"Sub-task" or "pset" containing parameters to find point sources used by **tweakreg** to build source catalogs for each **tweakreg** input image
tweakback	Apply an updated WCS solution created by **tweakreg** for a drizzled image to the constituent distorted (`flt.fits`) images
pixtopix	Convert pixel positions from an input image to pixel positions in an output WCS or image
pixtosky	Convert pixel positions from an input image to sky coordinates with full distortion correction as appropriate
skytopix	Convert sky positions to pixel positions in an image
resetbits	"Sub-task" to reset specified `flt.fits` data quality (DQ) values to 0
updatenpol	Add the names of the new ACS distortion reference files `NPOLFILE` and `D2IMFILE`, then update images to include residual distortion corrections as image extensions.
blendheaders	Merge the keywords from all input images used to create a drizzled product into a single output header with a table extension using rules defined for each instrument. A default set of rules have been developed for pipeline use for ACS and WFC3, with offline support of STIS, NICMOS and WFPC2 data being provided by very basic rules.

Detailed information on how to run these tasks are available in their respective help files. Tasks that handle coordinate transformations–**pixtopix**, **pixtosky**, **skytopix**–are relatively straightforward and won't be covered in this document (please consult the help file).

Data retrieved from the Archive before AstroDrizzle was implemented in the pipeline in June 2012 (check the DrizzlePac website for the exact date) have to be updated for compatibility with **drizzlepac** tasks. WFC3 and ACS images should be processed with the **updatewcs** task from the **stwcs** package to update SIP and other distortion keywords. ACS images should also be processed with the **updatenpol** task to insert FITS extensions containing information about residual distortions.

tweakreg (which calls the **imagefindpars** sub-task) is used for aligning images. Please see Section 4.4.2 for a description of the task.

tweakback is used for additional image alignment applications; when **tweakreg** has been used to align drizzled products (i.e., different filters, pointings, or detectors), the **tweakback** task can be used to propagate the updated WCS back to the original `flt.fits` images. **astrodrizzle** can then be used to process those updated `flt.fits`. Note that **tweakback** only aligns the WCS in the image headers. To ensure that the actual pixels are aligned in the final drizzled product, a reference output frame must be defined, as described in Section 7.5.

blendheaders, a task that's also run in the pipeline for ACS and WFC3 data, collects important keyword information from AstroDrizzle input images for inclusion in the final drizzle-combined image.

4.4.2 Aligning Images with TweakReg and ImageFindPars

Coverage of tweakreg and imagefindpars in this section is focused on describing the tasks and parameters. Examples of how to use tweakreg are available in Chapter 7and the DrizzlePac website**.**

Overview of the TweakReg Software

TweakReg provides an automated interface for computing residual shifts between images before they're combined by AstroDrizzle. It is especially useful for combining images taken in different visits.

In the Dither package that included MultiDrizzle, the **tweakshifts** task required distortion-free images as input, making it necessary to run **multidrizzle** to generate single-drizzled distortion-corrected images. TweakReg, however, supports the use of flat-field calibrated distorted images (`flt.fits` for ACS, STIS, and WFC3, `flc.fits` for ACS, and `c0m.fits` for WFPC2) as input images.

WCS information for each image is related to the guide star pairs used during telescope guiding. For images taken in the same visit and orbit, the alignment between images are generally very accurate, to about two to five milliarcseconds. For images in the same visit but across several orbit, guide star reaquisition could potentially introduce very small offsets between five to 20 milliarcseconds. However, for images taken at different visits using different guide stars, residual offsets that remain after the images are aligned based on their WCS information are due to uncertainties in the guide star catalog positions, which can be as large as 0.3 to 0.5 arcseconds. For more information about HST pointing stability, please see Appendix B.

The source-finding algorithm provided within TweakReg has been optimized for point sources. As a result, this task may not work optimally for fields containing primarily extended sources or even images dominated by cosmic rays. In addition, its reliance on catalog matching to determine offsets requires a large enough overlap between images and a large enough sample of valid sources in the overlap to obtain a good solution, making the use of this task on sparse fields or images with small overlap more problematic

TweakReg can be used to align sets of images if there are enough point sources to make reliable matches, in the following way.

1. Using flat-field calibrated images (`flt.fits` images[4]; `flc.fits` for CTE-corrected images for ACS; `c0m.fits` for WFPC2) as input, TweakReg can generate source catalogs for each image using an algorithm similar to **daofind**. Exclusion files, in the form of **ds9** regions files or simple target-radius specifications, can be used to instruct TweakReg to avoid certain parts of the image for source detection. The software also accepts user-provided catalogs for each input image.
2. For each image, its distortion model is used to correct source positions. (Therefore, it is unnecessary to drizzle the images before obtaining source positions, as was the case for MultiDrizzle.)
3. The offsets between each image and a reference frame are determined using a catalog-matching algorithm similar to **xyxymatch**. This reference frame could be an image in the input list (by default, the first image), a user-specified reference image, or undistorted sky coordinates (R.A., Dec.) and fluxes provided by the user. Matching sources between images can be severely exacerbated by cosmic rays, especially for long exposures. **imagefindpars** (called by **tweakreg**) parameters may be adjusted to impose flux detection ranges to exclude likely cosmic ray events. For pre-determined catalogs with magnitude values, potential matches that don't fall in a reasonable magnitude range could also be discarded as a false match. Alternately, detections can also be performed on cosmic ray-cleaned images.
4. With the culling of false detections over several iterations of parameter adjustments, TweakReg should be able to converge on an acceptable RMS for the offset solution fits. Plots showing the quality of the fits can be displayed for each reference-image offset solution.

When the user is satisfied with the result, TweakReg is run one last time with those final parameters to update the WCS information in the images to a common reference frame, making it ready for AstroDrizzle processing.

tweakreg Task Syntax in PyRAF and Python

As a reminder, the examples below (following the "`-->`" PyRAF prompt) illustrate how to call **tweakreg** from PyRAF and Python. Chapter 5 has a more in-depth guide to using **drizzlepac** in the PyRAF and Python interfaces.

4. With AstroDrizzle installed in the pipeline, `flt.fits and flc.fits` images from the Archive now have updated WCS information to make it compatible with **drizzlepac** tasks. AstroDrizzle has been written to operate on WFPC2 images processed using MultiDrizzle from the Static Archive.

Before **tweakreg** is called, the **drizzlepac** package must be loaded. This only needs to be done once per session. The command can be executed either in PyRAF or Python:

```
--> import drizzlepac
```

If the package needs to be reloaded,

```
--> reload(drizzlepac)
```

To run **tweakreg** using the TEAL GUI,

```
--> epar tweakreg
```

To run **tweakreg** as a command-line in PyRAF or Python (the parameter settings are for illustrative purposes):

```
--> from drizzlepac import tweakreg
--> tweakreg.TweakReg('*flt.fits',updatehdr=False,fwhmpsf=3.5,threshold=200,\
shiftfile=True,outshifts='shift.txt')
```

The command above can also be run in PyRAF or Python with the parameter values written to a configuration file (suffix `.cfg`) that is read by **tweakreg**,

```
-->from drizzlepac import tweakreg
-->tweakreg.TweakReg('*flt.fits',configobj='myparam.cfg')
```

The help file can be accessed either from the TEAL GUI or at the command-line as such:

```
-->from drizzlepac import tweakreg
-->tweakreg.help()
```

To write the help contents to an ASCII file,

```
-->tweakreg.help(file='help.txt')
```

tweakreg Parameter Details

Text in Arial are supplemental comments to the descriptions in the **tweakreg** configuration file. Additional information is available in the **tweakreg** help file.

Table 4.17: tweakreg Parameters: Set-upand Initialization

Parameter	Default	Description
input	*flt.fits	Input files (name, suffix, or @list) These are images that need to be aligned with respect to a reference image. • Name of a single image • Name of an association table • Images can be specified with a wildcard, as shown in the default parameter setting • Alist of images in an ASCII file, with prefix "@" before its filename(e.g., `@filelist`) • Image names separated by a comma
refimage	""	Filename of reference image The reference image from which reference catalog sources are derived.A name or catalog may be specified for this field. By default, the reference image is the first image on the list. .In some situation, the image with the largest area of overlap with other images in the set should be selected as the reference.
exclusions	""	Filename for source exclusions catalogs A set of files containing image regions to avoid when finding sources. This file MUST have one line for each input image, with the name of the input file in the first column. Subsequent columns would be used to specify an exclusions file for each chip, in order, in the science extension. If a chip does not require an exclusions file, the string ***None*** or ***INDEF*** can be entered for that chip. Each chip's exclusion file should conform to the **ds9**-generated "region" file format; currently, only "circle()" regions are interpreted and used, and only with units for the positions of the exclusions center of "fk[4|5]" and "image". Support for additional region file options will be added at a later date.
updatewcs	False	Update WCS keywords of input images? A user will typically run **tweakreg** several times till a satisfactory RMS to the fit solution is obtained. Then, **tweakreg** is run with those good settings again with the parameter value ***True*** to record new WCS information to the image header.
writecat	True	Write out source catalogs? Specify if the source catalogs for each image, created by the **tweakreg** source detection algorithm, should be saved.
clean	False	Remove intermediate files? Specify whether or not to remove the temporary files created by **tweakreg**, including any catalog files generated for the shift determination. The default setting is ***False*** because these files could come in useful for tracking down potential problems.
verbose	False	Print extra messages during processing?
runfile	tweakreg.log	Filename of processing log Comments displaying the execution progress of **tweakreg** are saved to a file; this is useful to track down potential problems.

Table 4.18: tweakreg Parameters: "Update Header"

Parameter	Default	Description
updatehdr	False	Update headers of input files with shifts? Specify whether or not to update the headers of each input image directly with the shifts that were determined. This will allow the input images to be combined by **astrodrizzle** without having to provide the shift file as was required for **multidrizzle**, since the shifts have already been applied to each input image's WCS.
wcsname	TWEAK	Name of updated WCS Each WCS has a unique name assigned to it. The default value used here is ***TWEAK*** but it can be any name specified by the user.

Table 4.19: tweakreg Parameters: "Headerlet Creation"

Parameter	Default	Description
headerlet	False	Create headerlet for solution? Specify whether or not to generate *headerlets* at the end of the **tweakreg** run? If ***True***, headerlets are created for the images, regardless of the value of the ***updatehdr*** parameter.
attach	True	Create headerlet FITS extension? If a headerlet is created, specify if the new headerlet should be incorporated into the input image as a new extension.
hdrfile	""	Filename for headerlet FITS file Name of the separately-written headerlet file. If a name specified for this parameter does not have the suffix `.fits`, the headerlet will be named after the rootname of the input image, the string entered for this parameter, followed by the suffix ***_hlet.fits***. (For example, if the image rootname is `j99da1f2q_flt.fits`, and if ***hdrfile*** has the value ***hdrlet1***, the FITS headerlet will be named `j99da1f2q_hdrlet1_hlet.fits`.
clobber	False	Overwrite existing headerlet FITS file? If a headerlet with the same name specified by the `hdrfile` parameter already exists, specify whether the old file should, or should not, be overwritten.
hdrname	""	Unique name (`HDRNAME`) for headerlet This parameter value is a unique name for the headerlet solution. It identifies the specific WCS alignment solution contained in the headerlet.
author	""	Author name for creator of headerlet Name or initials of the person who created the headerlet.
descrip	""	Short description of headerlet solution A one-line description of the WCS alignment contained in the headerlet.
catalog	""	Name of catalog used for headerlet solution Name of reference catalog used as the basis for the image alignment.
history	""	Name of ASCII file containing history for headerlet Name of an ASCII file containing details regarding the history of the headerlet WCS solution. This can include information on the catalog used for the alignment and processing notes on finalizing the headerlet WCS alignment. This information will be reformatted by TweakReg as a 70-character wide FITS `HISTORY` keyword section.

Table 4.20: tweakreg Parameters: "Optional Shift File Output"

Parameter	Default	Description
shiftfile	False	Create output shift file? This shift file is only generated for record-keeping. It has the same format as the old **multidrizzle** shift files.
outshifts	shifts.txt	Filename of generated shift file Name for the output shift file created by TweakReg. This shift file was formatted for use as direct input to **mulitdrizzle**, but is not used by **astrodrizzle**. This file simply serves as a human-readable summary of the shifts found by TweakReg.
outwcs	shifts_wcs.fits	Filename of shift file reference WCS Filename for the OUTPUT reference WCS file created by TweakReg. This is the reference WCS from which shifts get measured, and could be used by the now depreciated MulitDrizzle to interpret those shifts. The reference WCS file is a FITS file containing only the WCS keywords in a Primary header with no image data. The values will be derived by default from the first input image or, if given, a user-specified reference image.

Table 4.21: tweakreg Parameters: "Coordinate File Description"

Parameter	Default	Description

catfile	""	File containing coordinate filenames for input files
		Name of the file that contains a list of input images and associated catalog files generated by the user. Each line of this file will contain the name of an input image in the first column. The remaining columns will provide the names of the source catalogs for each chip in order of the science extension numbers ([SCI,1], [SCI,2], etc. ....). A sample ***catfile***, with one line per image would look like:
		`image1_flt.fts cat1_sci1.coo cat1_sci2.coo`
		`image2_flt.fts cat2_sci1.coo cat2_sci2.coo`

xcol	1	Column name(s) for X positions Column number of *X* position from the user-generated catalog files specified by the parameter ***catfile***.
ycol	2	Column name(s) for Y positions Column number of *Y* position from the user-generated catalog files specified by the parameter ***catfile***.
fluxcol	""	Column name for source flux/mag values Column number for flux values in the user-generated catalog files specified by the parameter ***catfile***. These values are only used if a flux limit has been specified by the user using the ***fluxmax*** or ***fluxmin*** parameters.
fluxmax	None	Maximum flux value for valid objects This value, if specified, is the upper limit in the range of source fluxes that will be kept for each image's catalog of sources. Those sources will then be matched with sources from the reference catalog for the final fit. For the default entry, ***None***, there is no upper limit in the range of fluxes for valid sources.
fluxmin	None	Minimum flux value for valid objects This value, if specified, is the lower limit in the range of source fluxes that will be kept for each image's catalog of sources. Those sources will then be matched with sources from the reference catalog for the final fit. If the value is set to ***None***, all objects fainter than the limit specified by the parameter **fluxmax** will be used. If values for both ***fluxmin*** and ***fluxmax*** are set to ***None***, all detected objects will be used.

Table 4.21: tweakreg Parameters: "Coordinate File Description" (Continued)

Parameter	Default	Description
fluxunits	counts	Units of flux values Units of flux for parameters ***fluxmin*** and fluxmax. Choices are ***counts***, ***cps***, and ***mag***.
xyunits	pixels	Units of X,Y positions Units for object positions in the catalogs. Choices are ***pixels*** or ***degrees***.
nbright	None	Number of brightest objects to keep Integer value specifying the number of brightest objects to keep in the source catalog for each input image. The default value ***None*** means that all objects are used.

Table 4.22: tweakreg Parameters: "Reference Catalog Description"

Parameter	Default	Description
refcat	""	Filename of reference coordinate catalog Name of the external reference catalog file to be used in place of the catalog extracted from one of the input images.
refxcol	1	Column name(s) for RA Integer value identifying the column numbers for Right Ascension in the reference catalog specified by ***refcat***.
refycol	2	Column name(s) for Dec Integer value identifying the column numbers for Declination in the reference catalog specified by ***refcat***.
refxyunits	degrees	Units of sky positions The units for object positions in the reference catalog as specified by ***refcat***. Choices are ***pixels*** or ***degrees***.
rfluxcol	""	Column name for source flux/mag values Integer value identifying the column number for source fluxes in the reference catalog specified by ***refcat***.
rfluxmax	None	Maximum flux value for valid reference objects Maximum flux value (a real number) for objects in the external reference catalog. This value, if specified, is the upper limit in the range of fluxes for reference sources to be kept for the final reference catalog. For the default entry, ***None***, there is no upper limit in the range of fluxes for sources kept in the reference catalog.
rfluxmin	None	Minimum flux value for valid reference objects Minimum flux value (a real number) for objects in the reference catalog. This value, if specified, is the lower limit in the range of fluxes for reference sources to be kept for the final reference catalog. For the default entry, ***None***, there is no lower limit in the range of fluxes for sources kept in the reference catalog.
rfluxunits	mag	Units of flux values Flux units for the values specified in parameters ***rfluxmax*** and ***rfluxmin***. Choices are ***counts***, ***cps***, and ***mag***.
refnbright	None	Number of brightest reference objects to keep Integer value for how many brightest objects to keep in the reference catalog. If ***refnbright*** is set to ***None***, all objects will be used. This value is used in conjunction with the ***refcat*** parameter.

Table 4.23: tweakreg Parameters: "Object Matching Parameters"

Parameter	Default	Description
minobj	15	Minimum number of objects acceptable for matching Integer value for the minimum number of identified objects from each input image to use in matching objects from other images.
searchrad	1.0	The search radius for a match One way to determine an optimal search radius would be to use **ds9** to display all the images being combined and use the "Align WCS" option to align the images using their WCS information. For images from different visits, expect there to be small offsets when the images are blinked. The maximum offset in arcseconds for a single valid source from one image to another as reported by **ds9** can then be used to set this parameter. Another way to determine an optimal search radius value is to make sure ***residplots*** is set to ***Both***, and that ***use2dhist*** and ***see2dhist*** are set to ***yes***. After several runs with increasingly larger ***searchrad*** values, the number of matched sources will eventually "level out," indicating that an optimal fit solution may have been found. Additionally, the peak value displayed in the two-dimensional histogram plot should be sharp and unambiguous, located far enough from the edge of the plot that there's no chance that a better solution could be excluded by the limits of the search range. Furthermore, incorrect fit solutions will result in anomalous organized flow patterns in the vector plot, and odd shapes and/or tilts in the residuals plot. For the alignment of intravisit observations (which use the same guide stars for all observations), the default value of ***1*** arcsecond works well. However, for the alignment of intervisit observations, the search radius may to be extended to as much as ***4*** arcseconds.
searchunits	arcseconds	Units for search radius Units for the search radius that was specified in parameter searchrad. Choices for this parameter are ***arcseconds*** and ***pixels***.
use2dhist	True	Use 2d histogram to find initial offset? A two-dimensional histogram that shows the initial computed offsets between the reference image and each input image.
see2dplot	True	See 2d histogram for initial offset? Display the two dimensional histogram plot.
tolerance	1.0	Matching tolerance for **xyxymatch** (pixels) The matching tolerance in pixels after applying an initial solution derived from the *2dhist* algorithm. This parameter gets passed directly to an algorithm similar to **xyxymatch** for use in matching the object lists from each image with the reference image's object list.
separation	0.0	Minimum object separation (pixels) The minimum separation for objects in the input and reference coordinate lists. Objects closer together than ***separation*** pixels are removed from the input and reference coordinate lists prior to matching. This parameter gets passed directly to the **xyxymatch** algorithm for use in matching the object lists from each image with the reference image's object list.
xoffset	0.0	Initial guess for X offset(pixels) Initial estimate for the offset in X between the images and the reference frame. This offset will be used for all input images provided. If the parameter value is set to ***None***, no offset will be assumed in matching sources with **xyxymatch**.
yoffset	0.0	Initial guess for Y offset(pixels) Initial estimate for the offset in Y between the images and the reference frame. This offset will be used for all input images provided.If the parameter value is set to ***None***, no offset will be assumed in matching sources in ***xyxymatch***.

Table 4.24: tweakreg Parameters: "Catalog Fitting Parameters"

Parameter	Default	Description
fitgeometry	rscale	Fitting geometry The fitting geometry to be used in fitting the matched object lists. This parameter is used in fitting the offsets, rotations and/or scale changes from the matched object lists. Choices are ***shift*** and ***rscale***. The ***rscale*** option tells ***tweakreg*** to fit for an offset, a scale (same scale in X and Y), and a rotation (same rotation for X and Y axes). The ***shift*** option only results in a fit of the overall offset between images without fitting for any rotation or scale differences.
residplot	both	Plot residuals from fit? The choices are ***No plot***, ***vector***, ***residuals***, ***both***. If ***both*** is selected, the vector and residuals plots will be displayed in separate plotting windows at the same time.
nclip	3	Number of clipping iterations in fit
sigma	3.0	Clipping limit in sigma units

imagefindpars Parameter Details

The task **imagefindpars** can be optionally called by **tweakreg** to fine-tune object detection settings. Its algorithm is similar to that used by **daofind**, and has parameter names that resemble those in the IRAF version of **daofind**. However, the ***skysigma*** and ***threshold*** parameters are not defined identically to those in **daofind**. The ***computesig*** parameter attempts to automatically determine the value of ***skysigma*** for each image, but it is prone to failure for images with low background such as those containing globular cluster regions and nebula. In those situations, a user-defined ***skysigma*** can be used for all input images.

Table 4.25: imagefind Parameters

Parameter	Default	Description
computesig	True	Automatically compute sigma for all inputs? This parameter controls whether or not to automatically compute a sigma value for use in object identification. If set to ***True***, the value computed will override any user input for the parameter ***skysigma***. The automatic sigma value gets computed for each exposure as *1.5 * imstatistics(image,nclip=3,fields='stddev')*.This computation works fairly robustly for nearly all fields.
skysigma	0.0	Standard deviation of background in counts The standard deviation of the sky pixel values, only used if the parameter ***computesig*** is ***False***.
conv_width	3.5	Convolution kernel width in scale units This value is typically twice the FWHM of the point source. For ACS/WFC and WFC3/UVIS, this value is ~3.5 pixels. For WFC3/IR, ~2.5 pixels.
peakmin	None	Min source peak value Set a lower limit for the selection of source values, in units of the input image.
peakmax	None	Max source peak value Set an upper limit for the selection of source values, in units of the input image.

Table 4.25: imagefind Parameters (Continued)

Parameter	Default	Description
threshold	4.0	Threshold in sigma for feature detection The object detection threshold above the local background in units of sigma (background standard deviation).
nsigma	1.5	Width of convolution kernel in sigma The semi-major axis of the Gaussian convolution kernel used to compute the density enhancement and mean density images in Gaussian sigma.
fluxmin	None	Min good total source flux Only select sources with a total flux (in the units of the input image) greater than this value. The flux gets computed for all pixels in the source extraction region, typically a five pixel wide aperture depending on the value of the ***conv_width*** parameter.
fluxmax	None	Max good total source flux Only select sources with a total flux (in the units of the input image) less than this value.

Format of Exclusions Catalog

The **tweakreg** parameter ***exclusions*** accepts a catalog containing a set of files defining regions to avoid when finding sources. The format of an exclusions catalog requires one line for every input image, regardless of whether that image has exclusions regions files. The example below shows the contents of an exclusions catalog file containing two images; no exclusions files are specified for the first image, and for the second image, an exclusions *regions* file (suffix `.reg`) is specified for [sci,1] (that regions file is in the second column).

```
j99da1emq_flt.fits
j99da1f2q_flt.fits test_exclusion.reg
```

The next example shows an exclusions file catalog containing two images. There are no exclusions files for the first image and one exclusions regions file for the second chip in the second image. For the second image, ***None*** is entered in the second column to indicate there is no `[sci,1]` exclusions regions files, while the third column contains the name of the `[sci,2]` exclusions regions file. (***INDEF*** can also be used instead of ***None***.)

```
j99da1emq_flt.fits
j99da1f2q_flt.fits None test_sci2_exclusion.reg
```

Format of Regions Files

Exclusions files specified in an exclusions catalog have a default file format called "regions" files that are in a format generated by **ds9** using the **ds9/Funtools**. Shown below is an example of a regions file using the circle() regions syntax.

```
# Region file format: DS9 version 4.1

# Filename: j99da1f2q_flt.fits[SCI]

global color=green dashlist=8 3 width=1 font="helvetica 10 normal roman"
select=1 highlite=1 dash=0 fixed=0 edit=1 move=1 delete=1 include=1 source=1

image

circle(3170,198,20)

circle(3269,428,20)

circle(3241.1146,219.78132,20.000014)
```

TweakReg can also work with regions files with a much simpler format where each region has two values for the source center and a third value for the radius of the region. The units of all the values can be either pixels or sky coordinates (sexagesimal R.A./Dec. or decimal degrees R.A./Dec., with a radius in arcseconds). The values on each line can be separated either by spaces or commas, as shown in the example below.

```
imagename

3170, 198, 20

3269 428 20

3241.1146, 219.78132, 20.000014
```

The first (non-comment) line in the file specifies the units of the regions values. Supported options are:

image
physical
pixel
fk5
fk4
sky

The terms ***image***, ***physical***, and ***pixel*** all refer to values in pixels, while the remaining refer to values specified in sky coordinates. A default units of ***pixels*** is assumed if the unit type is not specified at the beginning of the simple format regions file.

The format of the regions can actually be different from one line to the next, as in cases where multiple regions files, written out in different formats, are merged; regions using the simple format, for instance, could be included in the first example file that uses the **ds9** format.

A Note About "Difficult" Images

Examples in this handbook utilize images with enough stellar point sources to allow alignment using the **daofind**-like source-finding algorith in TweakReg. There are images, however, that are filled primarily with nebulosity, or with small faint galaxies and not much else, certainly not with enough stars useful for tweaking the alignment.

What can one do in such situations? In the case of nebulosity or high galactic latitude fields with few stars, direct cross-correlation can be done on prominent features in the nebulosity. But while cross-correlation is reliable and easy to use when there are only *x*- and *y*-shifts, it is far less adept at dealing with rotations or differences in scale.

For fields with lots of small galaxies, software such as SExtractor could be used to generate catalogs for input to TweakReg. This will require removing cosmic rays first, but if there are a few images that are reasonably well-aligned (such as those taken within a single visit), then cosmic ray-cleaned images could be used as a first pass in AstroDrizzle.

The CANDLES Treasury program has settled on a hybrid approach (Ferguson, H., Koekemoer, A., private communications). Catalogs are used to obtain the rotation and scale between images to do a first estimate of the shift. Rotation and scale are then fixed, and the shift is further refined using cross-correlation. At present, there are no straightforward ways for users to give the software a delta-shift and have it immediately translated into a change in header WCS. Users wanting to do this will need to update the headers themselves (by changing the `CRVAL*` keywords in their images, for instance). A task may be available to do this automatically in a future version of the software.

Other techniques are being investigated, such as wavelets, for use in aligning more difficult images in a more routine and self-contained manner. For now, users will have to experiment with techniques that may be best suited for their unique images. Fortunately, most images have enough stars in them to obtain fairly robust alignments with the current TweakReg software.

4.4.3 TweakBack

Drizzled images have been fully distortion-corrected, making it easy to perform a fit between drizzled images using TweakReg to calculate the overall offset, rotation, and scale differences between the drizzled images. While applying a fit to align two drizzled images can be done very simply due to the lack of distortion, the calibrated input images (`flt.fits` or `flc.fits`) which were combined to create those drizzled images, however, still contain distortion, so a fit computed using drizzled images cannot be applied to the input images in a simple manner. This typically results in the user updating the drizzled image headers with the results of a fit without any means of updating the original distorted input images.

TweakBack takes the WCS information from a drizzled image, one that has been aligned to another drizzled image, and propagates the newly updated WCS information back to all the input images used to create the drizzled images. This will

allow those input images to be combined again to produce new aligned drizzled products.

When to Use TweakBack

Some situations require alignment of drizzled images, such as:

- Aligning images taken in one filter to those taken in another filter
- Aligning long-exposure sets of images (that are dominated by cosmic rays) taken in different visits
- Aligning mosaics of a sparse field to an external astrometric catalog

Creating combined drizzled images for each filter results in images free of cosmic rays and detector defects, making it ideal for finding and matching sources. These cosmic ray-free drizzled images can therefore maximize the number of sources detected and minimize the work needed to find matches between sources, resulting in the best possible fit between the images using the most possible sources. Alignment of these drizzled images using TweakReg will result in highly reliable and accurate updates to the drizzled image headers that can then be passed back to the original input images.

TweakBack Algorithm

Any update to the WCS information of an image should only be done after copying the original WCS keyword values as an alternate WCS (using FITS Paper I standards). This gets done automatically when using TweakReg to update the headers of images, and the task also provides a means to recover the original alignment if errors are made in the new WCS keyword values. This task relies on the presence of the original WCS and the newly-updated WCS to be recorded in the drizzled image's header as the last two alternate WCSs. The difference between the previous WCS values and the newly updated WCS values will be used as the basis for computing the changes that need to be applied to each distorted input image.

The algorithm used by this function follows these steps:

1. Verify or determine a list of distorted images that need to be updated with the final solution from the drizzled images.
   - The `D00*DATA` keywords from the `PRIMARY` header of the drizzled image will be used to build a list of input images that will need to be updated with the new WCS solution.
   - If no `D00*DATA` keywords can be found in the drizzled image `PRIMARY` header, then the user must provide a list of filenames for the images that need to be updated.
   - If the user does not specify a list of images and no `D00*DATA` keywords can be found, this task will report an error and quit.
2. Read in HSTWCS objects for last two alternate WCS solutions.

- The last two alternate WCS solutions (as generated when **tweakreg** updates an image header with a new WCS solution) represent the drizzled images starting WCS and the newly updated WCS.

3. Generate footprints using the STWCS *.calcFootprint()* method for each WCS.
   - Each footprint corresponds to the location of the corner pixel from the drizzled image as it appears on the sky.
4. The sky positions of the corners of the drizzled image get computed using the new updated WCS solution and using the original WCS solution.
   - The pixel positions for corners of each WCS get computed by running the *.wcs_sky2pix()* method for the last (updated) WCS
   - These pixel positions will be used to do a fit between the corner positions for the updated WCS and the drizzled image's original corner positions.
5. Perform linear "rscale" fit between the twp sets of *X,Y* coordinates.
   - The results of this fit will represent the correction applied to the original WCS to get the new updated WCS.
6. Update each input image WCS with the fit using ***updatehdr=yes*** code in **tweakreg**

This algorithm essentially backs out the correction that would have been applied by TweakReg if it had been run on two distorted input (`flt.fits`) images instead of drizzled images, then applies that correction to the distorted images just as TweakReg does itself. This means that input images updated using TweakBack will use the same conventions for keeping track of multiple WCS solutions in the input image headers just as if TweakReg was run.

This process has been tested on both ACS/WFC and WFC3/UVIS datasets and have demonstrated that this process can update images to as low as 0.001 pixels depending on the number of sources.

tweakback Usage

The syntax for running this task in a Python task is:

```
>>> from drizzlepac import tweakback
>>> tweakback.tweakback(<drizzle filename>)
```

Example:

An image named `acswfc_mos2_drz.fits` was created from four images using AstroDrizzle. This drizzled image was then aligned to another image using **tweakreg** and the header was updated so the keyword `WCSNAME` was assigned the

value TWEAK_DRZ. The new WCS can then be used to update each of the four images that were combined to make up this drizzled image using:

```
>>> from drizzlepac import tweakback
>>> tweakback.tweakback('acswfc_mos2_drz.fits')
```

If the same WCS should be applied to a specific set of images, those images can be updated using:

```
>>> tweakback.tweakback('acswfc_mos2_drz.fits',\
input='img_mos2a_flt.fits,img_mos2e_flt.fits')
```

A step-by-step example of how to run **tweakback** on real image data is available in Section 7.5

Table 4.26: tweakback Parameters

Parameter	Default	Description
drzfile	""	Name of file with updated WCS Filename of undistorted image which contains the new WCS and WCS prior to being updated.
input	""	Name(s) of files to be updated with new WCS These can be provided either as a list of images in a file entered as @filename, a comma-separated list of filenames, or using wildcards. NOTE: A blank value will indicate that the task should derive the filenames from the ***drzfile*** itself, if possible. The filenames will be derived from the D*DATA keywords written out by AstroDrizzle. If they can not be found and no images are specified in this parameter, the task will quit.
origwcs	""	WCSNAME of WCS upon input to **tweakreg**
wcsname	""	Name of WCS updated by **tweakreg**
extnam	SCI	EXTNAME of extension to be updated with WCS Name of extension in input files to be updated with new WCS.
verbose	False	Print processing messages? This parameter specifies whether or not to print out additional messages during processing.
force	False	Force updates of WCS? This parameter specifies whether or not to force an update of the WCS even though WCS already exists with this solution or wcsname.

4.4.4 Updating Images from the Pre-AstroDrizzle Archive

Observations retrieved from the Archive before June 2012 (please check the DrizzlePac webpage for the exact date) were processed in the pipeline by MultiDrizzle. To reprocess this data with AstroDrizzle, users need to update the World Coordinate System (WCS) in their images, which includes geometric distortion information, to a format compatible with AstroDrizzle.

The task **updatewcs**, in the STWCS package, is used to insert and format linear and polynomial distortion information into the image header for HST images, such as WFC3 and even STIS data.

For ACS data, new non-polynomial distortion reference files need to be added to the header and folded into the image as new image extensions. The **updatenpol** task in the DrizzlePac package evaluates the `DGEOFILE` reference file specified in the header and selects the appropriate `NPOLFILE` and `D2IMFILE` reference files to replace the `DGEOFILE` and updates the header with the names of those new reference files. This task can then call **updatewcs** to add that reference file information to the input image as new extensions along with updating the header with the current SIP keywords based on the `IDCTAB`, although the user can always run **updatewcs** manually if desired. The **updatenpol** task really only needs to be run on ACS data that were obtained from the HST Archive prior to AstroDrizzle becoming operational in the pipeline, and it only needs to be run ONCE on any ACS image in order to record the names of the new reference files in its image header. The **updatewcs** task can be run any number of times afterwards as needed to reset the WCS solution to the default solution generated by the pipeline.

updatewcs Requirements

The tasks for updating geometric distortion correction information requires reference files for:

- Updating linear and polynomial distortion corrections using the distortion corrections reference file named by the header keyword `IDCTAB`
- Updating non-polynomial distortion corrections, currently applicable only for ACS data, the reference files named in header keywords `D2IMFILE` (for detector-based distortions) and `NPOLFILE` (for optical non-polynomial distortions) are required. These reference files are available at the ACS Reference Files webpage.

If reference files are kept in a specific directory, for example, `/mydata/reffiles/`, directory pointers need to be defined to tell the software where the reference files are located. For WFC3, by default, reference file names are preceded by `iref$` where `iref` is a directory location for that reference file. The `iref$` pointer can be defined using the following command–for UNIX-based machines, this needs to be done in the shell window before starting PyRAF.

```
setenv iref /mydata/reffiles/
```

or, for the ACS jref$ pointer,

```
setenv jref /mydata/reffiles
```

Using updatewcs

updatewcs can be run by using the **updatewcs** task itself, or by setting the ***updatewcs*** parameter in **astrodrizzle** to ***True***[5]. This task is required for populating

older (pre-AstroDrizzle Archive pipeline data) WFC3, ACS, and STIS images with linear and polynomial distortion correction information in a format compatible with AstroDrizzle. To do this, the IDCTAB reference file is required.

The IDCTAB reference file can be downloaded from the instrument webpages by following a link to their reference files. To check that the images have the best-available IDCTAB reference file, use the *Best Reference Files* search in *StarView*. If a new IDCTAB file is listed, use it instead of the older one and be sure to update the IDCTAB header keyword in the images with the new reference file name; updating the header keyword can be done in PyRAF using the **hedit** task, as show below.

```
hedit image_flt.fits[0] IDCTAB "New_IDCTAB_reference_file"
verify=yes show=yes update=yes
```

After starting PyRAF (or Python), import the **stwcs** package which contains the task **updatewcs**.

```
from stwcs import updatewcs
```

For all flt.fits files in the working directory, the WCSs are updated to be compatible with AstroDrizzle with this command:

```
updatewcs.updatewcs('*flt.fits')
```

Using updatenpol for ACS images

updatenpol requires access to the distortion correction image that contains column width corrections and residual wavelength-dependent distortion corrections, named in the header keyword DGEOFILE. The **updatenpol** task updates ACS images with:

1. New header keywords NPOLFILE and D2IMFILE that hold the names of reference files containing information about non-polynomial distortions and detector-based distortions, respectively. It identifies these new reference files using the geometric distortion image reference file that is named by the header keyword DGEOFILE.
2. Distortion correction information in the NPOLFILE and D2IMFILE reference files, containing non-polynomial optical distortion corrections and detector-based distortion corrections, respectively, are inserted into the flt.fits images as new FITS extensions. If it is not already done, there's also an option to run **updatewcs** to ensure that all WCS information from the IDCTAB reference file has been incorporated into the ACS images.

Making WFPC2 Images Compatible with AstroDrizzle

Since WFPC2 data resides in a static Archive, after being processed by MultiDrizzle, c0m.fits and c1m.fits data need to be updated for compatibility

5. For ACS, **updatewcs** can also be run while executing **updatenpol**.

with AstroDrizzle. To do that, the reference files identified by the header keywords `IDCTAB`, `OFFTAB` (for time-dependent chip offsets), and `DGEOFILE` (for detector-based distortions) are required.

The first time WFPC2 data is processed by AstroDrizzle, simply set the **astrodrizzle** parameter ***updatewcs*** to ***yes***. AstroDrizzle will extract and convert detector distortion data in the `DGEOFILE` to the same format as in the `D2IMFILE` (detector distortion correction array) and insert that information into the FITS image as an extension. Information from the `IDCTAB` and `OFFTAB` are saved as SIP coefficients and other distortion header keyword values in the image header.

4.4.5 Handling WCS Information With stwcs

DrizzlePac relies on STWCS, another software package within STScI_Python, to manage all WCS-related operations. **stwcs** supports not only reading in the WCS information from FITS images as WCS objects in memory, but allows users to perform coordinate transformations using the full distortion model from the image header, among many other operations, with those WCS objects.

This software also provides the capabilities to package a WCS solution and save it as a file of its own for application to a copy of the original image; more specifically, it defines and supports the use of headerlets.

The STWCS package, in turn, relies on the generally available **pywcs** package which provides a Python interface to the C library WCSLIB which serves as the definitive implementation of all the approved FITS standards for WCS information. In short, any operation in any DrizzlePac task (such as **astrodrizzle** or **tweakreg**) dealing with the WCS gets performed by calling the **stwcs** package and its tasks.

STWCS consists of two primary subpackages: **updatewcs** and **wcsutil**. The task **updatewcs** (from the **stwcs.updatewcs** package) performs corrections to the basic WCS and includes other distortion information in the science files as header keywords or file extensions. The **stwcs.wcsutil** package implements many tasks including the most basic *HSTWCS object* which extends default FITS standard implementation of the WCS (as implemented by the *pywcs.WCS object*) as well as all the headerlet related tasks. The HSTWCS Python object provides HST instrument-specific WCS support as well as methods for coordinate transformations and serves as the primary in-memory implementation of the WCS information used by all the tasks in the DrizzlePac package. The coordinate transformation tasks in the DrizzlePac package (**pixtosky**, **skytopix**, **pixtopix**) all rely on the HSTWCS object implemented in the **stwcs** package for performing the actual computations of the requested coordinate transformations. The **wcsutil** package also provides functions for manipulating alternate WCS descriptions in the headers. Most of these functions do not have TEAL GUI interfaces, as they are designed for use within Python tasks or scripts or for interactive use under Python.

The package can be loaded into either a pure Python environment or under PyRAF using:

```
>>> import stwcs
```

stwcs tasks that have TEAL interfaces are listed every time the STWCS package is imported or reloaded in PyRAF and Python. They are:

updatewcs	Compute the WCS keywords and import the distortion model from the reference files
apply_headerlet	Apply a headerlet to a file
archive_headerlet	Save a WCS solution as a headerlet extension and write it out as a headerlet FITS file
attach_headerlet	Attach a headerlet as an extension to a file
delete_headerlet	Delete a headerlet extension from a file
extract_headerlet	Write out a headerlet extension as a separate FITS file
headerlet_summary	Print a summary of all headerlet extensions in a file
restore_headerlet	Replace current WCS solution with the WCS solution from a headerlet extension
write_headerlet	Save a WCS solution as a separate headerlet FITS file

These tasks can be run in a pure Python environment:

```
>>> from stsci.tools import teal
>>> teal.teal(<taskname>)
```

or from PyRAF:

```
--> epar <taskname>
```

where the taskname corresponds to the name given in the list of available TEAL tasks reported when the **stwcs** gets (re-)loaded.

More About updatewcs

The task **updatewcs** from the **stwcs** package recomputes the entire WCS for an image to incorporate all distortion corrections in the form of SIP and other WCS-related keyword values. This task can be run directly, or it can be invoked as a processing step in **astrodrizzle** and **updatenpol**.

Since AstroDrizzle is run in the HST pipeline, calibrated (`flt.fits`) data from the Archive already contains all necessary information to represent astrometrically precise positions. Therefore, distortion reference files are no longer needed for post-pipeline drizzling, as was the case for MultiDrizzle. A full description of the astrometric computations and how they include the full distortion model can be found in Chapter 3.

The example below calls **updatewcs** as a standalone operation using Python syntax.

```
>>> from stwcs import updatewcs
>>> updatewcs.updatewcs(filename)
```

The parameters recognized by **updatewcs** are:

Parameter	Default	Description
input	""	Input files (name, suffix, or @list) List of filenames or a string (wild card characters allowed); input files may be in FITS, GEIS, or waiver-FITS format.
extname	SCI	EXTNAME of extensions to be archived
vacorr	True	Apply velocity aberration correction? If ***True***, the velocity aberration correction will be applied if the image has a VAFACTOR keyword value.
tddcor	True	Apply time dependent distortion correction? If ***True***, a time-dependent distortion correction will be applied, if appropriate. Currently, only ACS data has been calibrated to account for this effect.
npolcorr	True	Apply look-up table distortion? If ***True***, a look-up table distortion correction will be applied, if appropriate. Currently, only ACS data relies on an external NPOLFILE reference file.
d2imcorr	True	Apply detector to image correction? If ***True***, the detector to image correction will be applied, if appropriate. Only ACS and WFPC2 images have been calibrated to account for this effect. The ACS corrections have been stored as the new D2IMFILE reference files. The DGEOFILE reference file for WFPC2 data gets interpreted by this task to generate a D2IMFILE correction that gets copied directly into the updated WFPC2 image as a new extension when the WFPC2 image gets converted into a multi-extension FITS file.
checkfiles	True	Check format of input files? If ***True***, the format of the input files will be checked; GEIS and waiver-FITS files will be converted to Multi-extension FITS (MEF) format. The default format for WFPC2 data retrieved from the static Archive is the Multi-extension FITS (MEF) format, with suffixes c0m.fits (science images) and c1m.fits (data quality images).

All relevant distortion reference files must be available when this task is run, and image header keywords for specifying these reference files should be updated to reflect the correct file names. All instruments require the availability of the IDCTAB. For WFPC2 data, the DGEOFILE and OFFTAB reference files are also needed for the **updatewcs** task if run by itself or if run by turning on the ***updatewcs*** parameter in the **astrodrizzle** or **tweakreg** tasks. The DGEOFILE for WFPC2 data is converted into a D2IMFILE (that contains the 34th row corrections) which is inserted into the header as an FITS extension. Information from the IDCTAB and OFFTAB reference files are inserted into the header as keyword values. For ACS images, this task also requires the addition of the NPOLFILE and D2IMFILE header keywords and the availability (on disk) of the appropriate reference files for the exposure. The DGEOFILE reference files do not get used by **updatewcs** or **astrodrizzle** at all (for ACS data) and neither reference file needs to be accessible.

More About Running updatenpol for ACS Images

The **updatenpol** task in the DrizzlePac package uses the geometric distortion image named by the DGEOFILE header keyword to determine the names of new reference files for correcting residual optical distortions and detector-specific distortion keywords. These reference files are recorded in the NPOLFILE and D2IMFILE header keywords, respectively. This task will also optionally run the **updatewcs** task to insure that all the WCS information from those new reference files, and the IDCTAB, has been imported into the ACS file(s) as needed by AstroDrizzle, making it unnecessary to have the **updatewcs** parameter in **astrodrizzle** turned on.

This program requires access to the jref$ directory in order to evaluate the DGEOFILE specified in the input image header. This evaluation allows the program to get the information it needs to identify the correct NPOLFILE reference file.

The **updatenpol** task can be called from PyRAF and Python task using this syntax:

```
from drizzlepac import updatenpol
  updatenpol.update(input, refdir='jref$', local=None,
interactive=False, wcsupdate=True)
```

or the TEAL interface can be used from within PyRAF using:

```
epar updatenpol
```

The definitions of all the parameters for this task can be found in the following table.

Table 4.27: Parameters for the Task updatenpol

Parameter	Default	Description
input	*flt.fits	Input files (name, suffix, or @list) Name of input file or files; acceptable forms: • Single filename with or without directory • @-file • Association table • Comma-separated list of filenames • Filenames as a "Python list" • Wildcard specification of filenames
refdir	jref$	Directory containing new reference file Path to directory containing new reference files, either environment variable or full path. This task understands IRAF-styled definitions for directories (like jref$) assuming those variables have been defined in the users environment. Typically, these variables need to be defined at the operating system level for the Python environment (or PyRAF) to use (i.e., setenv jref /user/myreffiles/)

Table 4.27: Parameters for the Task updatenpol (Continued)

Parameter	Default	Description
local	False	Copy new reference files to local directory? Specifies whether or not to copy new reference files to local directory for use with the input files.
interactive	False	Prompt user to update each file? Specifies whether or not to interactively ask the user for the exact names of the new reference files instead of automatically searching a directory for them.
wcsupdate	True	Update WCS information using new reference files? By default, this parameter will run the **updatewcs** task on the input files to incorporate the new reference files and distortion information into the image headers for use by **astrodrizzle**. If turned off, the user can always run the **updatewcs** task manually or simply turn on the ***updatewcs*** parameter when running either **astrodrizzle** or **tweakreg**.

4.4.6 blendheaders

A fundamental problem exists when trying to combine multiple images into a single product; namely, how to account for the header information from all the input images that went into generating the output product. The operation of drizzling multiple input images together to create a single output product of higher scientific value runs into this problem every time. The original solution to this problem implemented by MultiDrizzle was to simply use the first input image header as the basis for the final output header, modifying only a few keywords (namely, `EXPSTART`, `EXPEND`, `EXPTIME`, `TEXPTIME` and `ROOTNAME`) to make it appropriate for the output image. This resulted in an output header which did not accurately reflect the full set of input image information while also containing keywords which no longer apply to a drizzled image at all.

The solution implemented for AstroDrizzle, the new **blendheaders** task, results in a final output image header which contains a more complete record of all the keyword values from all the input images along with a table extension (called `HDRTAB`) that records values which change from one input image to another while eliminating keywords that no longer apply to drizzle products. This solution relies on rules specified by the user, or by the instrument teams for pipeline use, that describe what should be done with each keyword from every input image.

Some keywords from the input images can be merged into a reasonable single value for the output image using a simple operation such as mean, or first value, or last value based on the full list of input values from all input images. Other keywords, however, vary from one image to another and can not be combined into a reasonable single value and those keywords get flagged by the rules for population of a new table extension with the name `HDRTAB`. This table has a column for each keyword, and each row corresponds to the values derived from each chip (not file). This allows the user to get a full record of, for example, how the `CRVAL` values varied for all the input images.

The merging of headers into a new header and a HDRTAB table extension has been implemented as the **blendheaders** task in the new **fitsblender** package. A TEAL interface allows users to run this task on drizzled images that were created by AstroDrizzle or MultiDrizzle.

Many users may have data that had been combined using MultiDrizzle and would like to see the full set of input image header values that went into creating that product. Running this task manually allows them to do this. Alternatively, anyone running AstroDrizzle has the option of defining their own set of rules for combining the headers, and use that set of rules to create a new drizzle product header and table that would be different from the default header generated by AstroDrizzle. For most users, though, the default rules should be sufficient as they were designed to account for the majority of keywords found in most HST imaging data. Additional documentation on the task itself can be found online under the Documentation link of the STScI_Python web page at STScI at:

http://www.stsci.edu/institute/software_hardware/pyraf/stsci_python.

This software was designed primarily to support merging headers of images which all share the same basic FITS structure. For example, **blendheaders** could be run on a list of images that have a PRIMARY header and two SCI extensions, or on a list where all inputs are simple FITS files with only a PRIMARY header. It may be possible to combine images with differing numbers of science FITS extensions with missing information being represented with values of ***INDEF*** or ***NaN*** in the summary table.

Running blendheaders

This task can be run (after loading the **drizzlepac** package) under PyRAF using the standard syntax:

```
epar blendheaders
```

or from a pure Python command:

```
>>> from stsci.tools import teal
>>> teal.teal('blendheaders')
```

The function can also be called from within a Python task or interactively using the pure Python syntax:

```
>>> from fitsblender import blendheaders
>>> blendheaders.blendheaders(drzfile,inputs=None,\
output=None,sciext='SCI',errext='ERR',dqext='DQ',verbose=False)
```

The parameters used by **blendheaders** are described in the following table.

Table 4.28: Parameters for the Task blendheaders

Parameter	Default	Description
drzfile	""	Name of drizzled file The drizzled file will be updated with the new header unless the output parameter has been specified. In addition, the drizzled file header will be interpreted to get a list of all the input images that went into its creation, if inputs is not specified, so that those images can be read in to get all the input keyword values.
inputs	""	Names of input files used to create the drizzled product Filenames with extensions for each chip in this list will be used to get the headers which will be blended into the final output headers. For example, `['j9cd01kqq_flt.fits[sci,1]','j9cd01kqq_flt.fits[sci,2]']` would create a blended header based on these chips headers.
output	""	Name of new file with blended headers If specified, a new file will be written out with this filename that contains the updated (blended) headers.
sciext	SCI	`EXTNAME` of science extension from input images `EXTNAME` of extensions with science data from the input FITS files. The header of this extension will be used as the basis for the `SCI` header of the drizzled product FITS file.
errext	ERR	`EXTNAME` of error extension from input images `EXTNAME` of extensions with the error array from the input FITS files. The header of this extension will be used as the basis for the `WHT` header of the drizzled product FITS file. If blank or ***INDEF***, it will use the `SCI` header as the basis for the output header for the `WHT` array.
dqext	DQ	EXTNAME of data quality extension from input images `EXTNAME` of extensions with the data quality array from the input FITS files. The header of this extension will be used as the basis for the `CTX` header of the drizzled product FITS file (when a `CTX` extension gets created). If blank or ***INDEF***, it will use the `SCI` header as the basis for the header of any generated `CTX` array.
verbose	False	Print out additional processing messages?

Drizzled Image Header Summary Table

The summary table in the drizzled image contains a table of useful header keyword parameters that characterize each input image. The table itself consists of a single row for each SCI array (chip) that was combined to create the final image product. Each column of the table represents a keyword from the `PRIMARY` and `SCI` extension of each chip. The names of the columns, by default, will match the names of the original keyword from each input header. However, the rules can rename those columns (more about that later) to whatever meets the needs of the software generating the final combined product. Renaming the columns, though, introduces a level of indirection when trying to map the values in the table to original header keywords and should only be done sparingly.

This table will get written out (by **blendheader**) as a binary table (*pyfits.BinTableHDU*) extension with `EXTNAME` of type `HDRTAB`. The header for this table will only contain the column definitions for the table and other required FITS keywords for the binary table.

blendheaders Rules

The operation of **blendheaders** relies, as implied earlier, on a set of rules that specifies what to do with all the input values for each keyword in the input file headers. The rules which define how to manage specific keywords need to be specified as a list defining which keyword arguments are to be aggregated and how.

The default set of rules for all HST instruments are included with the package using the filename convention <*instrument*>*_header.rules*. Any local file with an extension of `*.rules` will be read in as a user-supplied set of rules if it meets the formatting requirements described in this section.

- Each element in the list should be a sequence with two to five elements
- First element: name of the keyword from the input header
- Second element: name of the keyword that will be used to report the output value, or the name of a table column to record all the input values
- Third element: if given, it specifies a function to be used on the list of input values for a keyword to generate a new output value
- Fourth and fifth elements: these specify how to deal with error conditions and what value to report when an error occurs in processing a set of input values

These rules have been refined for specification in ASCII files, as opposed to using direct Python syntax. The format for the rules file has been defined as:

```
!VERSION = <floating point number>
!INSTRUMENT = <name of instrument: ACS, WFC3, STIS, NICMOS, WFPC2>
#
# Comments in the file can be included with this syntax
#
# Each line corresponds to a single rule for how to handle a single keyword
# Multiple lines for a single keyword can be specified
# This can be used to specify how to generate a new header value and to give
# the name of the column for the input keyword values
#
# Syntax follows the rules:
#
[<delete>]<keyword> [<new name> [<function name>]] # Comment
#
# Examples of valid rules would be:
#
EXPSTART     # Record all EXPSTART values in the table column with the same name
EXPSTART ESTART        # Record all EXPSTART values in the table column "ESTART"
EXPSTART EXPSTART min  # New value of EXPSTART is minimum of all input values
<delete> REFFRAME      # do not include REFFRAME in any output product
/ PROPOSAL INFORMATION # copy all keywords from this section into the table
<delete> / POST FLASH  PARAMETERS # delete all keywords in the section from outputs
```

The lines !VERSION and !INSTRUMENT are used to recognize this file as a valid rules file, and to associate this file with a specific version of fitsblender and apply it to headers for data from this instrument. This format, as supported by the current implementation of the code, only supports the use of the first three elements recognized by the fitsblender engine. Later versions can be updated to support use of the error elements for rules.

CHAPTER 5:

Introduction to the drizzlepac Interface

In this chapter…

5.1 Introduction to TEAL and Python Interfaces

DrizzlePac[1] introduces a new approach to HST data processing. Historically, HST data analysis had its roots in IRAF, specifically the STSDAS package. Over the past decade, there has been a gradual migration towards using Python and C for coding the tasks. This led to a Python interface that mimicked IRAF/STSDAS called PyRAF, which also came with a GUI interface for editing task parameters.

With the introduction of the DrizzlePac[2] package, more Python syntax is being utilized and a new GUI interface, called TEAL, is available for editing task parameters. DrizzlePac tasks can be run from the TEAL GUI, or as Python commands in the PyRAF or Python environment.

For users unfamiliar with Python but know PyRAF or IRAF, it is best to start learning DrizzlePac in PyRAF because PyRAF has both full Python functionality and

1. DrizzlePac is part of a library of Python routines called stsci_python that also include PyFITS, PySynphot, and Numdisplay.

2. Throughout this document, direct references to tasks and packages are written in bold lowercase, to impress on users that they must be entered this way in the command-line (i.e.,**astrodrizzle**). General references to the software are written in the default font (i.e., AstroDrizzle).

capability to run commonly-used legacy IRAF tasks (e.g., **imexam**) to inspect the data products.

The following sections describes how to load **drizzlepac**, use TEAL and Python commands for running **drizzlepac** tasks.

5.2 Starting drizzlepac in PyRAF and Python

Python packages contain modules and functions that can be run after the package has been loaded. These modules and functions are also displayed every time the package is loaded or reloaded.

IRAF packages are loaded by typing the package name at the ***cl*** command-line. The **drizzlepac** package can only be run in the PyRAF and Python environments, and must be loaded using Python syntax

The command to load a Python package is **import**.

```
--> import drizzlepac
The following tasks in the stwcs.gui package can be run with TEAL:
  apply_headerlet    archive_headerlet  attach_headerlet
  delete_headerlet   extract_headerlet  headerlet_summary
  restore_headerlet  updatewcs          write_headerlet

The following task in the fitsblender package can be run with TEAL:
  blendheaders

The following tasks in the drizzlepac package can be run with TEAL:
  astrodrizzle  imagefindpars  pixtopix   pixtosky
  resetbits     skytopix       tweakback  tweakreg
  updatenpol
```

In this document, the software is sometimes written as "DrizzlePac" when it is mentioned in general terms, but as a command, it is case-sensitive and has to be called in lowercase, "drizzlepac".

If **drizzlepac** is already imported, the **reload** command provides a reminder of the task names.

```
--> reload(drizzlepac)
The following tasks in the drizzlepac package can be run with TEAL:
  astrodrizzle  imagefindpars  pixtopix   pixtosky
  resetbits     skytopix       tweakback  tweakreg
  updatenpol
<module 'drizzlepac' from
'/usr/stsci/pyssgdev/2.7.stsci_python/drizzlepac/__init__.pyc'>
```

5.3 Using the TEAL GUI

DrizzlePac tasks for aligning and combining images can be run using a new interactive GUI interface called TEAL. It allows users to set parameter values for controlling operation of the task. TEAL also guides users in setting the appropriate task parameters; it can load prepared text parameter files (with suffix `.cfg`) as well as save new parameter settings to a uniquely-named text parameter file. Help files can be accessed from TEAL, and displayed as the user continues to edit tasks parameters.

Visually, the TEAL interface looks similar to the "**epar**" GUI task editor developed for IRAF tasks in PyRAF. The TEAL interface, however, provides additional capabilities to assist users in editing **drizzlepac** task parameters.

In PyRAF, the TEAL GUI for the **drizzlepac** task, **tweakreg**, is launched using the **epar** command, for example, as follows:

```
--> epar tweakreg
```

The Python command-line syntax, typed from either [3]PyRAF or the conventional Python interface, requires a command to load the TEAL task before using it to edit **tweakreg**. To repeat the example above using Python syntax:

```
--> from stsci.tools import teal   #Only needs to be done once
--> teal.teal('tweakreg')
```

The window's title bar shows which configuration file was used to fill in the parameter values seen when the GUI is launched. Any changes made to the parameters are saved to this file when the task is executed. The pull-down menus found in the menu bar contain general TEAL operation options. Parameter changes to a task can be saved to a *configuration file* (more about that in Section 5.5) using the ***Save As*** option

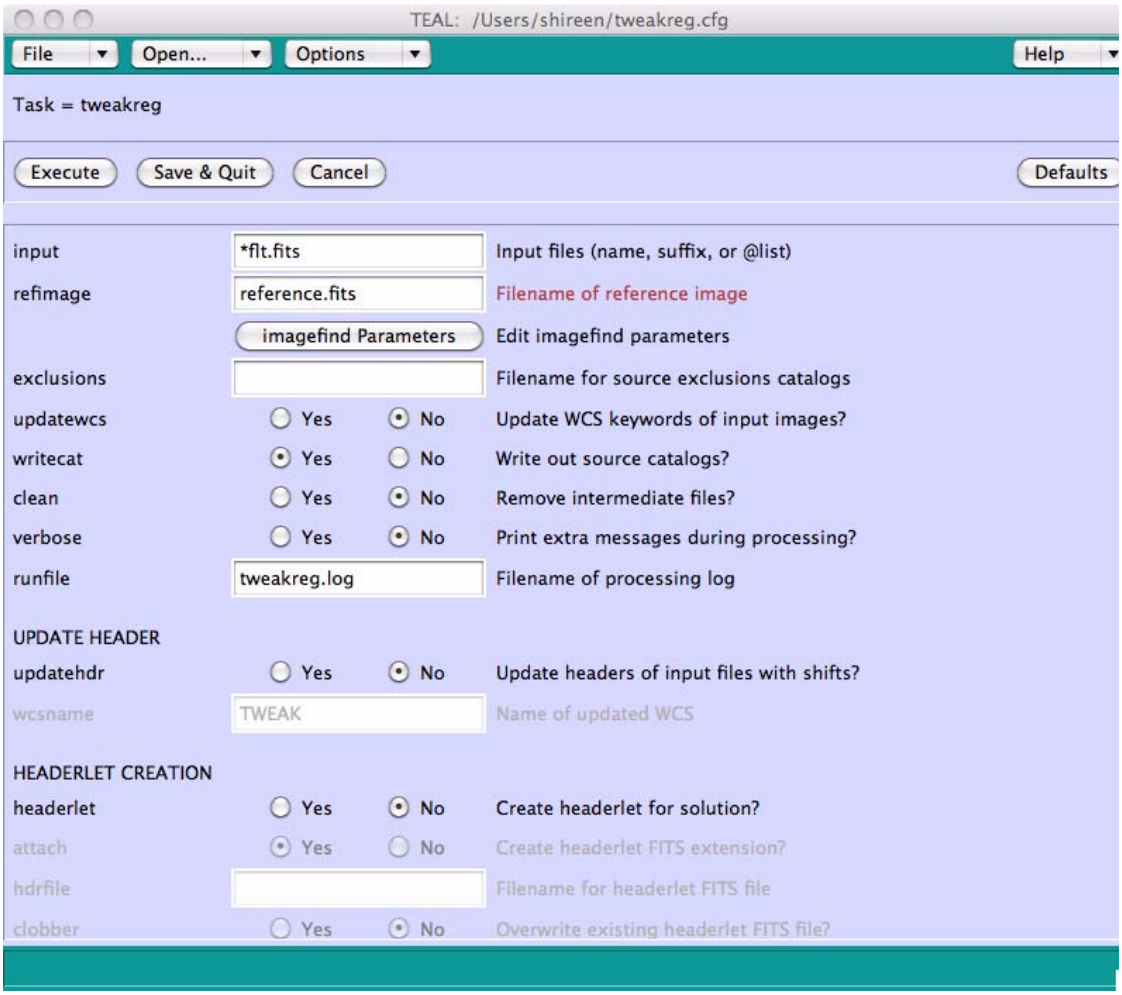
3. Note the difference between the PyRAF (- - >) and Python (>>>) command-line prompts

in the ***File*** pull-down menu. Under the '***Help***' menu, there are two options: either display help for TEAL or for the task that's being run in the GUI. The task help file can be displayed while parameters are being edited in the GUI, some help files are in the form of a html file displayed in a browser, and others appear as a text file displayed in a window.

The toolbar, found below the menu bar, has four buttons for commonly-used menu bar actions:

- ***Execute***: Runs the task using current parameter settings. Parameter settings will also be saved to the file listed at the top of the window
- ***Save & Quit***: Saves the current parameter values to the configuration file listed at the top of the window, without running the task
- ***Cancel***: Closes the window without saving any changes and without running the task
- ***Defaults***: Changes ALL the parameter values to their defaults

Figure 5.1: The TEAL GUI Displaying a Portion of the tweakreg Task



The TEAL GUI interface is especially useful for large tasks because it offers some new features that will help clarify which parameter settings to use for specific steps. Whenever a step in a task is turned off, the parameters that control that step are greyed out to be unavailable for editing by TEAL.For example, in Figure 5.1, the ***UPDATE HEADER*** and ***HEADERLET CREATION*** sections have been set to "***No***". As a result, parameters affecting those sections have been greyed out and cannot be edited.

Each parameter has a short description next to it. Whenever a parameter is changed from its default value, its description changes color to indicate a change has been entered. For example, in Figure 5.1, the ***refimage*** parameter, which is blank by default, has the name of a reference image, and as a result, its description is colored red. To clear or change an individual parameter back to its default value, right-clicking on the parameter will bring up a menu that will provides these options.

5.4 Using the PyRAF/Python Command-line

Some users prefer command-line interaction with software, rather than a GUI interface. This can be done from the PyRAF or Python environment using Python commands. For the convenience of users who are familiar with IRAF, a few IRAF commands have been implemented such that the "IRAF" command is translated to Python syntax. The equivalent Python syntax can be viewed in Table 5.2.

epar: launches the TEAL GUI

unlearn: resets all the parameters in a task to their default values

help: shows the help files for a task

When **drizzlepac** is imported, it displays these messages on the screen:

```
--> import drizzlepac
 The following tasks in the stwcs.gui package can be run with TEAL:
   apply_headerlet     archive_headerlet   attach_headerlet
   delete_headerlet    extract_headerlet   headerlet_summary
   restore_headerlet   updatewcs           write_headerlet
 The following task in the fitsblender package can be run with TEAL:
   blendheaders
 The following tasks in the drizzlepac package can be run with TEAL:
   astrodrizzle  imagefindpars  pixtopix   pixtosky
   resetbits     skytopix       tweakback  tweakreg
   updatenpol
```

DrizzlePac tasks are executed using Python syntax. This syntax is analogous to running a command in a Unix shell in that a "path" needs to be provided to the task. Options for the task are entered as "keyword-value pairs," all wrapped in a parenthesis. The first parameter in that parenthesis is always the files to be operated on, followed by the rest of the keyword-value pairs which may be optional. A typical command has the following syntax:

```
package.module.task(files,keyword1=value1, keyword2=value2.....)
```

For example, here is **tweakreg** as a Python command:

```
--> drizzlepac.tweakreg.TweakReg('*flt.fits')
```

Table 5.1 lists the full Python command syntax for each of the tasks available in drizzlepac.

Table 5.1: Python Syntax for DrizzlePac Tasks

Task	Python Command
astrodrizzle	drizzlepac.astrodrizzle.AstroDrizzle(<files>,<options>)
imagefindpars	(a "pset" or parameter set, only callable by tweakreg.)
pixtopix	drizzlepac.pixtopix.tran(<files>,<options>)
pixtosky	drizzlepac.pixtosky.xy2rd(<files>,<options>)
resetbits	drizzlepac.resetbits.reset_dq_bits(<files>,<options>)
skytopix	sdrizzlepac.skytopix.rd2xy(<files>,<options>)
tweakback	drizzlepac.tweakback.tweakback(<files>,<options>)
tweakreg	drizzlepac.tweakreg.TweakReg(<files>,<options>)
updatenpol	drizzlepac.updatenpol.update(<files>,<options>)

See the help file for each task to see the full list of options available.

There is no need to import an entire module if only one task will be used. The individual task can be imported in the following manner:

```
--> from drizzlepac import tweakreg
```

From that point forward, the package name does not have to be specified whenever that task is run during the Python session. A slightly shorter version of the command can be used:

```
--> tweakreg.TweakReg(<files>,<options>)
```

Commands for running **drizzlepac** from a Python command line are identical to the PyRAF commands, with the exception of the "IRAF-like" commands mentioned earlier and summarized in Table 5.2. Those "IRAF-like" commands require the TEAL module from the **stsci** package to mimic IRAF functionality. To import the TEAL package:

```
>>> from stsci.tools import teal
```

Table 5.2: "IRAF-like" Commands that Call Python Syntax to Execute a Task

IRAF Command	Python Equivalent	Note
epar	teal.teal('<taskname>')	taskname must be in quotes
unlearn	teal.unlearn('<taskname>')	taskname must be in quotes
help	help(drizzlepac.<taskname>)	

These commands are only available in PyRAF.

As previously mentioned, Python commands can be executed with optional keyword parameters. These are passed to the task as keyword-value pairs (i.e., ***searchrad=2.0***). The keywords specified in Python are the same as those seen in the TEAL GUI.

As shown in Figure 5.1, the TEAL GUI has several types of keyword value types: string; float; integer; boolean "Yes" or "No" buttons; pull-down menus. Each of these requires a different format when they're entered in the Python command-line.

The following command illustrates how these value types are specified; here, the back-slash simply continues the command on the next line. The input "***`flc.fits`***" files specification is not a typo–this is a new CTE-corrected calibrated ACS data product from the pipeline that can be used instead of `flt.fits` files.

```
>>> tweakreg.TweakReg('*flc.fits',updatewcs=True,\
minobj=20,searchunits='pixels')
```

In this command, the first parameter value is for the input files, written as a string wrapped in quotes: ***`'flc.fits'`***. Next, the ***updatewcs*** parameter value is a boolean set to ***True***–note that it does not require quotes and that it is case-sensitive. (In the TEAL GUI, ***updatewcs*** has a yes/no button). The third parameter, ***minobj***, requires an integer value. Whenever a float or integer value is needed, simply enter the number without quotes. Finally, in this command, is the ***searchunits*** parameter that has a choice of values which are documented in the help files. One of them, used in this example, is ***'pixels'***–a quote is required but it is not case sensitive. (In the TEAL GUI, this parameter has a pull-down menu that allows the user to select a value from several settings.)

Table 5.3: Summary of Python Syntax Requirements for Types of Parameter Values

Parameter Value Type	Format	Quotes Needed?	Case-sensitive?
epar	string	no	yes
unlearn	True or False	no	yes
help	integer or float	no	no
Pull-down	string	yes	no

The real power of having **drizzlepac** written in Python is that it can be called and used from within a Python script. For instance, if a user wishes to run **updatenpol** and **tweakreg** on ACS/WFC images in a directory, a Python script can be written to run it as follows:

```
from drizzlepac import updatenpol,tweakreg
from stsci.tools import teal
import glob

list_of_images = glob.glob('*flt.fits') # Makes a list of flt.fits images in
the current directory

teal.unlearn('updatenpol') # Reset all parameters to default values
teal.unlearn('tweakreg') # Reset all parameters to default values

updatenpol.update(list_of_images,refdir='/path/to/reference/files/',\
          local=True)

tweakreg.TweakReg(list_of_images,refimage='myreferenceimage.fits',\
        clean=True,verbose=True,runfile='ngc1234.log',searchrad=2.0,\
        searchunits='pixels')
```

Not only does this script keep a record of how images were processed, changes can be easily implemented by editing the script and re-running the commands.

5.5 Configuration Files (cfg)

Sharing parameter values with collaborators can be daunting for large tasks like **astrodrizzle** and **tweakreg**. That is why configuration files were created; these are ASCII text files containing parameter names and corresponding values for a task.

A task configuration file containing specific parameter values can be loaded into the TEAL GUI to execute that task. To load a particular configuration file, click on the "Open" pull-down menu at the top left of the TEAL window. If the configuration file is already listed, simply click on it. Otherwise, click "Other" to open a window for loading the file.

Configuration files can also be run from the Python command-line by using the ***configobj*** parameter name. If many specific **tweakreg** parameter settings are required, specifying them as a Python command would look quite untidy. Instead of listing all parameters in one line of code, a separate uniquely-named configuration file (e.g., a

file called "`tweakreg_example.cfg`") could be created with custom settings, and passed to the task using ***configobj*** parameter.Configuration files can also be run from the Python command-line by using the ***configobj*** parameter name.

A sample configuration file for **tweakreg** is shown below. Users who wish to create their own custom configuration files for a specific task can create one by opening a task in TEAL and saving the parameters and their values to a file. (To start off with default values, be sure to reset the parameters by clicking on the ***Defaults*** button before saving the configuration file. The ***Save As*** button is under the ***File*** menu item.) Alternately, a configuration file can be retrieved from the ~/.teal directory in the user's home directory–this directory and a default-setting configuration file were created when the software was first used by the user.

These are the contents of a customized teal configuration file, called `tweakreg_example.cfg`, that matches the last **tweakreg** command example above.

```
_task name = tweakreg #
input = "" # Input files (name, suffix, or @list)
refimage = myreferenceimage.fits # Filename of reference image
exclusions = "" # Filename for source exclusions catalogs
updatewcs = False # Update WCS keywords of input images ?
writecat = True # Write out source catalogs?
clean = True # Remove intermediate files?
verbose = True # Print extra messages during processing?
runfile = ngc1234.log # Filename of processing log

[UPDATE HEADER]
updatehdr = False # Update headers of input files with shifts?
wcsname = TWEAK # Name of updated WCS

[HEADERLET CREATION]
headerlet = False # Create headerlet for solution?
attach = True# Create headerlet FITS extension?
hdrfile = "" # Filename for header let FITS file
clobber = False # "Overwrite existing headerlet FITS file?"
hdrname = "" # Unique name (HDRNAME) for headerlet
author = "" # Author name forcreator of headerlet
descrip = "" # Short description of headerlet solution
catalog = "" # Name of catalog used for headerlet solution
history = "" # Name of ASCII file containing history for headerlet

[OPTIONAL SHIFTFILE OUTPUT]
shiftfile = False # Create output shiftfile?
outshifts = shifts.txt # Filename of generated shiftfile
outwcs = shiftswcs.fits # Filename of shift file reference WCS
```

```
[COORDINATE FILE DESCRIPTION]
catfile = "" # File containing coordinate filenames for input files
xcol = 1 # "Column name (s) for X positions"
ycol = 2 # "Column name (s) for Y positions"
fluxcol = "" # "Column name for source flux/mag values"
fluxmax = None # Maximum flux value for valid objects
fluxmin = None # Minimum flux value for valid objects
fluxunits = counts # Units of flux values
xyunits = pixels # Units of X/Y positions
nbright = None # Number of brightest objects to keep

[REFERENCE CATALOG DESCRIPTION]
refcat = "" # Filename of reference coordinate catalog
refxcol = 1 # "Column name (s) for RA"
refycol = 2 # "Column name(s) for Dec"
refxyunits = degrees # Units of sky positions
rfluxcol = "" # "Column name for source flux/mag values"
rfluxmax = None # Maximum flux value for valid reference objects
rfluxmin = None # Minimum flux value for valid reference objects
rfluxunits = mag # Units of flux values
refnbright = None # Number of brightest reference objects to keep

[OBJECT MATCHING PARAMETERS]
minobj = 15 # Minimum number of objects acceptable for matching
searchrad = 2.0 # The search radius for a match
searchunits = pixels # Units for search radius
use2dhist = True # Use 2d histogram to find initial offset?
see2dplot = True # See 2d histogram for initial offset?
tolerance = 1.0 # Matching tolerance for xyxymatch (pixels)
separation = 0.0 # Minimum object separation (pixels)
xoffset = 0.0 # Initial guess for X offset (pixels)
yoffset = 0.0 # Initial guess for Y offset (pixels)

[CATALOG FITTING PARAMETERS]
fitgeometry = rscale # Fitting geometry
residplot = both # Plot residuals from fit?
nclip = 3 # Number of clipping iterations in fit
sigma = 3.0 # Clipping limit in sigma units
```

The **tweakreg** command in the example program would now be much shorter, and the code cleaner.

```
from drizzlepac import updatenpol,tweakreg
from stsci.tool s import teal
import glob

list_of_images = glob.glob('*flt.fits') # Makes a list of flt.fits images in
the current directory

teal.unlearn('updatenpol') # Reset all parameters to default values
teal.unlearn('tweakreg') # Reset all parameters to default values

updatenpol.update(list_of_images,refdir='/path/to/reference/files/',\
          local=True)

tweakreg.TweakReg(list_of_images,configobj='tweakreg_example.cfg')
```

CHAPTER 6:

Reprocessing with the DrizzlePac Package

In this chapter…

6.1 Beyond the Standard Calibration Pipeline

DrizzlePac is written in Python (with algorithms written in C). Its interface is a departure from conventional IRAF usage that's familiar to most users. To learn more, please refer to Chapter 5 for an introduction to using the TEAL GUI interface and Python commands to run drizzlepac tasks.

Instrument pipelines provide data calibrated to a level suitable for initial evaluation, but may not be suitable for scientific analysis. When users place a data request at the HST Archive, their data is processed by the *On-The-Fly Reprocessing* (*OTFR*) system using the best-available software to calibrate data with the best-available reference files. The same calibration software updates are also propagated in releases of the IRAF/PyRAF STSDAS software package. If it has been a long time since their data was retrieved from the Archive, users are encouraged re-request it to ensure that the data contains the most up-to-date header information and calibrations.

There are presently two major steps in the OTFR process: (1) calibration of individual datasets using instrument-specific calibration software, such as **calacs** for ACS and **calwfc3** for WFC3; (2) combining associated data with AstroDrizzle to produce a combined, distortion-corrected, and largely cosmic ray-free image.

The second step cannot succeed without good results in the first. There may be occasions when pipeline calibration of individual images require custom calibration by the user. Instances when On-The-Fly reprocessing is not ideal may include a user's preference for self-made calibration reference files, the use of non-default calibration switches, and non-default software parameter values. Reason for these actions could be to improve hot pixels and cosmic ray removal, and deal with image persistence or other additional sources of noise.

For example, NICMOS data may require special attention; images from this camera often contain additional signal in the sky, persistence or pedestal effects (differing bias levels between quadrants in the chip) that require extra processing for removal. For more detailed information on recognizing and removing these effects in NICMOS data, please refer to Chapter 4, Anomalies and Error Sources, in the NICMOS Data Handbook.

Even when individual datasets from the Archive appear well-calibrated, users should consider if reprocessing their images with AstroDrizzle on their home machines is necessary. Drizzled pipeline data was created with conservative values (stored in the `MDRIZTAB` reference file). Some things to consider, depending on the type of data: drizzling with a finer output scale may produce better cosmic ray rejection. Using a smaller pixfrac will reduce correlated noise. Using both a smaller pixfrac and a smaller scale can produce a sharper PSF. In many cases, one can produce better images with a bit of effort. But if upon examining the pipeline output, if a user believes it is sufficient for his or her purpose, there is no need to re-drizzle the data.

6.2 Image Alignment

Flat-field calibrated images from the OTFR pipeline have been updated to incorporate the full distortion model that is stored in SIP header keywords and the updated CD matrix. This was done in the pipeline using the updatewcs task in the STWCS package. For ACS, non-polynomial distortion corrections are stored as FITS extensions that were inserted in the images during pipeline processing by the updatenpol task in DrizzlePac. Images retrieved from the Archive before AstroDrizzle was installed in the pipeline should either be re-retrieved, or processed using updatewcs, and for ACS, also with updatenpol, before running any DrizzlePac tasks.

6.2.1 Alignment Error Sources

Images may not be well-aligned for several reasons:

- Accurate Pointing repeatability
  - For the most part, commanded and actual telescope dither pointings in a single visit are highly accurate: about two to five milliarcsec. within an orbit, and five to twenty milliarcsec. for contiguous orbits that need guide star reacquisitions within a visit. (However, it is always useful to verify this by measuring the positions of a few objects in the `*single_sci.fits` images). See Appendix B
  - Images taken in different visits typically use different guide stars; since the positions in the guide star catalog have uncertainties as high as 0.2 to 0.5 arcseconds, it is very likely that the WCS from each visit will be misaligned at the same level.
  - On rare occasions, observations undergo loss-of-lock on guide stars, causing drifts and pointing offsets. A quick check of the keyword `QUALITY` (with details in the `QUALCOM*` keywords) in the image header will indicate if this anomaly occurred. If it did, it's best to discard bad observations and realign those that are still useful.
- The ability to get accurate centroids on objects like extended sources, targets obscured by dust, and faint objects with low signal-to-noise
- Long exposures that may suffer from blurring due to the changing velocity aberration of the telescope. Neglect of the velocity aberration correction can result in misalignments on the order of a pixel for WFC images taken six months apart for targets near the ecliptic. For further discussion of the effect of velocity aberration see the paper on "The Effect of Velocity Aberration Correction on ACS Image Processing proceedings" from the 2002 HST Calibration Workshop.

6.2.2 Processing Large Images

The same methods used to align and drizzle small images are also applicable to large mosaics and deep surveys. Every effort has been taken to ensure that drizzle algorithms are structured to provide the fastest computation and memory management. However, the user should consider limitations which exist due to size of the data, and the amount of memory available in the processing computer. Processing large datasets on a 32-bit system will be limited by the OS restrictions for addressing memory, so only two Gb of memory can be used for all the output arrays. This results in a limiting size of about 16000 x 16000 for the combined image, if no context image is generated. This limitation can be avoided by running the AstroDrizzle code on a 64-bit system where the OS can address much larger blocks of memory.

6.2.3 Using TweakReg for Image Alignment

The **tweakreg** task provides an automated interface for computing residual offsets for a group of flat-field calibrated images (`flt.fits`, `c0m.fits` for WFPC2, `flc.fits` for CTE-corrected ACS images) before they are combined by **astrodrizzle**.

Images are first aligned based on WCS information in the header. But if the images still remain slightly misaligned, they have residual offsets. This can occur when images are taken in different visits using different guide stars; residual shifts between the visits are due to uncertainties in the guide star positions (as much as 0.2 to 0.5 arcseconds). Smaller-scale residual offsets could also occur during guide star re-acquisitions for observations taken in a multi-orbit visit, (see Appendix B).

TweakReg is a WCS-based task, not pixel shift-based like MultiDrizzle; for images with residual offsets with respect to a reference image or catalog, WCS information in their headers are modified to "tweak" them to a common WCS with the reference image or catalog. In other words, TweakReg computes residual offsets that are used to update WCS header information in the images to put all images in a common coordinate frame.

Processing Steps Overview

TweakReg performs the following processing steps to determine a fit between each input image and a specified reference WCS:

- It builds a catalog of source positions for each input image using one of these modes:
  - Using a DAOFIND-like algorithm called *ImageFind* to detect stellar sources (the default mode)
  - Accepts user-supplied source catalogs
- The WCS from a reference image is selected from one of these options:
  - The first input image (the default mode)
  - An image specified by the user–this could be one of the input `flt.fits` files, perhaps an image that has the most overlap with other images. It could also be a different type of image, from another HST instrument or a different telescope
- A reference catalog of sources is identified using either:
  - A source catalog from the chosen reference image
  - A catalog of source positions on the sky (R.A., Dec.) provided by the user
- All source positions for the input `flt.fits` images, and reference source positions, are converted to X,Y positions in the reference WCS tangent plane using all available distortion corrections
- For each input image and reference image pair, the difference in source positions is represented in a two-dimensional histogram, allowing the determination of an initial offset based on the histogram peak in ΔX and ΔY
- Algorithm based on the **xyxymatch** IRAF task is used to match input source positions and reference image source positions using the initial offset

- For each input image, a fit to determine the most accurate offsets is performed on the matched sources lists[1]; at this point, the user may inspect the fit residuals for each input image, then re-run **tweakreg** with different parameter values until a satisfactory solution is obtained
- When the user is satisfied with the fit, **tweakreg** can be run a final time, but this time, the software can update input images with WCSs that put all images in the same coordinate frame
- A headerlet can also be (optionally) created from the updated input image WCS. More details on what a headerlet is, and how to use it can be found in Section 3.4.

Catalog Matching

A widely utilized method for computing offsets between images begins with identifying sources in each image. For each image, these sources are then matched with those in an overlapping section of the reference image or catalog, allowing offsets to be computed. This technique requires that each image contains recognizable sources, like point sources, that can be accurately identified and positionally measured by the software. There has to be enough overlap in each input-reference image pair so that enough real sources can be identified to calculate accurate offsets. TweakReg creates a catalog of source positions for each input image using an object identification routine similar to **daofind**. The user also has the option to provide his or her own source position catalogs for each input chip.

Input files can be passed to **tweakreg** in several forms:

- Filename of a single image
- Name of an association (`ASN`) table
- Wild card specification for files in directory (i.e., `*flt.fits`)
- Filenames, each separated by a comma
- An ASCII text file containing a list of input images, one per line, where the prefix "@" is specified before the file (i.e., `@file_list`)
- A Python list, if the task is run from the PyRAF/Python command line

When comparing the input images, the TweakReg software defines the reference frame either by:

- The first image from the list of input images
- A catalog derived from a reference image specified by the user
- A source catalog provided by the user

1. A matched sources list is a list of common sources found in an input image and the reference image or catalog.

The ***refimage*** parameter in **tweakreg** specifies the reference image from which the reference catalog was derived. This parameter allows the user to specify a reference image with a specific WCS solution that will define the coordinate frame to which all other input images will be transformed. If no reference image is designated in the ***refimage*** parameter, **tweakreg** will, by default, use the first image in the list of input images.

6.2.4 Aligning Subsampled Images

The source finding algorithm built into TweakReg has been optimized for point sources. The algorithm used to center on each source in the image works best with properly sampled PSFs, although it will work fairly well on the most strongly undersampled detectors on HST:WFPC2, NIC3, and WFC3/IR.

In Figure 6.1, the apparent positions of stars in two WFC3/IR images (`iabf01bxq` and `iabf01ckq`), which have been offset by a simple shift along the detector *X* axis of 24 arcseconds, exhibit some very systematic residuals of up to +/- 0.1 pixels. These residuals arise because typical centroiding and PSF-fitting applications tend to move the position of a star in an undersampled detector towards the center of the pixel in which the star is brightest. In order to avoid this bias, one must explicitly take the undersampling of the detector into account. One method for doing this is the *ePSF* (effective PSF) method of Anderson and King (2006). Figure 6.2 shows the residuals from the alignment of the same stars; in this case, instead of using the native position-finding algorithm in TweakReg, the star positions supplied to TweakReg were created by running the ePSF algorithm by Anderson and King.

At present the ePSF method is not included as part of the DrizzlePac software. However, it may be part of a future release, as an option in TweakReg (at a minimum for WFC3/IR). Nonetheless, the effect of not including the ePSF correction does not substantially bias the final fit as the oscillations one sees largely average out. The difference in the solutions with and without the ePSF correction is a shift of ~0.015 pixels.

Figure 6.1: Residuals from Aligning Two WFC3/IR Images Using tweakreg Source-Finding Algorithm

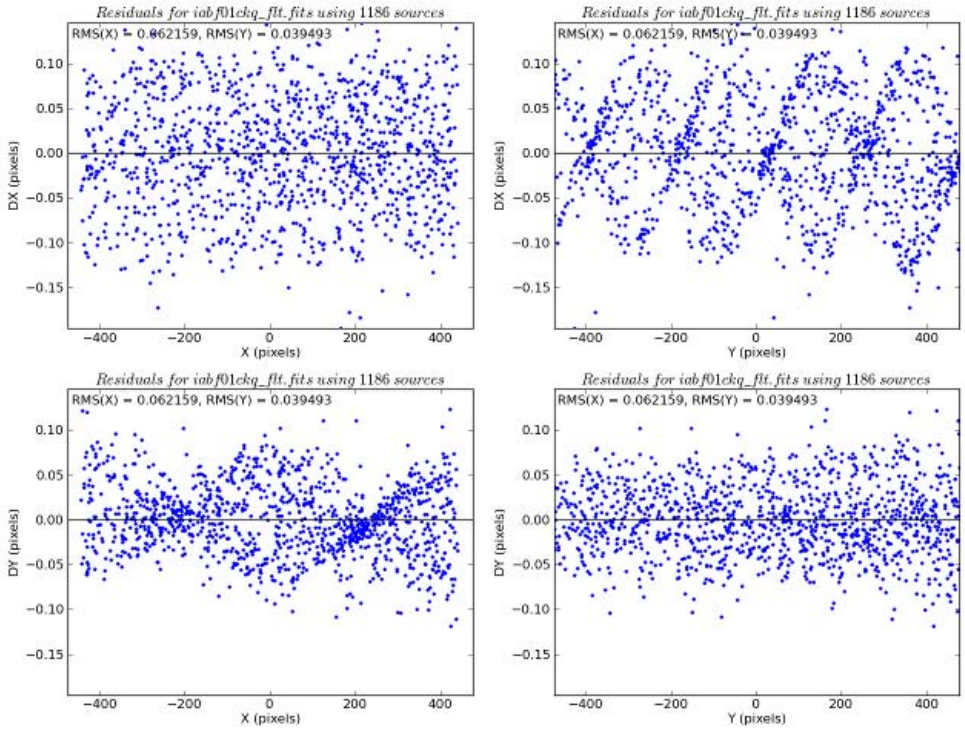
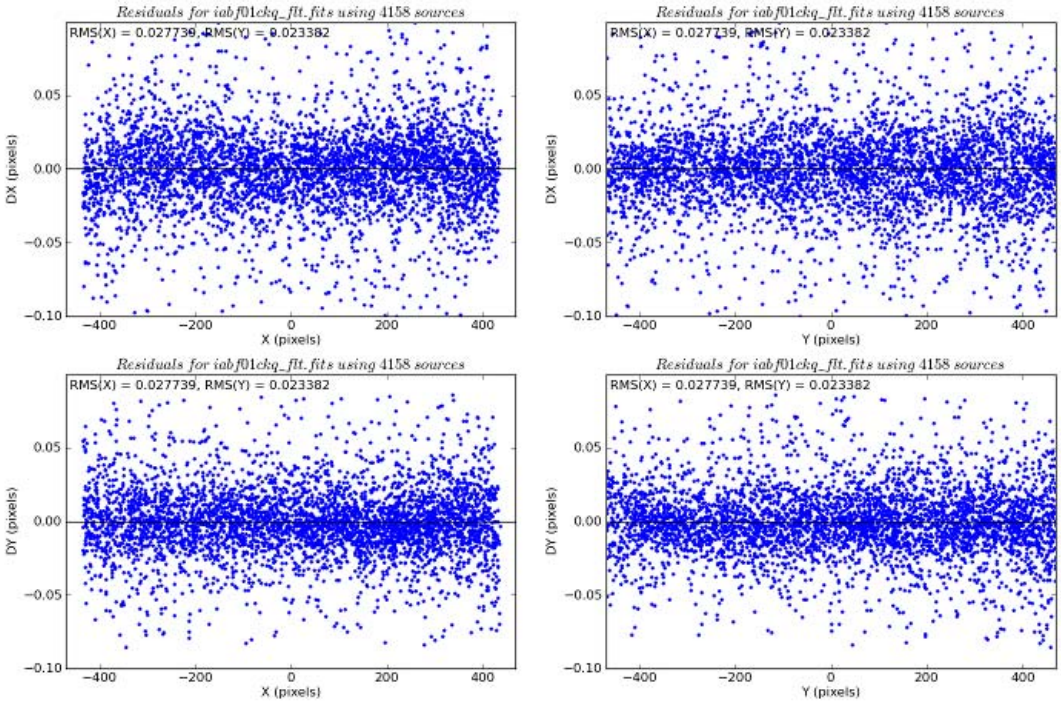


Figure 6.2: Residuals from Aligning two WFC3/IR Images with tweakreg Using Positions Determined with the ePSF Method



6.3 Running AstroDrizzle

Reprocessing images with AstroDrizzle requires consideration of the science to be performed on the images, as well as the field of view in the images, in order to determine the optimal set of parameters in **astrodrizzle**. This section provides some thoughts on key aspects of AstroDrizzle processing that can help guide the user in deciding on the best way to reprocess the images.

6.3.1 Sky Subtraction Considerations

Many astronomical fields of view cover parts of the sky devoid of any large objects and as a result the default sky subtraction performed by AstroDrizzle will generally work well enough without much modification. Incorrect sky subtraction by AstroDrizzle can slightly bias the cosmic ray identification but more noticeable result in each input image showing up as a tile with distinct edges in the final combined output image. Sky subtraction will be turned off in the pipeline when processing any narrow-band exposures as they typically do not detect enough background to affect the final drizzle product.

The sky subtraction step can also be biased by the presence of large extended sources in the field of view. If the extended source does not cover the entire field of view, it may be possible to simply change the sky computation to use the ***mode*** instead of the default clipped ***median*** as specified by the ***skystat*** parameter. Any observation where no true background (sky) pixels have been observed due to the presence of an extended source filling the field of view will almost certainly require that the sky subtraction step be turned off.

astrodrizzle does, though, support a mode where the user can either compute a custom value for the sky or perform sky subtraction prior to running AstroDrizzle and still account for an average sky value. A custom sky value can be computed and added to each image's science (SCI) header as a new keyword of the user's choosing. The name of this keyword can then be provided to **astrodrizzle** through use of the ***skyuser*** parameter with sky subtraction turned on. This value will then be used by **astrodrizzle** when performing cosmic ray identification but will not be applied during drizzling as is usually done with sky subtraction. These custom value will then be copied into the `MDRIZSKY` keyword as a record to indicate it was used during processing. This allows the user to apply their own custom sky-subtraction.

The user may find he or she wants a record of the custom sky subtraction values applied to each image, in which case, a file with a custom sky value for each chip could be written to a file. This file, named by the ***skyfile*** parameter, would also allow the user to specify whether or not these values have been used to sky-subtract the input images prior to processing by **astrodrizzle**. If they were not applied, **astrodrizzle** would use the values from the file for sky subtraction when drizzling the data. More details on the use of the ***skyuser*** and ***skyfile*** parameter can be found in Section 4.2.3.

The precise value chosen for the sky is rarely a concern; in general one will subtract the sky from around an object when doing photometry on it. What is crucial in Drizzle is that the sky from the various images be the same. This is because data from the images being combined goes into different pixels with different weights. Therefore, if the sky values of the input images are not the same, the drizzling can create artificial sky noise. If a user does the sky subtraction by his or herself, then she or he must be sure the images are left with sky values that match.

6.3.2 Cosmic Ray Rejection

As described in Section 4.2, AstroDrizzle drizzles each of the input images separately onto the same output frame and then creates a median of these images to produce an image largely free of bad pixels. This median image is "blotted" or interpolated back to the frame of each of the input images and compared with the inputs to determine the locations of bad pixels which are not included in the pixel masks created by the instrument groups.

By default, the pipeline drizzles both the intermediate images and the final image onto an output frame with pixels that are the same size as the input pixels. In most images, the vast majority of these bad pixels are caused by cosmic rays. Using a finer output pixel scale than the default can often given a better interpolation, and thus a more accurate cosmic ray removal. Therefore the same rules that are discussed in the next section for choosing a final pixel scale should be considered for choosing the pixel scale in the cosmic ray removal step. Choosing a small "pixfrac," however, is less crucial, and indeed less desirable, as a larger "pixfrac" gives more images from which to determine a median. So in general, the user may wish to keep ***driz_sep_pixfrac = 1.0***.

Cosmic ray rejection in the pipeline also relies on the original header astrometry. If that astrometry is not accurate (to about 0.1 pixels or better) then the cosmic ray rejection will be compromised, and in particular, pixels in stars may be incorrectly marked as cosmic rays. One way to check to see if this has occurred is to compare the data quality and image extensions from the same `flt.fits` file. If pixels on stars (usually the brightest pixel) are frequently marked as bad, then the pipeline had a problem with the dataset. Correctly aligning the images and using a smaller output scale will usually solve this problem. In some cases the user may find it necessary to adjust the ***driz_cr_scale*** and ***driz_cr_snr*** parameters. These adjust how sensitive the cosmic ray rejection algorithm is to differences between the blotted median and the image. In particular, ***driz_cr_scale*** is a fudge factor that multiplies the local derivative of the blotted image. This is added to the calculated statistical noise in the pixel to create a new (larger) estimate of the noise, making it less likely that a pixel will be marked as bad.

Image Alignment Requirement

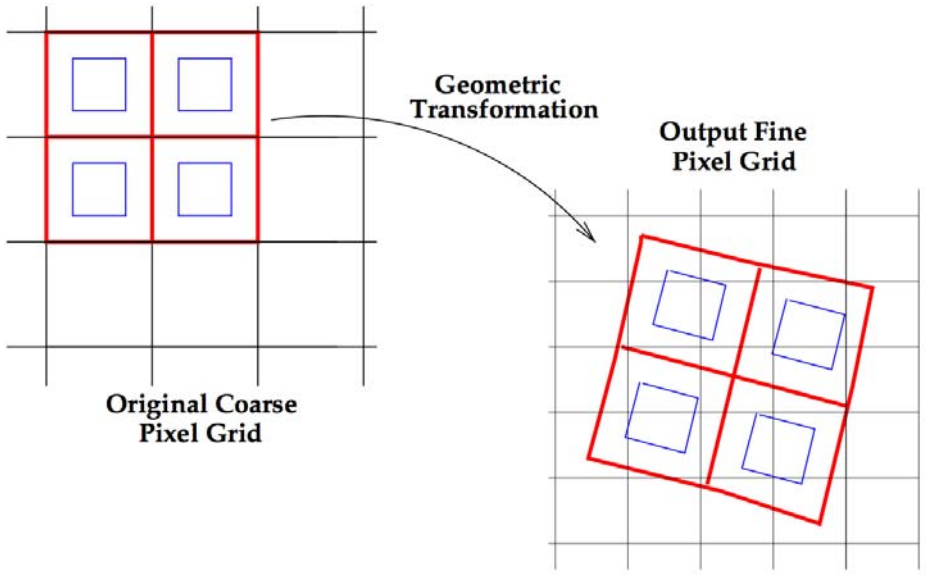
A major complicating factor in accurately determining which pixels are affected by cosmic rays remains the alignment of the images. Any misalignment of a star from one image to the next by more than 0.1 pixel will result in it being flagged as a cosmic ray. This level of error can result from pointing errors or inaccurate distortion models, both of which must be addressed prior to accurate identification of cosmic rays. The images generated in the first step of this process, the images drizzled to the final output WCS, can be used to verify the alignment and accuracy of the distortion correction for each of the input images. Any processing which cross matches sources from one image to the next can be used to compute the offset (shift) between the images. The task **tweakreg** demonstrates how these images can be used to determine the shifts using catalog matching of sources.

6.3.3 Selecting the Optimal Scale and Pixfrac

In combining images, the Drizzle algorithm maps pixels from input images into pixels in a subsampled output image, taking into account shifts and rotations between input images, as well as the geometric distortion of the images. However, to avoid re-convolving the output image with the large pixel "footprint" of the input images, Drizzle allows users to "shrink" the input pixel before it is assigned to a location in the output image.

The shrunken pixels, called "drops," can be imagined as raining down on the sub-sampled output image. The input pixel's value is applied to an output pixel with a weight proportional to the area of overlap between the "drop" and the output pixel.

Figure 6.3: Transformation of Input Pixel Grid to Output Pixel Grid in Drizzle



A schematic representation of Drizzle. The input pixel grid (shown on the left) is mapped onto a finer output grid (shown on right), taking into accounts shift, rotation and geometric distortion. The user is allowed to shrink the input pixels to smaller pixels, called "drops" (faint inner blue squares). A given input image only affects output image pixels under drops. In this particular case, the central output pixel receives no information from the input image.

Subsampling in the output image is set in **astrodrizzle** using the ***final_scale*** parameter, which is in units of arcseconds. The drop size is controlled in **astrodrizzle** by the parameter ***final_pixfrac***, which is the ratio of the drop's linear size to the input pixel's linear size before geometric distortion corrections.

There is no single setting of these parameters that is optimal for all observations and scientific goals. If one has only a few images, one will want to use a large "pixfrac" (close to or equal to one) to make sure that all of the output image is well covered. Even in this case, however, an output ***final_scale*** which is smaller than the input pixel is usually preferable. (An exception may be the single image case, where one is primarily removing distortion from the image. In this case one may want to set both ***final_pixfrac*** and ***final_scale*** to the original image size and use the ***Lanczos*** kernel. This kernel is very good in preserving the PSF; however it introduces strong artifacts around bad pixels and cosmic rays).

When a user has few–two or three–images, but has used one of the standard dither patterns which does an optimal subpixel dither, he or she can use an output pixel scale that is one half (in linear size) of the input pixel. The ***final_pixfrac*** should probably be kept at greater than 0.7 in order to insure good coverage of the output.

One way to test that one has good final coverage of the output pixels is to examine the final weight map and make sure that the ratio of the standard deviation of the weights over the weights in a typical area of the image does not go significantly below 0.3. As users generally do not employ the weight map in final photometry, this insures that ignoring the weight map will not significantly increase the noise of the final result.

Using an even finer scale or pixfrac may be advantageous when one has large numbers of images and one wishes to obtain the best possible SNR on point sources (the PSF is convolved in quadrature with both the final scale and pixfrac). Thus, the Hubble Ultra Deep Field (Beckwith et al. 2006) used a scale equal to 0.4 of the original pixel and a pixfrac of zero! Since the addition of the effect is in quadrature, however, and the original pixel and PSF are substantially larger than the pixfrac, the main effect of using such a small pixfrac is to entirely eliminate correlated noise (See Section 2.3.2). However, using a smaller pixfrac also reduces the ability of the eye to detect low surface brightness features in the image.

In practice many users may find convenience outweighs perfection. In particular, using an output pixel size of 0.03333 arcseconds for ACS and WFC3 is generally sufficiently small to give fine resolution but has the benefit that three pixels are just about 0.1 arcseconds across. A final output pixel size of 0.06666 arcseconds for the WFC3/IR is closer to one half of an original pixel and has the advantage that three pixels are 0.2 arcseconds across. Again users with four or fewer dithers will probably want to keep their pixfrac >~0.7, but users should look at the examples in Chapter 7 for more discussion and ideas on choices of pixfrac and scale, and are free to experiment. To summarize, when experimenting with pixfrac and scale users should keep these points in mind:

1. For subpixel dithered data, select an output scale that's smaller than the native scale. It will even help in the cosmic ray rejection step.

2. A smaller ***final_pixfrac*** gives higher resolution and lower correlated noise, but also reduces sensitivity to low-surface brightness features (though it is possible to convolve a high resolution image later to go after low surface brightness features).
3. Keep the standard deviation of the weight map over the main part of the image to above ~0.3 of the mean to insure that one does not lose significant signal-to-noise in ignoring the weight map in final photometry.

6.3.4 Controlling the Bit Mask

Data Quality Flags

Data quality flags which were set during image calibration can be used as bit masks when drizzling. These specific DQ flag values are unique for each detector and are defined in the instrument Data Handbooks.

AstroDrizzle will mask all pixels with non-zero DQ flag values, unless otherwise specified. When a pixel is flagged in one image but has a corresponding un-flagged (good) pixel in any other image in the stack, that pixel will be replaced with the "good" value during image combination. Otherwise, the pixel will be replaced with a "fill value" specified by the user. (If the "fill value" is set to ***INDEF***, and if there are no "good" pixels in the stack, the pixel will retain its original value but show zero weight in the drizzled weight image.)

While the calibration pipeline assigns DQ flags to a large fraction of pixels, the science quality of those pixels may not in fact be compromised. Choice of which pixels to treat as "good" or "bad" depends largely on the number of images being combined and the dithering strategy. The user may override flags in the DQ array by specifying which bit values should be considered as "good" for inclusion in the final image combination. Control over these DQ flags is specified by parameter values in both the "single-drizzle" step (Section 4.2.4) and in the "final drizzle" step (Section 4.2.8). (These parameters tell AstroDrizzle which "suspect" pixels in the DQ array to keep.) This is particularly useful when only a handful of images are combined and excessive flagging compromises the final product. The drizzled weight image should be carefully examined to get an idea of how many input pixels contributed to each output pixel. When the weight image has a significant number of "holes" where no valid input pixel was available, the user may need to re-evaluate the stringency of selecting which DQ bits should be considered as "good" and adjust the parameters accordingly.

During pipeline processing, values for ***driz_sep_bits*** and ***final_bits*** are specified in the `MDRIZTAB` reference file. For ACS/WFC or WFC3/UVIS images, for example, AstroDrizzle assumes all DQ flags are bad except bits 64 and 32 (warm pixels and CTE tails of hot pixels in the superdark used to create the dark reference file). During

manual reprocessing, users may want to consider treating other DQ flags as good pixels. For example, for the ACS CCDs and the WFC3 UVIS detector, when the total counts in a given pixel have exceeded the full well, those pixels are flagged with a value of 256. However, since counts still accumulate in a highly linear manner, photometry of saturated stars is quite practical if using a gain that samples the full well depth. (For details, see WFC3 ISR 2010-10 and ACS ISR 2004-01). In this case, the user may want to define 256 as a "good" bit so that those pixels will not be rejected while drizzling.

Because AstroDrizzle was designed for use with multiple instruments, the default value for the ***driz_sep_bits*** and ***final_bits*** parameters is set to zero for offline reprocessing. This assumes that all pixels flagged in the DQ array are "bad" pixels and will be masked. When only a few images are being combined, this may be an overly aggressive approach. The user can decide the best strategy for assigning these parameters based on the number of input frames and the dither pattern used. Ultimately, one wants to avoid having too many pixels with no "good" input pixel in the stack of images used to create the final product.

Note that in **multidrizzle**, these parameters were specified as an integer sum of the DQ flags to be treated as "good" pixels. In **astrodrizzle**, they may be given as either the sum of those DQ flags or as a comma-separated list, as shown in the examples in Chapter 7.

Drizzled Masks

While running AstroDrizzle, several sets of masks are created for each image during processing.

The first is called the "single" mask, where every pixel flagged in the input image DQ array is assigned a value of zero. All other pixels are assumed to be good and will be assigned a value of 1. The single mask image is used in the "driz_separate step" (Section 4.2.4) to create the `*_single_sci.fits` images which are used as input for creating the median image. The user may tell **astrodrizzle** to ignore specific input image DQ flag values by specifying them in the parameter ***driz_sep_bits***.

The second mask image is called the "final" mask. The final mask is used in the "driz_combine" step (Section 4.2.7) to create the final drizzled product. The parameter ***final_bits*** allows the user to tell **astrodrizzle** which input image DQ flags to ignore for the final image combination. When this parameter is left to the default value of zero, every pixel flagged in the input image DQ array is assigned a value of zero in the mask.

These two mask files are created during software initialization and are subsequently updated when running **astrodrizzle** steps to compute the static mask (Section 4.2.2) and the cosmic ray masks (Section 4.2.7). The static mask is computed to correct unusually low pixels which were over-subtracted when applying the dark image calibration. (Note that this mask is not written to disk but retained in memory only.) The static mask is combined with the original single and final masks to create updated versions of these images.

The cosmic ray mask is computed (Section 4.2.7) when the parameter ***driz_cr_corr=yes***, and is written to a file called `*_crmask.fits`. The cosmic ray mask should be inspected and blinked with the original input image to verify the

quality of the rejection. Note that during pipeline processing, **astrodrizzle** flags potential cosmic rays with a value of 4096 in the input image DQ array when combining data which is part of an association. During reprocessing, the parameter ***resetbits*** may be used to reset the input image DQ bit value from 4096 to zero (designating them as good pixels), so that cosmic rays can be re-identified using user-specified parameter values in **astrodrizzle**. The specific value of the flag may be defined via the ***crbit*** parameter, if desired. These new optimized cosmic ray flags will be used to update the DQ array of the original input image (`flt.fits` or `flc.fits`).

During the final drizzle combination (Section 4.2.8), the static mask and the cosmic ray mask are combined with pixels selected as "bad" in the input `flt.fits` DQ arrays (the ***final_bits*** parameter specifies which input image DQ pixels should be treated as "good.") The resulting "master" mask is then used during the final image drizzle-combination step to create the drizzled product `*_drz.fits`.

CHAPTER 7:

Examples: Using DrizzlePac for Combining Images

In this chapter…

Introduction

Data and related files for the examples in this chapter are available at the DrizzlePac website. Please check the website latest news and additional resources for using DrizzlePac.

This chapter contains five examples covering a variety of observation types that may likely be encountered by users. These include data analysis goals such as creating combined distortion-free images, optimizing image resolution for subpixel dithered data, improving cosmic ray removal, and aligning images taken using different filters or detectors.

Table 7.1 contains a summary of the examples in this chapter. Users who are unfamiliar with drizzling are encouraged to try the first example (Section 7.1) where each processing step is accompanied by detailed explanations.

A special note to users working with older data

ACS and WFC3 data retrieved before AstroDrizzle was installed in the Archive's on-the-fly recalibration (OTFR) pipeline have to be updated to insert astrometric information needed to run tasks in the drizzlepac package. Section 4.4 has detailed information on how to update older WFC3, ACS, and WFPC2 images.

Since AstroDrizzle became part of the OTFR pipeline, ACS and WFC3 products now contain significant changes geared towards simplifying the handling of astrometric information. Distortion correction reference files that used to be required for running MultiDrizzle are now used behind-the-scenes in OTFR processing. In the pipeline, AstroDrizzle extracts distortion correction information that is relevant to the images from reference files and inserts them directly into the data file headers.

Distortion corrections that are now incorporated in the new pipeline products can be described as follows:

1. Linear distortion corrections, including velocity aberration and detector-related distortions, are built into the *CD matrix*
2. Coefficients for distortion corrections that can be expressed as higher order polynomials are now stored as values to *SIP header keywords*
3. Non-polynomial distortions for ACS images are stored as *look-up tables* attached to the image file as FITS extensions

Section 4.4 has detailed information on how to update older WFC3, ACS, and WFPC2 images. Users who wish to update their data should retrieve the relevant distortion correction reference files named in the image header keywords in order to manually update their data.

Users trying these examples on their home machines may notice small differences in their results compared to the example results archived in the DrizzlePac website. This could be due to different software versions since these examples were prepared in the test

Table 7.1: A Summary of the Examples in this Chapter

Example	Goal	Data Description	drizzlepac Tasks
Section 7.1 WFC3/UVIS: Optimizing Image Alignment for Multiple Visits	Align and combine three WFC3/UVIS images taken at different visits and orientations.	Three full-frame UVIS F606W images of Omega Centauri, each with exposure time of 40 sec., each taken in a different visit with different roll angle and offset.	**tweakreg astrodrizzle**
Section 7.2 WFC3/IR: Optimizing Image Sampling for a Single Visit	Optimize image sampling, using ***final_pixfrac*** and ***final_scale*** settings in **astrodrizzle**, for WFC3/IR images.	Spiral galaxy NGC 3370 in F160W, observed with a `WFC3-IR-DITHERBOX-MIN` pattern. All observations in the same visit with same orientation.	**astrodrizzle**
Section 7.3 ACS/WFC: Optimizing Image Alignment for Multiple Visits	Align and combine three ACS/WFC images taken at different visits and orientations.	Three F606W images of the dense star field NGC 104, each with exposure time 339 sec. The images are CTE-corrected flat-field calibrated files with suffix `flc.fits`.	**tweakreg astrodrizzle**
Section 7.4 ACS/WFC: Optimizing Image Sampling for a Single Visit	Optimize image sampling, using ***final_pixfrac*** and ***final_scale*** settings in **astrodrizzle**, for ACS/WFC images.	A two-point dither line with subpixel dithers at each point: four images of galaxy NGC 4449 taken with the F555W filter. The images are CTE-corrected flat-field calibrated files with suffix `flc.fits`.	**astrodrizzle**
Section 7.5 WFC3/UVIS: Aligning Images in Different Filters by Defining an Output Reference Frame	Two sets of drizzled WFC3/UVIS images, each for a different filter, are aligned by WCS and in pixel space.	Observations of galaxy M83 in F814W and F438W, each filter set taken in different visits. There are three images per filter.	**tweakreg tweakback astrodrizzle**

Data and associated files for all examples in this chapter are available at the DrizzlePac website.

7.1 WFC3/UVIS: Optimizing Image Alignment for Multiple Visits

Introduction

This example demonstrates the use of the **tweakreg** and **astrodrizzle** tasks to align and combine three UVIS images, each taken in different visits at different orientations.

There is a great deal of explanatory text in this example; users new to DrizzlePac are encouraged to start here.

Summary of Steps

1. Description of the data
2. The images are aligned to the coordinate frame of a reference image using the **tweakreg** task. In this example, one of the three images has been selected as the reference image.
3. After the three images have been aligned to a common WCS, they're drizzle-combined using AstroDrizzle.

7.1.1 Description of the Data

In this example, three single full-frame WFC3/UVIS images of the globular cluster Omega Centauri (WFC3/CAL Program 11911), each with exposure time of 40 seconds, taken in F606W, will be combined using **astrodrizzle**, after the images are aligned using **tweakreg**. These observations were acquired over a 6-month period in three separate visits at different orientations and offsets.

Table 7.2: Summary of Images in this Example

Image Name	Association ID	Proposal ID	Visit & Line Number	Pointing[1] (degrees)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
ibc301qrq_flt.fits	NONE	11911	01.001	201.6821267139 -47.46661075341	105.0002	2010-01-14	40.00
ibc304v3q_flt.fits	NONE	11911	04.001	201.7121069398 -47.47292818852	199.9971	2010-04-29	40.00
ibc307qyq_flt.fits	NONE	11911	07.001	201.7051092234 -47.49082103744	279.9995	2010-07-04	40.00

1. Original CRVAL1, CRVAL2, in degrees, at the reference pixel in [sci,1]

7.1.2 Aligning Images with tweakreg

In order to correctly align and combine multiple images into a distortion-free final drizzled product, AstroDrizzle relies on the World Coordinate System (WCS) information stored in the header of each image.

WCS information for an image is tied to the positions of guide stars used for the observation. As a result, if these three images were aligned based on the original WCS information in their headers, they would still have small offsets due to the uncertainty in guide star positions that can be as large as 0.5 arcseconds (12 pixels).

The TweakReg software can align a set of images to better than subpixel accuracy (< 0.1 pixel). For each image, it computes the geometric transformation (shift, rotation, and scale) after correcting for geometric distortion relative to a reference image based on positions of stars common to both images. That transformation is used to update the WCS information for the image.

Like other **drizzlepac** tasks, **tweakreg** can be run several different ways: using Python syntax in either the PyRAF or Python environments; setting parameter values using a TEAL GUI[1]; using a preset parameter configuration file (with suffix `.cfg`) that can be loaded into the TEAL GUI, or run using Python directly to call the **tweakreg** module in the **drizzlepac** package.

In the example below, **tweakreg** is run as a Python command in PyRAF. First, the **drizzlepac** package is imported into the Python environment, followed by importing the **tweakreg** task from the **drizzlepac** package. (**imagefindpars** does not need to be imported because it is automatically called from within the **tweakreg** task.) **tweakreg** and **imagefindpars** are then reset to default parameters (in case non-default parameters were used in a previous run of the tasks). Finally, **tweakreg** is run with two non-default parameters values: ***threshold*** and ***peakmax***.

```
--> import drizzlepac
--> from drizzlepac import tweakreg
--> unlearn tweakreg
--> unlearn imagefindpars
--> tweakreg.TweakReg('*flt.fits',updatehdr=False,conv_width=3.5,\
threshold=2000,peakmax=50000)
```

tweakreg is typically run several times until the best fit is found for the image offsets. When the user is satisfied with the results, **tweakreg** is run a final time with ***updatehdr=yes*** so WCS information in the images can be updated using the calculated transformations required to align the images.

Listed below are some **tweakreg** parameters, as well as parameters from the **imagefindpars** task called by **tweakreg**, that are directly related to this example,

1. Running **tweakreg** from the TEAL GUI is helpful for several reasons: (1) task parameters, with descriptions are displayed. (2) Quick access to help files. (3) Task settings can be saved to uniquely-named configuration files (suffix `.cfg`) for later use or for record-keeping purposes. (4) Previously-created configuration files can be loaded for execution.

To set parameter values and run **tweakreg** in TEAL, type "`epar tweakreg`" in PyRAF. Don't forget to click the "ImageFindPars" button to open the **imagefindpars** TEAL window, and after setting those parameters, click "Save and Quit" to return to the **tweakreg** window to continue setting those parameters.

including those with non-default values. (Those not mentioned use default values.) The parameters are broadly grouped into three categories: logistics, source detection, and source matching.

Logistics Parameters in tweakreg

input: images that need to be aligned. For this example, input files are expressed as "`*flt.fits`".

refimage: name of the reference image. Input images will be aligned with respect to the reference image. For this example, the default setting is used, which means that the reference image is the first image in the set.

updatewcs: the default setting, ***no***, is retained because the **updatewcs** task was performed in the pipeline.

writecat: the default value, ***writecat=yes***, is kept so that source coordinate lists are created for each image using an algorithm similar to IRAF's **daofind**.

updatehdr: this switch determines if the geometric transformation solution for aligning each image, computed by **tweakreg**, should be used to update WCS information in each image header. It's highly recommended that this parameter be set to its default value, ***no***, until an optimal alignment solution is found for all images. This way, **tweakreg** can be run several times to optimize image alignment without any undue modification of header WCS information. When the best alignment solution is found, rerun **tweakreg** with this parameter set to ***yes***.

headerlet: when set to ***yes***, the newly-updated WCS information is written to a headerlet. More information on headerlets is available in Section 3.4. For the purposes of this example, the default value, ***no***, is used.

shiftfile: this shift file is in the same format as that used by **multidrizzle**, but it is only generated for informational purposes because offsets determined by **tweakreg** are used to directly update the WCS of each input image that will then be combined using **astrodrizzle**. The default setting, ***no***, is used for this example.

Source Detection Parameters in imagefindpars

Parameters for source detection are in a task called by **tweakreg**, called **imagefindpars**. Optimal parameters may vary on a case by case basis. In general, the best way to determine the best combination of parameters is to explore the parameter space of these values until an adequate number of sources are found in each image. A few hundred sources (or more) is ideal, but accurate solutions may also be obtained using only a few dozen sources matched between images.

threshold: the threshold above the background for source detection, in units of sigma (standard deviation). The default value is 4. But for this example, previous tests determined that a value of ***2000*** yielded good results.

peakmin and ***peakmax***: these cutoffs allow the user to reject sources that are too faint or too bright, based on the peak value of the source. ***peakmin*** and ***peakmax*** can be used to reject saturated sources, and bright and/or saturated cosmic rays. For the purposes of this example, ***peakmin*** is left blank (default setting), but ***peakmax*** is set to ***50000***.

conv_width: the convolution kernel width in scale units is used to identify potential sources in the image which look closest to a gaussian with the FWHM equal to this

value. The value is typically twice the FWHM of the point source. For this example, the default value, ***3,5*** is used.

skyigma: standard deviation of background in counts. Since ***computesig*** was set to ***yes***, this parameter is not used.

computesig

Automatically compute sigma for all inputs?

The default value, ***yes***, is used for this example. **tweakreg** automatically computes a value for sigma for each image, according to the following formula:

```
1.5 * imstatistics(image,nclip=3,fields='stddev')
```

(where **imstatistics** is an IRAF task, although a Python version of **imstatistics** actually gets used, not the IRAF task).

The computed sigma will override any user-specified value of sigma in the parameter ***skysigma***.

fluxmin* and *fluxmax minimum and maximum total source flux.

These are cutoffs that allow the user to reject sources that are too faint or too bright based on the integrated flux in a circular region. The radius of that circular aperture is the value of the ***conv_width*** parameter. For this example, it was not necessary to set values for these parameters.

Source Matching Parameters in tweakreg

searchrad

The search radius for a match.

tweakreg uses this search radius for finding matched sources common to an image and its reference image. For this example, the value ***1*** (arcseconds), the default, is used. If the value is too small, an optimal fit solution will not be found because the residual offset between the images is larger than the search range. If the value is too large, **tweakreg** will take longer than needed to find an optimal fit solution. A larger search space also includes more sources, so there's an increased risk of finding false matches which can affect the final fit solution.

searchunits

Units for search radius.

The choice is pixels or arcseconds, the latter being the default that's also used in this example. (Note, the WFC3/UVIS plate scale is 0.04 arcsec./pixel, so a search radius of one arcsecond (the default) translates to 25 pixels.)

use2dhist and ***see2dplot***

Use and display a two-dimensional histogram to find initial offsets for each image.

In this example, the default values, ***yes***, are used for both parameters. This creates a plot of the initial offset, as seen in Figure 7.1

fitgeometry

Fitting geometry.

There are two choices: ***shift*** and ***rscale***. The default, ***rscale***, is used here. It fits for *x* and *y* shifts, as well as rotation and scale. The ***shift*** option only fits for *x* and *y* offsets, with rotation assumed to be zero and scale factor assumed to be one.

residplot

Plot residuals from fit?

For the default parameter, ***Both***, a residual plot and vector plot are displayed for each image. The vector plot, which shows source magnitude with direction of residuals plotted as a function of location, is useful for spotting localized systematic deviations in image alignment. A good image alignment results in a vector plot that appears as small randomly-oriented vectors with no clear organized flows or structures. The residuals plot, which plots the *x* and *y* components of the residuals vs. *x*- and *y*-axis position, is useful for spotting subtle and/or larger scale and alignment issues. Good residual plots are flat with typical RMS values less than 0.1 pixels.

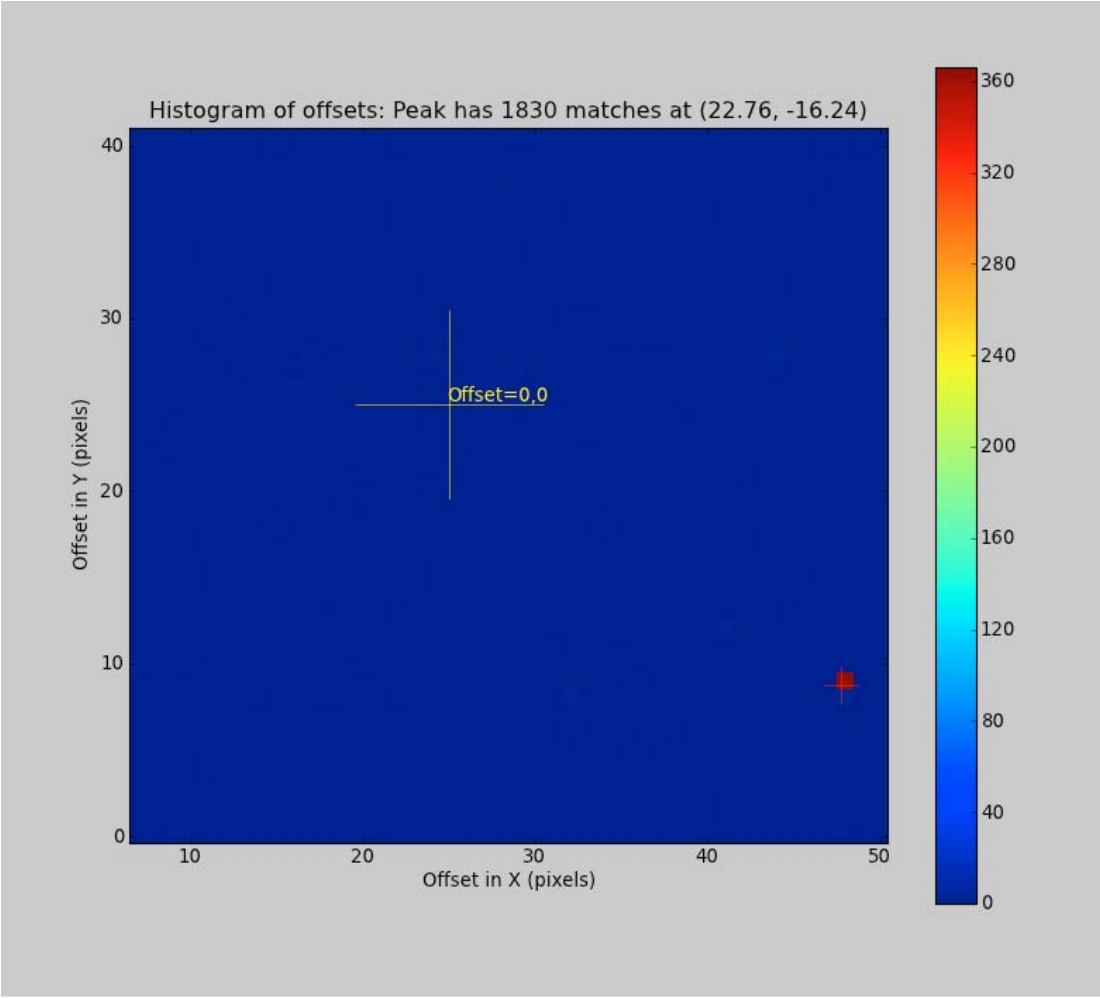
When **tweakreg** is executed, it computes the geometric transformation between the non-reference images and the reference image. In this example, `ibc301qrq_flt.fits` is the first image on the list and is selected as the reference image. Offsets are computed between it and each of the two remaining images in the set, `ibc304v3q_flt.fits` and `ibc307qyq_flt.fits`.

For each image-reference image pair, **tweakreg** will first display the two-dimensional histogram (Figure 7.1) for the initial shift and wait for the user to confirm that the guess is acceptable. After confirmation, it will display the residual plots (Figure 7.2) and the vector plot (Figure 7.3), as well as the fit solution for the first pair of images. The user will be prompted to either proceed to the next set of plots (by hitting the return key), or quit running **tweakreg**. If the user elects to continue, the same plots will be displayed for the next image-reference image pair in the image set.

While using **tweakreg** with different parameter settings to find the optimum fit for image offsets, the parameter ***updatehdr*** is set to ***no***, meaning that the fit solutions computed by **tweakreg** are not applied to the WCS information in the image headers.

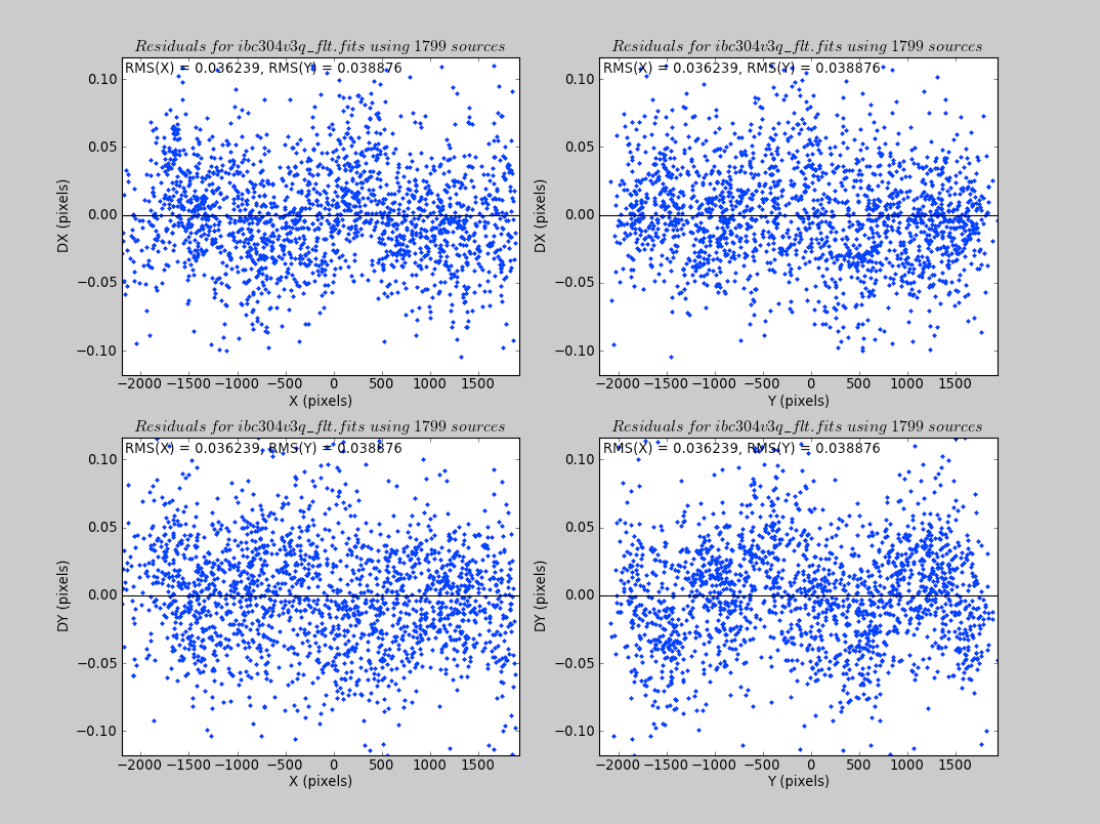
When a good fit is found using parameter values that work best for this example, **tweakreg** gets run one last time with the ***updatehdr*** parameter set to ***yes***. This not only gives the user one last chance to verify that the parameters produce acceptable residuals, but applies the determined fit to the headers of each of the images. The updated headers then get used by **astrodrizzle** in the next step to combine the now properly aligned images.

Figure 7.1: Two-dimensional Histogram Plot Generated by tweakreg.



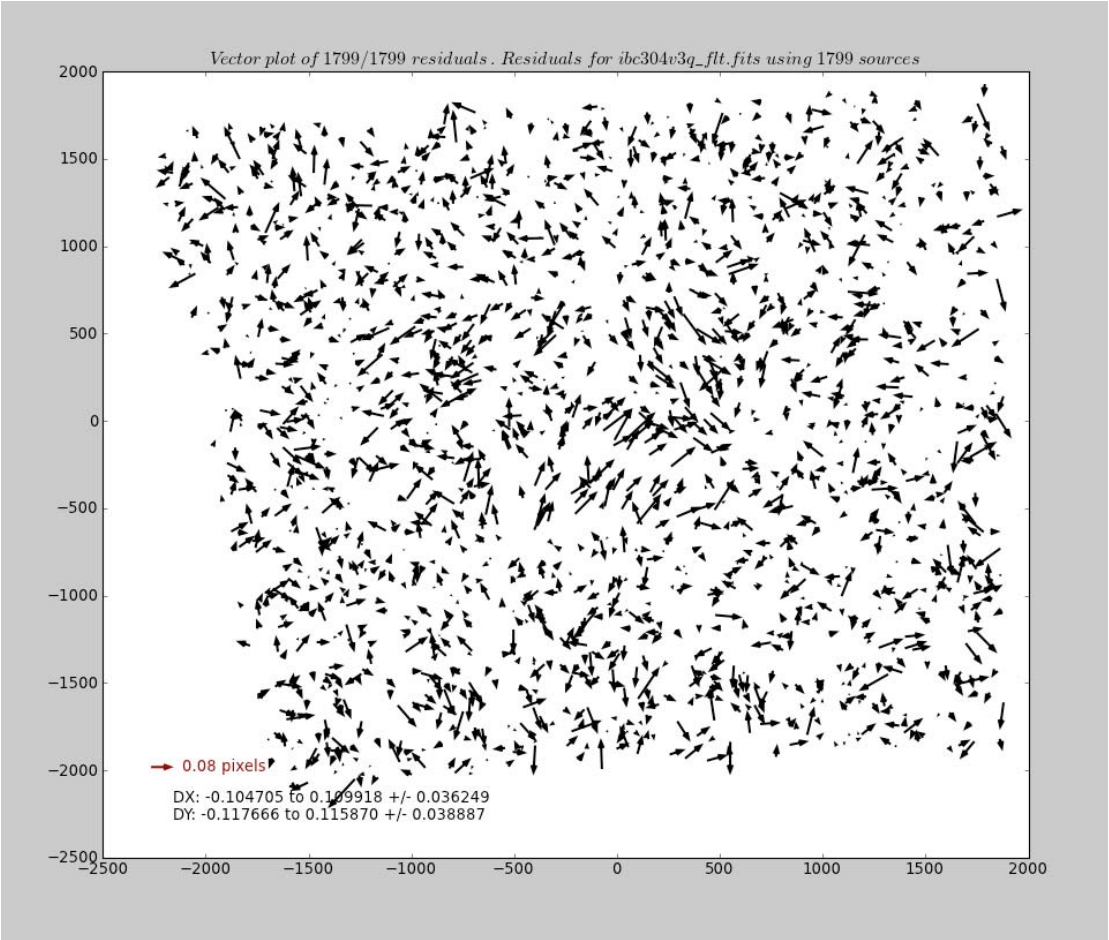
Two-dimensional histogram generated during the alignment of `ibc304v3q_flt.fits` with the reference image `ibc301qrq_flt.fits`

Figure 7.2: Residual Plots Generated by tweakreg



Residual plots from the alignment of `ibc304v3q_flt.fits` with reference image `ibc301qrq_flt.fits`. Wavy patterns seen in these plots are caused by lithography patterns (artifacts of the CCD chip manufacturing process) of the detector. The detector lithography patterns are not yet fully accounted for in the distortion correction solution. This demonstrates that the fit solution computed by tweakreg is so good that it is actually being limited not by any factor associated with tweakreg itself, but by the accuracy of the distortion solution for the detector.

Figure 7.3: Vector Plot Generated by tweakreg



Vector plot generated during the alignment of `ibc304v3q_flt.fits` with reference image `ibc301qrq_flt.fits`.

7.1.3 Combining the Images using astrodrizzle

Now that the images are aligned to a common WCS, they are ready for combination with AstroDrizzle. In a nutshell, AstroDrizzle processes a set of pipeline standard calibrated files (with suffix `flt.fits`), and also CTE-corrected calibrated files (with suffix `flc.fits`). CTE-calibrated files are currently available for ACS, and will be eventually available for WFC3 data as well.

The **astrodrizzle** steps are summarized below. A more detailed description is available at Section 4.2.

- Each input image is aligned, based on WCS information in its header, to a reference image.
- A static pixel mask is created to flag bad detector pixels.
- Sky subtraction is performed on working images (not the original images)

- Each image is individually drizzled, with geometric distortion corrections, to a common reference frame.
- The distortion-free drizzled images are combined to create a median image.
- The median image is blotted, or reverse-drizzled, back to the frame of each input image.
- By comparing each `flt.fits` image with its counterpart blotted median image, the software locates bad pixels in each of the original frames and creates bad pixel masks (typically cosmic rays and faulty pixels in the detector).
- In the final step, `flt.fits` images with updated WCS information from **tweakreg** are drizzled together, using the mask files, onto a single output image.

Running **astrodrizzle** using default parameter values is not recommended, as these defaults may not provide optimal science products. Users are strongly advised to inspect the quality of the sky subtraction and cosmic ray rejection, and to experiment with the output ***final_scale*** and ***final_pixfrac*** parameters in the final drizzle step. Information on evaluating the final products is available in Chapter 8.

The following sections will walk the user through each of the seven AstroDrizzle processing steps.

Note that the software may be run interactively by

- manually changing parameters in the TEAL GUI
- loading a custom configuration file (with suffix `.cfg`) into the task from TEAL or the command-line interface
- using command-line syntax executed in PyRAF or Python (as shown below)

The entire example may be reproduced with the three commands listed below. Since the task has been reset to default values, or "unlearned," it is only necessary to specify non-defaults parameters from the command-line.

```
--> from drizzlepac import astrodrizzle
--> unlearn astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits',output='f606w',\
driz_sep_bits='64,32',driz_cr_corr=yes,final_bits='64,32')
```

Here's a detailed description of what's done by executing the commands:

Creating the Static Mask

When the **astrodrizzle** parameter ***static*** is set to ***yes*** or ***true*** (the default setting): for each input image, the software calculates the RMS value for each chip, and identifies pixels that are below the median value by more than the "***static_sig*** value times the RMS". This step is used for identifying pixels that may have high values in the dark frame but not in the corresponding image pixels. When the science image gets subtracted by the dark during calibration, this results in anomalously large negative values in the image. Such pixels are not always flagged in the `flt.fits` DQ array, so this step allows them to be identified. Sometimes, such pixels fall on bright objects

so instead of being negative, they would be positive but lower than surrounding pixels. If the images are dithered, those abnormal pixels should land on blank sky at least some of the time, in which case they will appear negative and will be flagged.

Removing the Sky

When ***skysub*** is set to ***ye***s or ***True***, **astrodrizzle** will subtract the sky from each drizzled exposure. Sky subtraction is recommended for effective cosmic ray flagging and removal, but only if sufficient blank sky is available to perform an accurate determination. For more details, refer to Section 6.3.1.

Many astronomical fields of view cover parts of the sky devoid of any large objects. As a result, the default **astrodrizzle** sky subtraction parameters are usually sufficient, as was the case for the data in this example.

AstroDrizzle computes the sky background independently for each chip and the lowest value is taken to represent the true sky value for both chips. This value is then written to the header keyword `MDRIZSKY` in the `flt.fits` image header in both science extensions (`[sci,1]` and `[sci,2]`). Note that the sky is not subtracted from the science array pixel values in the `flt.fits` image, only in the drizzled science products.

Separately Drizzling Each Input Image onto a Common Output Frame

When ***driz_separate*** is set to ***ye***s or ***True***, each input image is drizzled to create a single-drizzled science image with suffix `single_sci.fits` and a corresponding weight image with suffix `single_wht.fits` for each input frame. These output images are used in the creation of the image median in the next step, and for the flagging of cosmic rays in subsequent steps.

An important parameter to consider in this step is ***driz_sep_bits***. Data quality flags in the `flt.fits` file, which were set during calibration, can be used as bit masks when drizzling, and the user may specify which bit values should actually be considered "good" and included in image combination. In **astrodrizzle**, this parameter may be given as the sum of those DQ flags or as a comma-separated list, as shown in this example. In this example, ***driz_sep_bits*** setting of "***64,32***" is recommended, so that warm pixels, and CTE tails in the superdark reference file, are treated as valid input pixels.

Creating the Median image

When ***median*** is ***yes***, **astrodrizzle** creates a median image from the sky-subtracted, separately-drizzled input images, allowing a variety of combination and rejection schemes. In this example, the default image combination method, ***combine_type***, is set to ***minmed***. For this computation, each pixel in the output combined image will either be the median or the minimum of the input pixel values, depending on whether the median is above the minimum by more than *n* times ***sigma***. An estimate of the "true" counts is obtained from the median image while the minimum is actually the minimum unmasked ("good") pixel. This algorithm is designed to perform optimally in the case of combining only a few images (three or four), where triple-incidence cosmic rays often pose a serious problem for more simplified median-combination strategies. The ***minmed*** algorithm performs the following steps:

1. Create a median image, rejecting the highest pixel and applying masks.
2. Use this median to estimate the true counts, and thus derive an RMS.

3. If the median is above the lowest pixel value by less than the first value in ***combine_nsigma***, then use the median value, otherwise use the lowest value.
4. If ***combine_grow*** > 0, repeat the three steps above for all pixels around those that have already been chosen as the minimum, this time using a lower significance threshold as specified as the second value in ***combine_nsigma***.

The last step is very successful at flagging the lower signal-to-noise "halos" around bright cosmic rays which were flagged in the first pass.

When the parameter ***median_newmasks=yes*** or ***True*** (default value), the single-drizzled weight maps (with suffix `single_wht.fits`) are used to create pixel masks for each image based on the pixel flag information present in the `flt.fits` DQ arrays. These masks are used when creating the median image to prevent bad pixels from adversely affecting the median.

Selecting the best parameters for the median step can be an iterative process and should always involve examination of the clean, combined median image to verify its quality:

- The majority of cosmic rays and other artifacts should have been successfully removed
- The width and shape of PSFs should be inspected over the field of view to confirm the quality of the image registration. A PSF which is "round" and "narrow" is a sign that the alignment was successful
- To confirm that no residual offsets or rotations remain, the positions of stars and the geometric transformations may be examined by running **tweakreg** on the separately-drizzled images.

The median combination may require some fine-tuning of the rejection parameters to produce a clean result. To avoid overwriting the median image when fine-tuning the parameter values, the user may wish to copy this product to a unique filename before rerunning AstroDrizzle.

Computing Cosmic Ray Masks

For optimal cosmic ray flagging, sky subtraction should be performed when the background is more than a few electrons.

When ***blot=yes*** or ***True*** (default value), the median image is transformed back ("reverse drizzled") to the reference frame of each original input image. This is done by backing out the shifts and rotations which were applied in the "single drizzle" step, and by applying inverse distortion corrections. The median image is resampled to the pixel scale of the original images and is trimmed to match the dimensions of each input image. The blotted frames, created for each chip, are named `_sci?_blt.fits` ("?" refers to the extension number). If input images are blinked with their blotted counterparts, the blotted images should align perfectly with their respective input images and should be similar in appearance, except for being cleaned of cosmic rays and other defects.

When ***driz_cr=yes*** or ***True*** (default value), **astrodrizzle** uses the original input images, the blotted median image, and the *derivative* of the blotted image (created using the *deriv* algorithm) to create a cosmic ray mask for each input image.

The deriv algorithm uses the blotted median image to compute the absolute value of the difference between each pixel and its four surrounding neighbors; for each

pixel, the largest of these four values is then used by the *driz_cr* algorithm to flag cosmic rays and other blemishes, such as satellite trails. Where the difference is larger than can be explained by noise statistics, the flattening effect of taking the median, or an error in the shift (the latter two effects are estimated using the image derivative), the suspect pixel is masked.

Cosmic rays are flagged using the following rule:

$$|data\_image - blotted\_image| > scale \times deriv + SNR \times noise$$

Here, *scale* is the multiplicative factor applied to the derivative, *deriv*. This expression is used to determine if the difference between the data image and the blotted image is large enough to require masking. *noise* is calculated using a combination of the detector read noise and the poisson noise of the blotted median image, plus the sky background.

The user must specify two cut-off signal-to-noise (*SNR*) values for determining whether a pixel should be masked: the first for detecting the primary cosmic ray, and the second for masking lower-level bad pixels adjacent to those found in the first pass. Since cosmic rays often extend across several pixels, the adjacent pixels make use of a slightly lower *SNR* threshold. If desired, a third-pass cosmic ray rejection can be carried out by "growing" the cosmic rays via the ***driz_cr_grow*** parameter.

Images obtained from the Archive have already been run through AstroDrizzle using a set of default parameters. For most[2] detectors, pixels identified as cosmic rays during pipeline processing are flagged with the value of 4096 in the data quality (DQ) array of the `flt.fits` images. Because pipeline parameter settings may not provide optimal flagging for every dataset, users are encouraged to fine-tune the cosmic ray rejection parameters ***driz_cr_snr*** and ***driz_cr_scale***.

When **astrodrizzle** is run, cosmic rays flagged during pipeline processing or previous **astrodrizzle** runs are reset in the `flt.fits` DQ arrays via the parameter ***resetbits=4096*** (default value) which changes the bit value 4096 to 0. That way, cosmic ray detection can be performed from scratch by **astrodrizzle**.

When ***driz_cr_corr*** is set to ***yes*** or ***True***, the task will create both a cosmic ray mask image (suffix `sci?_crmask.fits` where "?" is the extension number) and a clean version of the original input images (suffix `crclean.fits`), where flagged pixels are replaced by pixels from the blotted median.

It is strongly recommended that the quality of the cosmic ray masks be verified by blinking the original `flt.fits` input image with both the cosmic ray-cleaned image (`cr_clean.fits`) and the cosmic-ray mask (`crmask.fits`) as shown below.

```
disp ibc301qrq_flt.fits[sci,1] 1 zs- zr- z1=0 z2=250
disp ibc301qrq_crclean.fits[sci,1] 2 zs- zr- z1=0 z2=250
disp ibc301qrq_sci1_crmask.fits 3 zs- zr- z1=0 z2=1
```

Figure 7.4 shows the original input image `ibc301qrq_flt.fits[sci,1]` with its corresponding cosmic ray mask in Figure 7.5.

2. For WFC3/IR images, the flag value for cosmic rays identified by AstroDrizzle are 8192, to distinguish them from cosmic rays detected in **calwfc3** with flag 4096.

Figure 7.4: Chip 2 of Input Image `ibc301qrq_flt.fits`

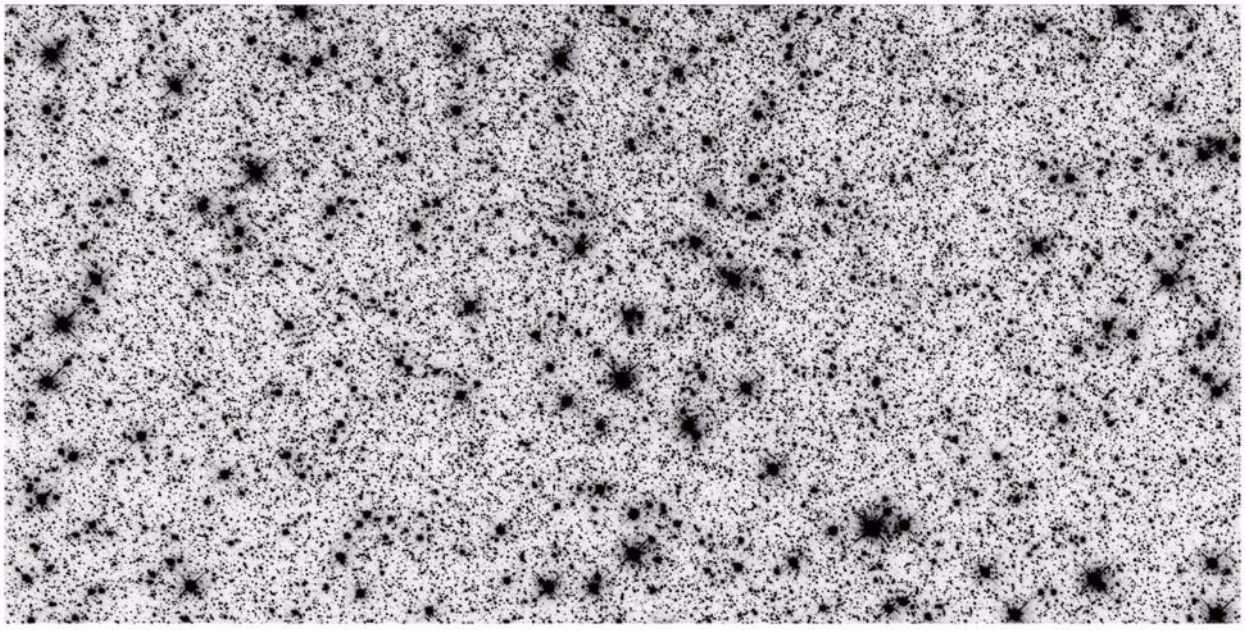
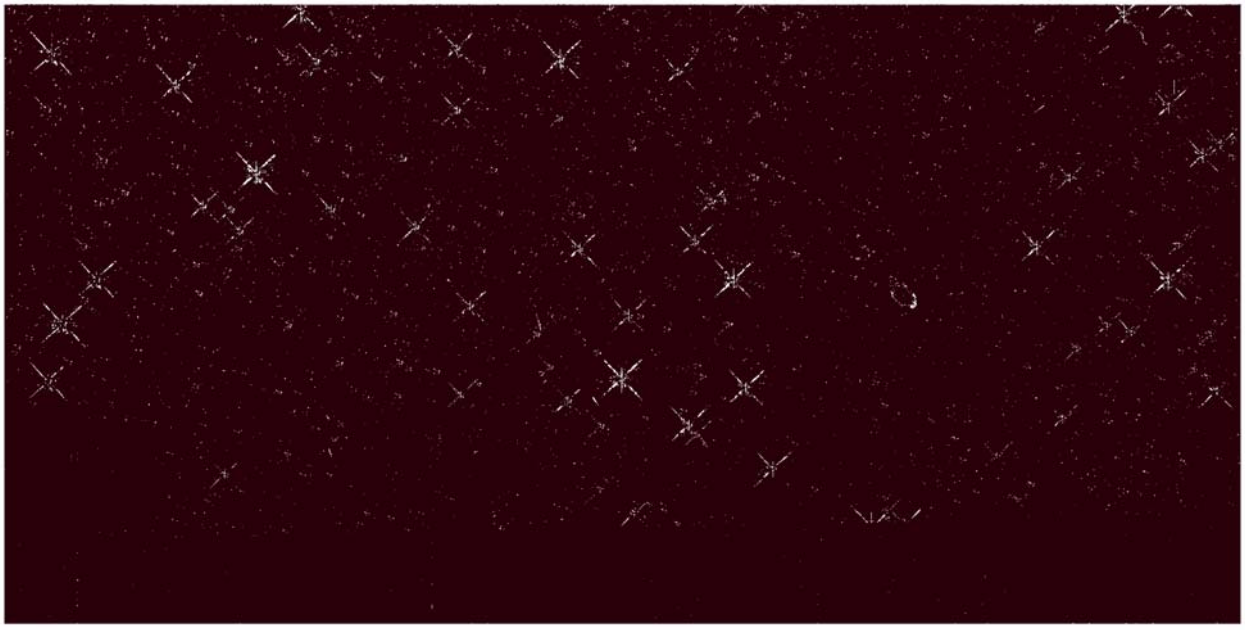


Figure 7.5: The Cosmic Ray Mask for `ibc301qrq_flt.fits[1]` (Chip 2)



A mask value of 1 (shown in black) corresponds to good pixels, while a mask value of 0 (white) corresponds to pixels flagged as cosmic rays. The lack of flagging in the lower portion of the figure is because there were no overlapping images in that area.

Final Drizzle Combination

When ***driz_combine*=*yes*** or ***True*** (default), **astrodrizzle** takes the original input images, together with the final masks, and drizzles them onto a single output image. The final masks are a combination of the static masks, the cosmic ray masks, and the DQ values assigned as "bad" by the user via the ***final_bits*** parameter.

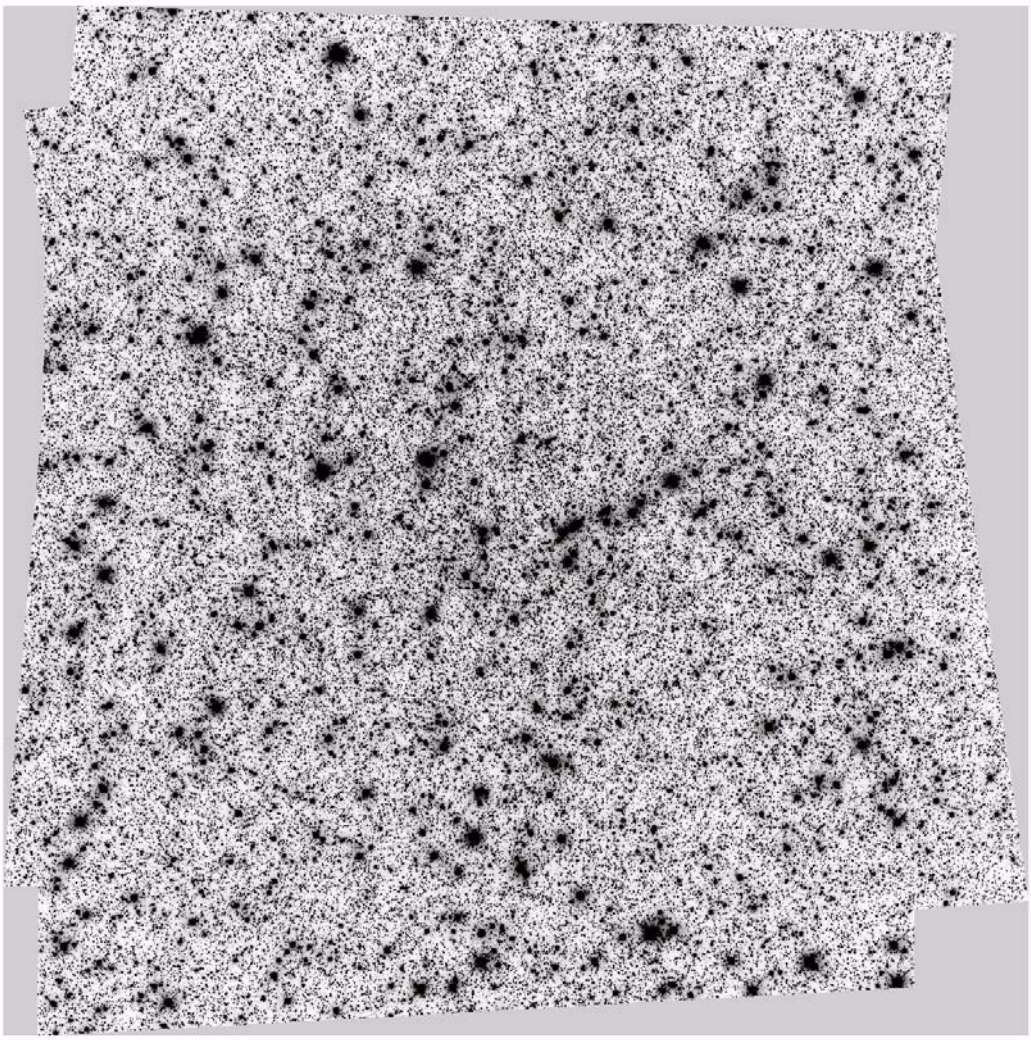
By default, UVIS images are drizzled to a scale of 0.0396 arcseconds/pixel, but the user is encouraged to experiment with other options (e.g., shrinking the ***final_scale*** and ***final_pixfrac*** value to yield a sharper output PSF.) For more information, refer to Section 6.3.3 and the example at Section 7.2.

In this example, the final image is weighted by the exposure time of each image, so ***final_wht_type=EXP***. This weighting is a good approximation in the regime where the noise is dominated by photon counts from the sources, while the contributions from sky background, read noise, and dark current are negligible. For a discussion about photometry and weights in AstroDrizzle, please refer to Section 4.2.12.

When the parameter ***build=yes*** or ***True*** (a non-default value), AstroDrizzle generates a single multi-extension FITS file named `<output>_drz.fits`, where the ***output*** parameter defines the string to name the final product. For this example, the image is named `f606w_drz.fits` (so ***output*='*f606w*'**).

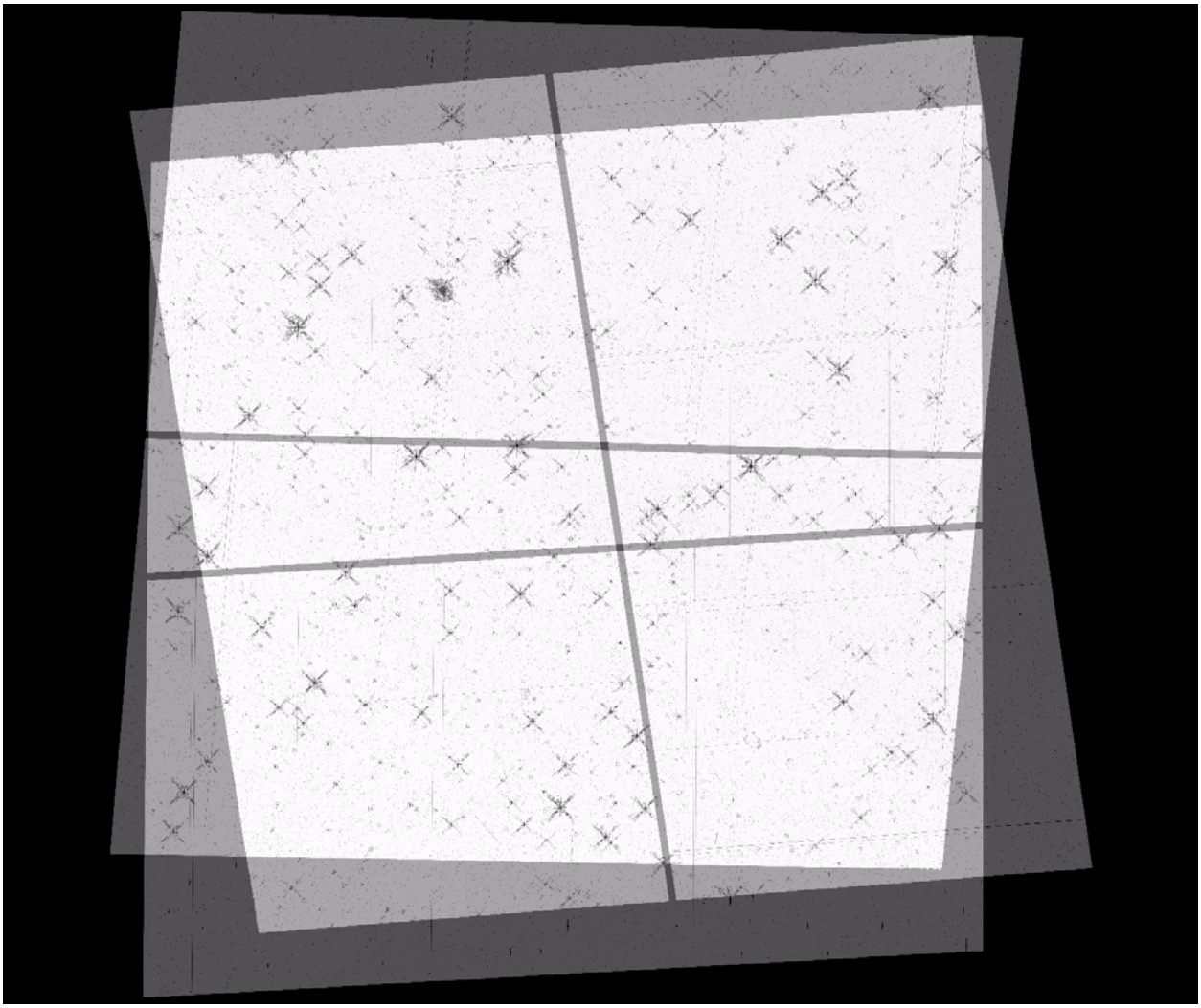
The first extension of the drizzled product, shown in Figure 7.6, represents the WCS-aligned combination of science extensions of the three input images. Each pixel now covers an equal area on the sky and has an equal photometric normalization across the field of view. Thus, the drizzled science image is photometrically and astrometrically accurate for point sources and extended sources. The dimensions of the output image are computed on-the-fly by AstroDrizzle. The user also has the option to define a custom size and orientation for the final output image, by setting parameter values in the ***final_wcs*** section of the **astrodrizzle** parameters.

Figure 7.6: Extension #1 of `f606w_drz.fits`, the Final Combined Science Image



The second extension of the drizzled image contains the weight image (Figure 7.7). When ***final_wht_type*** is set to ***EXP***, the weight image can be considered an effective exposure time map. The weight image, in Figure 7.7, shows lighter areas that have higher weight. The chip gaps are clearly visible, as are column defects and cosmic ray features. The majority of the image is white, with a value of ~120 which is the combined exposure time of the three 40-second exposures. Regions that have less than a full contribution from all three observations are colored various shades of gray. Examples of this can been seen around the edges of the weight image, where there was little image overlap, in the three intersecting chip gaps, and in the saturated cores and diffraction spikes surrounding bright stars.

Figure 7.7: Extension #2 of `f606w_drz.fits`, the Final Weight Image



The third extension of the drizzled image (shown in Figure 7.8) contains the *context* information. Here, each input image chip is identified by a bit (power of 2) in a 32-bit integer. For instance, image 1/chip 1 = 2^0 (1), image 1/chip 2 = 2^1 (2), image 2/chip 1 = 2^2 (4), image 2/chip 2 = 2^3 (8), image 3/chip 1 = 2^4 (16), image 3/chip 2 =2^5 (32), and so on[3]. Each context pixel value is an additive combination of these chip bits, depending on which images contributed to the corresponding pixel in the drizzle-combined image. For example, if the pixel (200,100) in the drizzle-combined image has contributions from image 1 / chip 1, image 2/chip1, and image 3/chip 2, the value for pixel (200,100) in the context image would be the sum of the image bit values, 1+4+32=37.

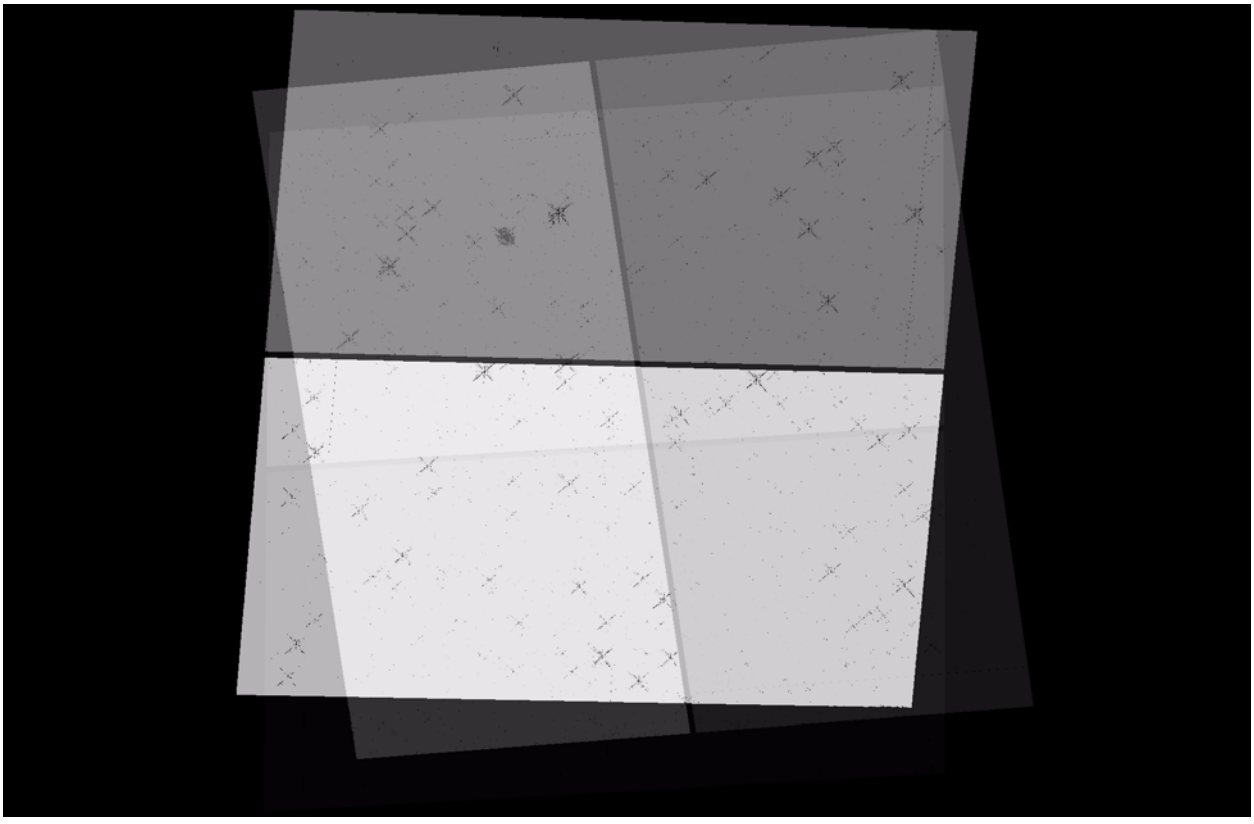
3. Should more than 32 chips be combined into a final output image, this context array will be "expanded" by creating another 32-bit integer array of the same size, layered on top of the original to create a 3D array with 2 layers. The new layer would then be used to keep track of the next 32 chips, with additional layers being created for more sets of 32 chips, as needed.

For optimal cosmic ray rejection, the drizzled science product in this example has been sky-subtracted. Some photometry packages, however, require that the sky be added back in to the final product prior to analysis. For example, the DAOPHOT Reference Guide recommends that

> "the sky background should NOT be subtracted from imaging prior to photometry. DAOPHOT fitting routines use an optimal weighting scheme which depends on the read noise, gain, and true counts in each pixel. If the mean sky has been subtracted, the computed weights will be incorrect."

Therefore, the user can add the average sky value that was determined from the individual input frames. This average sky can be determined by querying the `MDRIZSKY` keyword in the *merged header table extension* `HDRTAB`, described in Section 4.4.6. Additionally, **daophot** assumes that the final image is in units of electrons, so if the image was drizzled with the parameter ***final_units*** set to ***cps***, it may be necessary to convert the image back to units of electrons by multiplying the image by the exposure time prior to performing photometry.

Figure 7.8: Extension #3 of f606w_drz.fits, the Final Context Image



7.2 WFC3/IR: Optimizing Image Sampling for a Single Visit

Introduction

This example was written to help users better understand the subtleties in improving image sampling for dithered data. Four images of a spiral galaxy were acquired using WFC3/IR, following the `WFC3-IR-DITHERBOX-MIN` dither pattern that was designed to provide optimal sampling of the PSF.

Summary of Steps

1. Description of the Data
2. Run **astrodrizzle** several times using different settings for the ***final_pixfrac*** and ***final_scale*** parameters
3. Compare and evaluate results of using different ***final_pixfrac*** and ***final_scale*** values

7.2.1 Description of the Data

Four WFC3/IR images[4] of the spiral galaxy NGC 3370 (Program 11570), taken in the F160W filter, were acquired in a single visit and at the same telescope orientation. Observations were obtained using the default WFC3/IR dither pattern, `WFC3-IR-DITHERBOX-MIN`, with relative pixel coordinates (0, 0), (4.2, 1.4), (2.6, 3.8), (-1.6, 2.4), which is designed to provide optimal PSF sampling.

Calibrated data products from the Archive are:

- An association table, with suffix `asn.fits`
- Flat-field calibrated images, with suffix `flt.fits`
- Drizzled image product, with suffix `drz.fits` that was created by running AstroDrizzle in the pipeline with a default set of parameters.

The pipeline `drz.fits` image may be saved to a separate directory for later comparison with the drizzled products from this example. In general, drizzled data from the Archive should be regarded as “quick look” data products, used to make an initial evaluation of the observations.

4. Data for this example can be retrieved from the HST Archive by searching for Dataset `ib1f19010`.

Table 7.3: Summary of Images in this Example

Image Name	Association ID	Proposal ID	Visit & Line Number	POS TARG)x,y in arcseconds)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
ib1f19l6q_flt.fits	IB1F19010	11570	19.001	0.0000,0.0000	320.9999	2010-04-04	502.9365
ib1f19l7q_flt.fits	IB1F19010	11570	19.001	0.5423, 0.1818	320.9999	2010-04-04	502.9365
ib1f19l9q_flt.fits	IB1F19010	11570	19.001	0.3389,0.4848	320.9999	2010-04-04	502.9365
ib1f19laq_flt.fits	IB1F19010	11570	19.001	-0.2034,0.3030	320.9999	2010-04-04	502.9365

Since the data were obtained in a single visit as part of a subpixel dither box pattern, the WCS of the individual frames are usually aligned to 0.1 pixels. This example does not describe the use of TweakReg to verify (and/or improve) image alignments, but users are strongly encouraged to do so because even the smallest misalignment can compromise the photometric integrity of the final drizzled products.

For the IR detector[5], calibrated data products (`flt.fits`) consist of five extensions:

- science image (SCI)
- error array (ERR)
- data quality array (DQ)
- number of samples array (SAMP)
- integration time array (TIME)

A WFC3/IR FITS file will therefore contain the primary header unit and five extensions, which together form a single IR exposure. To see the contents of the IR file structure, the user can use the IRAF task **catfits**, shown below in a PyRAF session.

```
--> catfits ib1f19l6q_flt.fits
EXT#  FITSNAME       FILENAME  EXTVER DIMENS     BITPIX
0     ib1f19l6q_flt  ib1f19l6q_flt.fits          16
1     IMAGE          SCI       1      1014x1014  -32
2     IMAGE          ERR       1      1014x1014  -32
3     IMAGE          DQ        1      1014x1014  16
4     IMAGE          SAMP      1      1014x1014  16
5     IMAGE          TIME      1      1014x1014  -32
```

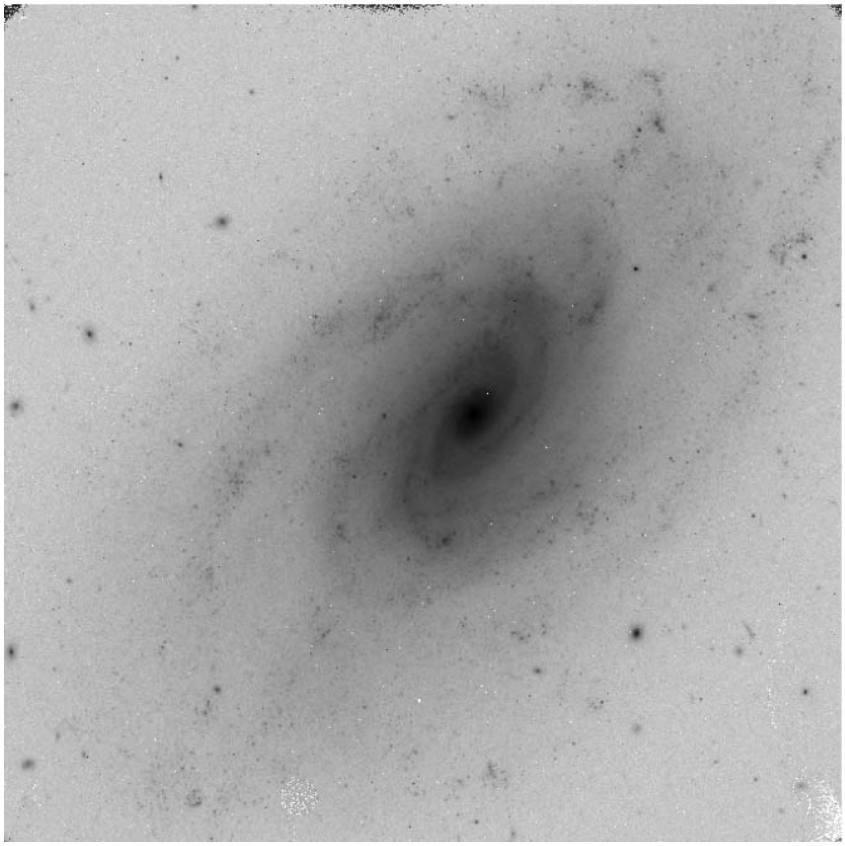
5. For details on the IR channel file structure, see Section 2.2.2 of the *WFC3 Data Handbook* at http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/wfc3_Ch23.html#96833

The SCI, DQ, and TIME extensions are shown in Figures 7.9, 7.10, and 7.11 for the first flt.fits image in the association. They were displayed using these IRAF commands:

```
--> display ib1f19l6q_flt.fits[sci,1]  1 zs- zr- z1=0.50 z2=100 ztr=log fill+
--> display ib1f19l6q_flt.fits[dq,1]   2 zs+ zr+ fill+
--> display ib1f19l6q_flt.fits[time,1] 3 zs+ zr+ fill+
```

The TIME extension is useful for identifying pixels which were saturated in one or more samples (such as the core of the galaxy in this example, which saturated after 8 of 12 total samples), or cosmic rays which were flagged in "up-the-ramp" fitting, usually in a single sample. (For more information, see Section 3.4.3 of the *WFC3 Data Handbook*.) Note that cosmic rays flags[6] are actually flagged in the IMA[7] files with a bit value of 8192, but their effect can be seen in the "reduced" exposure time in the 5th extension of the flt.fits files.

Figure 7.9: Science Portion of the Calibrated WFC3 Image



6. The specific DQ flag values are unique for each detector and are defined in the Instrument Data Handbooks. For a table of WFC3/IR DQ flags, see Table 2.5 in the *WFC3 Data Handbook* at http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/wfc3_Ch23.html#98193

7. For more information about WFC3 data products, see Section 2.1.1 in the *WFC3 Data Handbook* at http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/wfc3_Ch22.html#96161

Figure 7.10: Data Quality Portion of the Calibrated WFC3 Image

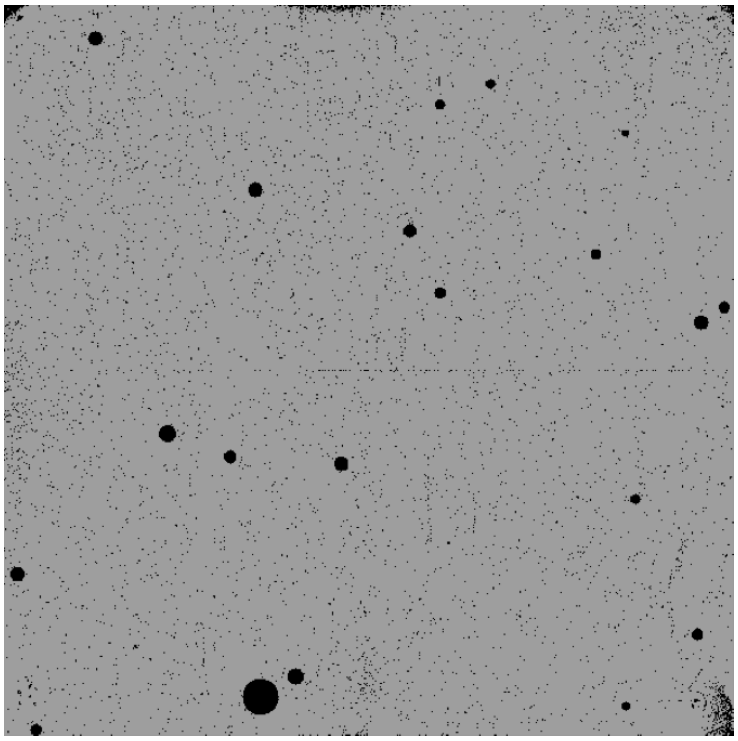
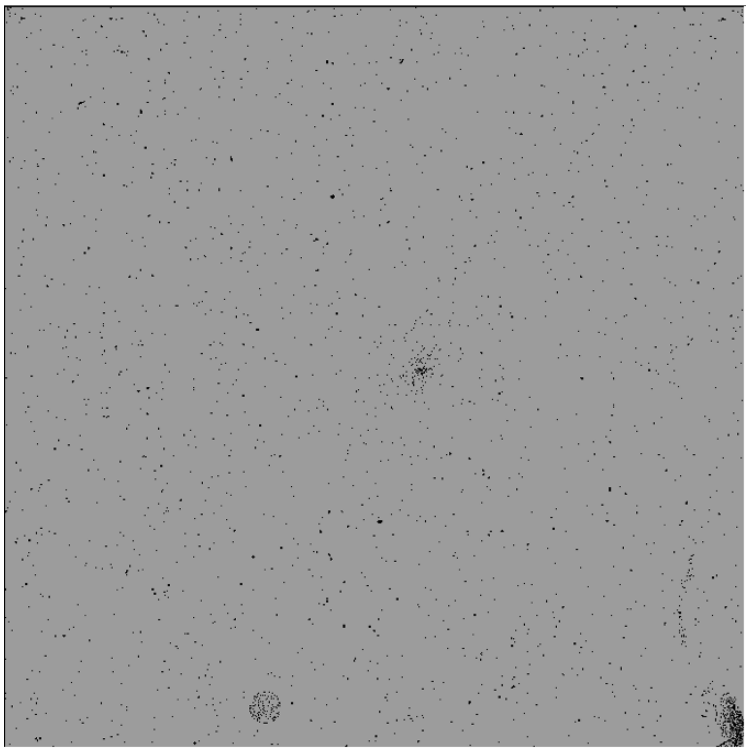


Figure 7.11: Exposure Time (TIME) Extension of the Calibrated WFC3 Image



Black pixels indicate a lower total exposure time. The galaxy core, just off the center of the frame, was saturated after 8 of the 12 total samples.

7.2.2 Recommendations on Selecting Optimal "Scale" and "Pixfrac" Parameter Values

To optimize the parameters for drizzle combination, users are encouraged to experiment with various combinations of the parameters ***final_scale*** (size, in arcseconds, of the output pixels) and ***final_pixfrac*** (the fractional linear size of the input pixel "drop" into the output image frame).

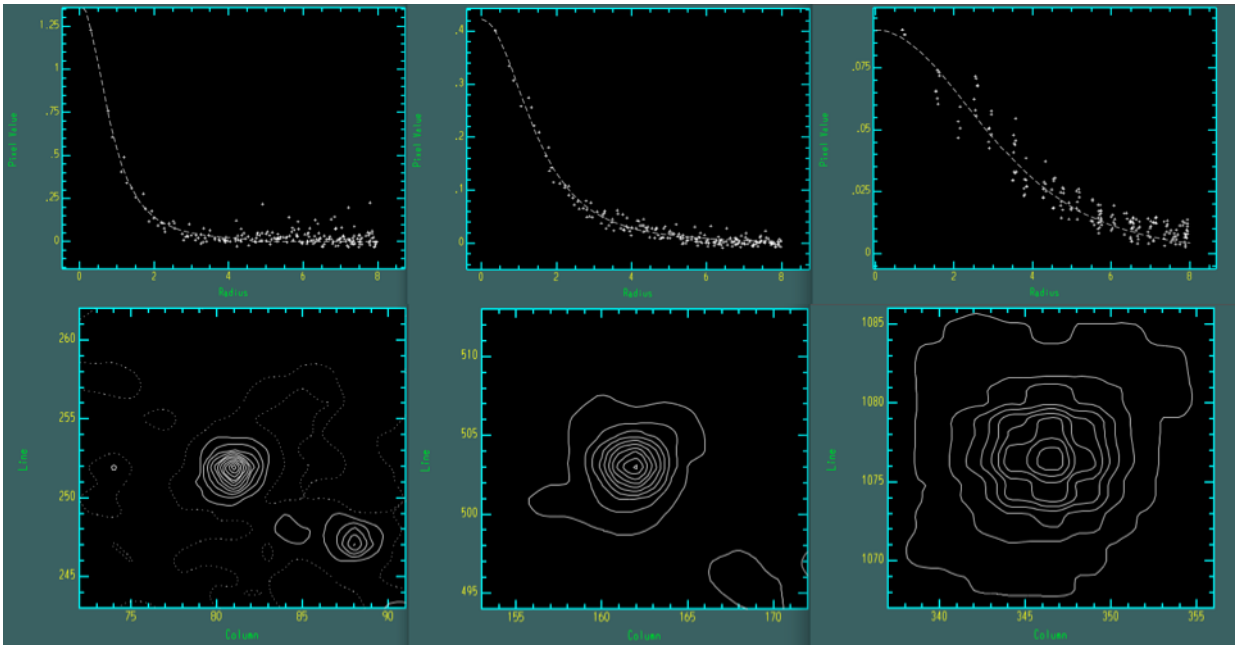
The recommended method is to first select the ***final_scale*** value, then the ***final_pixfrac*** value. While experimenting with ***final_scale***, the ***final_pixfrac*** should be fixed at 1.0. Ideally, the scale is chosen to sample the PSF FWHM by about ~2.0 to 2.5 pixels, if allowed by the data. Non-integral (subpixel) dithers allow the recovery of some information lost to undersampling by pixels that are large compared to the point spread function. The FWHM of the IR point spread function is approximately 1.0 pixel, so subpixel dithering allows the user the ability to recover spatial resolution. (Ideally one would like a minimum two samples per FWHM for the full recovery of the image resolution.)

While reducing the ***final_scale*** from the default value, the PSF will begin to degrade and resemble the dither pattern (a "cross-shaped" PSF, for example, for a four-point dither). This is illustrated in Figure 7.12, where ***final_scale*** has been decreased from 0.1283 arcseconds/pixel to 0.0642 arcseconds/pixel, then to 0.032 arcseconds/pixels, while maintaining ***final_pixfrac*** at 1.0. In general, the ***final_scale*** value should never be less than half the native plate scale.

The task **imexamine** was used to plot both the radial profile (top panel) and a contour plot (bottom panel) of a bright star in the drizzled image, where the FWHM is 1.6 pixels, 2.8 pixels, and 6.2 pixels, respectively. With a well-sampled 4-point subpixel dither, the best ***final_scale*** is approximately half the native scale. Often, a "convenient" number is chosen, for example, 0.065 arcseconds/pixel.

Alternately, when UVIS and IR images are obtained with four-point dithering, the former scale could be set to 0.03333 arcseconds/pixel and the later set to 0.06666 arcseconds/pixel, a factor of two difference. For WFC3/UVIS (and ACS/WFC) an output scale is 0.03333 arcseconds/pixel gives good subsampling of the PSF. It is not quite a factor of two smaller than the original pixel (which would essentially recover all of the fine scale information in the image) but tends to sample the PSF very well. For a dozen or more pointings well distributed over the image (not just a multiple repetition of a four-point dither) a finer output pixel scale could be used if high-resolution imaging is important. A scale of 0.03333 arcseconds/pixel has the virtue that three pixels is ~0.1 arcseconds, making it easy to look at the output image and know the size of an object. When WFC3/UVIS (or ACS/WFC) images are obtained with corresponding WFC3/IR observations, it may be "convenient" to select a final scale which is a factor of two larger, where three pixels is ~0.2 arcseconds.

Figure 7.12: Radial Profile and Contour Plot of a Bright Star in Images with Three Different *final_scale* Values



Radial profile and contour plot of the same star in drizzled frames obtained when *final_scale* is "shrunk" from 0.1283 arcseconds/pixel to 0.0642 arcseconds/pixel, to 0.0320 arcseconds/pixel. When the scale is set too small, the PSF shape begins to resemble the four-point dither pattern used in this observing program. The optimal scale value ultimately depends on the dataset, the number of dithers, and the amount of subpixel sampling. In this case, the middle panel is ideal.

The ***final_pixfrac*** value has to be small enough to avoid degrading the final drizzle-combined image, but large enough that when all images are "dropped" onto the final frame, coverage of the output frame is fairly uniform. In general, ***final_pixfrac*** should be slightly larger than the final output scale to allow some "spillover" to adjacent pixels. This will help avoid "holes" in the final product when a given pixel has been flagged as "bad" in several frames. As a rule of thumb, statistics performed on the drizzled weight image in the region of interest should yield an RMS value (standard deviation) that is less than 20% of the median (midpoint) value. This threshold is a balance between the benefits of improving the image resolution at the expense of increasing noise in the background.

7.2.3 Image Combination with AstroDrizzle

In default mode, AstroDrizzle performs each of its seven steps in the order outlined in Section 4.2. For IR images, however, steps three to six may be turned off since cosmic rays are flagged in **calwfc3** as part of the "up-the-ramp fitting." While it is omitted from this specific example, running these steps (using a different bit flag, like 8192, for "cosmic rays" found during **astrodrizzle** processing) may still be useful for flagging additional detector artifacts not present in the data quality arrays of the

calibrated images. Note that it is very important to subtract the sky (step two) prior to drizzing the final image, or the science array will be compromised by increased noise. The size of the effect will depend on the variation in the sky between exposures. (An example of this effect is shown in Figure 7.16)

The commands shown below run a test grid of varying ***final_scale*** and ***final_pixfrac*** values to show how the images change at different settings.

When the parameter ***build=yes*** (a non-default value), the final AstroDrizzle output image for this example will be a single multi-extension FITS file named `f160w_drz.fits`, containing the science image in extension one, the weight image in extension two, and the context image in extension three. When ***build=no***, the science, weight, and context images are written to separate output files. Since the output file `f160w_drz.fits` will be overwritten with each successive run, this example renames the drizzled product with a unique name between each separate trial. The commands below use the command-line syntax; non-default parameter values are highlighted in bold.

```
--> import drizzlepac
--> from drizzlepac import astrodrizzle
--> unlearn astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.1283,\
final_pixfrac=1.0)
--> imrename f160w_drz.fits f160w_drz_test1.fits
```

Next, run the previous **astrodrizzle** commands, varying only the ***final_scale*** parameter.

```
--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0898,final_pixfrac=1.0)
--> imrename f160w_drz.fits f160w_drz_test2.fits

--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0642,final_pixfrac=1.0)
--> imrename f160w_drz.fits f160w_drz_test3.fits

--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0513,final_pixfrac=1.0)
--> imrename f160w_drz.fits f160w_drz_test4.fits
```

Once the scale is chosen, the value of ***final_pixfrac*** may then be varied. Note that while ***final_scale*** is represented in arcseconds, ***final_pixfrac*** is represented as a fraction of the native pixel size.

```
--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0642,final_pixfrac=0.9)
--> imrename f160w_drz.fits f160w_drz_test5.fits

--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0642,final_pixfrac=0.8)
--> imrename f160w_drz.fits f160w_drz_test6.fits

--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0642,final_pixfrac=0.7)
--> imrename f160w_drz.fits f160w_drz_test7.fits

--> astrodrizzle.AstroDrizzle('*flt.fits',output='f160w',build=yes,\
static=no,skysub=yes,driz_separate=no,median=no,blot=no,driz_cr=no,\
driz_combine=yes,final_wcs=yes,final_bits=576,final_scale=0.0642,final_pixfrac=0.6)
--> imrename f160w_drz.fits f160w_drz_test8.fits
```

The first extension of the drizzled product, `160w_drz.fits[1]` contains the science (`SCI`) image, a combination of the four dithered images which has been corrected for distortion. All pixels cover an equal area on the sky and have an equal photometric normalization across the field of view, giving an image that is photometrically and astrometrically accurate for both point and extended sources. The `SCI` portion of the drizzled product, shown in Figure 7.13, is in units of electrons/seconds. (Changing the ***final_units*** parameter from the default value ***cps*** (counts per second) to ***counts*** will produce a drizzled image in units of electrons.)

The second extension of the output image contains the weight (`WHT`) image. When ***final_wht_type*** is set to ***EXP***, the weight image can be considered an effective exposure time map of the science (`SCI`) image. In Figure 7.14, darker areas in the `WHT` extension image have lower weights. IR weight images represent several different types of information; when ***final_pixfrac=1.0***, the weight image will resemble the `TIME` extension of the `flt.fits` image, minus the pixels which were flagged in the flt.fits DQ array and not specifically set as "good" in the final_bits parameter.

Note that in this example, the ***final_bits*** value is ***576*** (it can also be written as ***512,64***) to tell **astrodrizzle** that `flt.fits DQ` flags of 512 (bad pixels in the flat field) and 64 (warm pixels) should be treated as "good" pixels. All other DQ flags in the `flt.fits` images, treated as "bad," are reflected in the `single_wht.fits`

weight images. The smooth top-to-bottom gradient in the `WHT` image reflects the geometric distortion in the IR detector where detector pixels represent different areas on the sky. When ***final_pixfrac*** is shrunk to values smaller than ***1.0***, the RMS of the `WHT` image increases, as shown in the bottom panel of Figure 7.14. When ***final_pixfrac*** is too small relative to `final_scale`, there will be pixels with "holes" in the weight image where less than one pixel contributed to the value of the final flux in the drizzled science image. The majority of the variations in the bottom `WHT` image is due to the change in geometric distortion over the chip, where the input pixels cover significantly different areas on the sky.

Figure 7.13: Science Extension of the Final WFC3 Drizzled Product

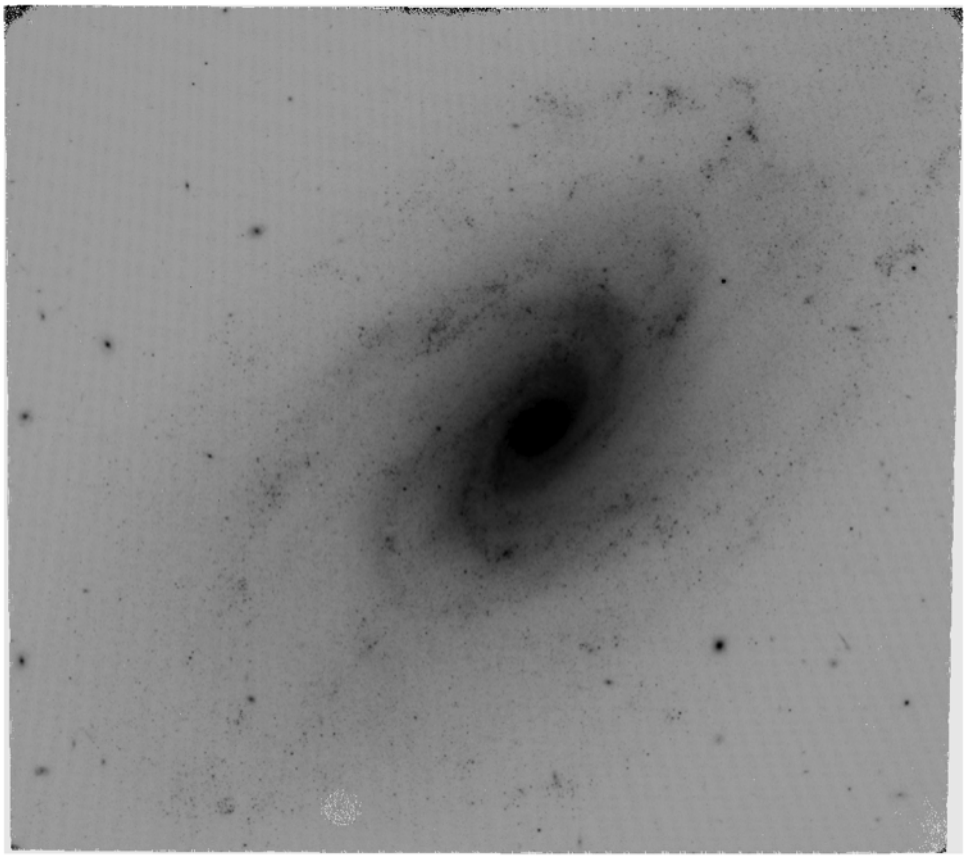
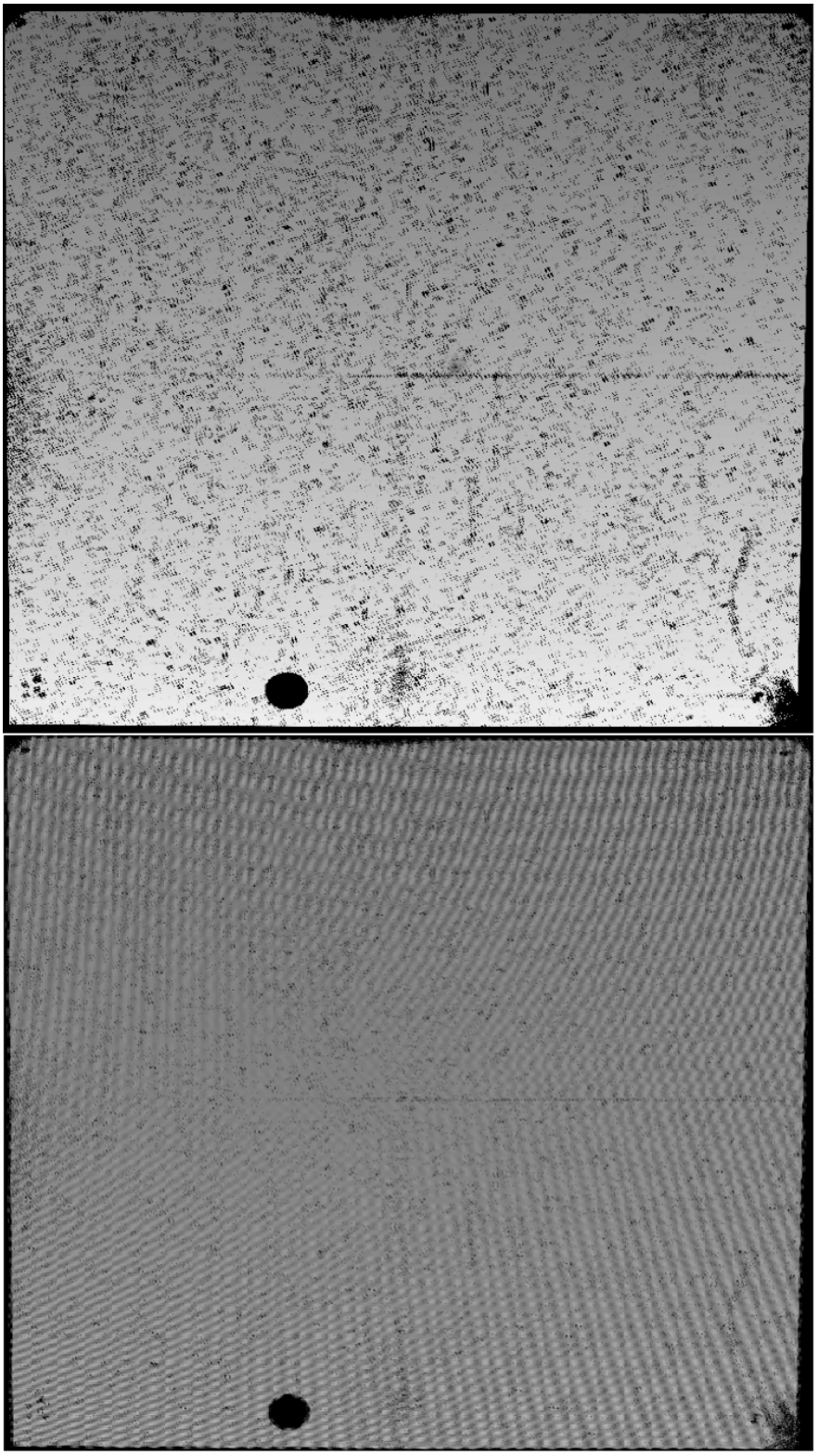


Figure 7.14: Weight Extension of the Final WFC3 Drizzled Product (Top: *final_pixfrac=1.0***; Bottom:** *final_pixfrac=0.8*



Statistics in the weight image (RMS/median) are reported in Table 7.4 for several trials, computed using the IRAF task **imexam** for a 200x200 pixel box in the center, and in the top left corner of each weight image. The PSF FWHM was measured using an isolated star at coordinate (430, 1746) in the trial image, where the ***final_scale*** setting used to create it is ***0.0642*** (in arcseconds/pixel). Note that the table gives the value of ***final_scale*** (shown in the table as "Scale") in two different ways: as a fraction of the default plate scale and in arcseconds/pixel (the units used in **astrodrizzle**).

Table 7.4: Weight Image Statistics and PSF FWHM for Various Final Drizzle Scale/Pixfrac Combinations

Trial Number	Pixfrac (fraction)	Scale (fraction)	Scale (arcsec.)	RMS/Median (center)	RMS/Median (corner)	PSF FWHM (pixels)	PSF FWHM (arcsec.)
3	1.0	0.5	0.0642	0.061	0.066	2.97	0.191
5	0.9	0.5	0.0642	0.068	0.073	2.90	0.186
6	0.8	0.5	0.0642	0.076	0.076	2.85	0.183
7	0.7	0.5	0.0642	0.083	0.095	2.78	0.179
8	0.6	0.5	0.0642	0.090	0.109	2.70	0.173

Statistics of the weight image for both regions of the detector meet the general guideline of rms/median < 0.2 for all the trials. However, if one visually compares the science products in Figure 7.15, it becomes apparent that maintaining a larger ***final_pixfrac*** ensures overlap between pixels and less correlated noise in the science array (bottom panel). When ***final_pixfrac*** has been shrunk too much (top panel), a "beating pattern" can be seen in the sky. While this pattern may look alarming to the eye, it has only a very minor effect on the photometric integrity of the drizzled products.

Determining which is the best solution is a matter of judgment, depending on the preferred resolution and quality of image. If the target is primarily in the center of the frame, the ***final_scale*** and ***final_pixfrac*** selection may be more aggressive. If sources cover the entire field of view, however, a more conservative set of parameters may be preferable.

While trial number eight gives a narrower PSF FWHM, it does not do a good job at removing detector artifacts. Trial number 6 is shown in Figures 7.13 and 7.14, where the ***final_scale*** is equal to 0.5 times the default pixel scale and ***final_pixfrac*** is ***0.8***. The resulting image has a plate scale of 0.0642 arcseconds/pixel with the PSF FWHM at 0.183 arcseconds. Because the WFC3/IR detector pixels are significantly undersampled, optimizing the ***final_scale*** and ***final_pixfrac*** parameters will produce a dramatic improvement in resolution, as seen in Figure 7.17.

Figure 7.15: Comparison of *final_pixfrac=0.6* (top) and *0.8* (bottom)

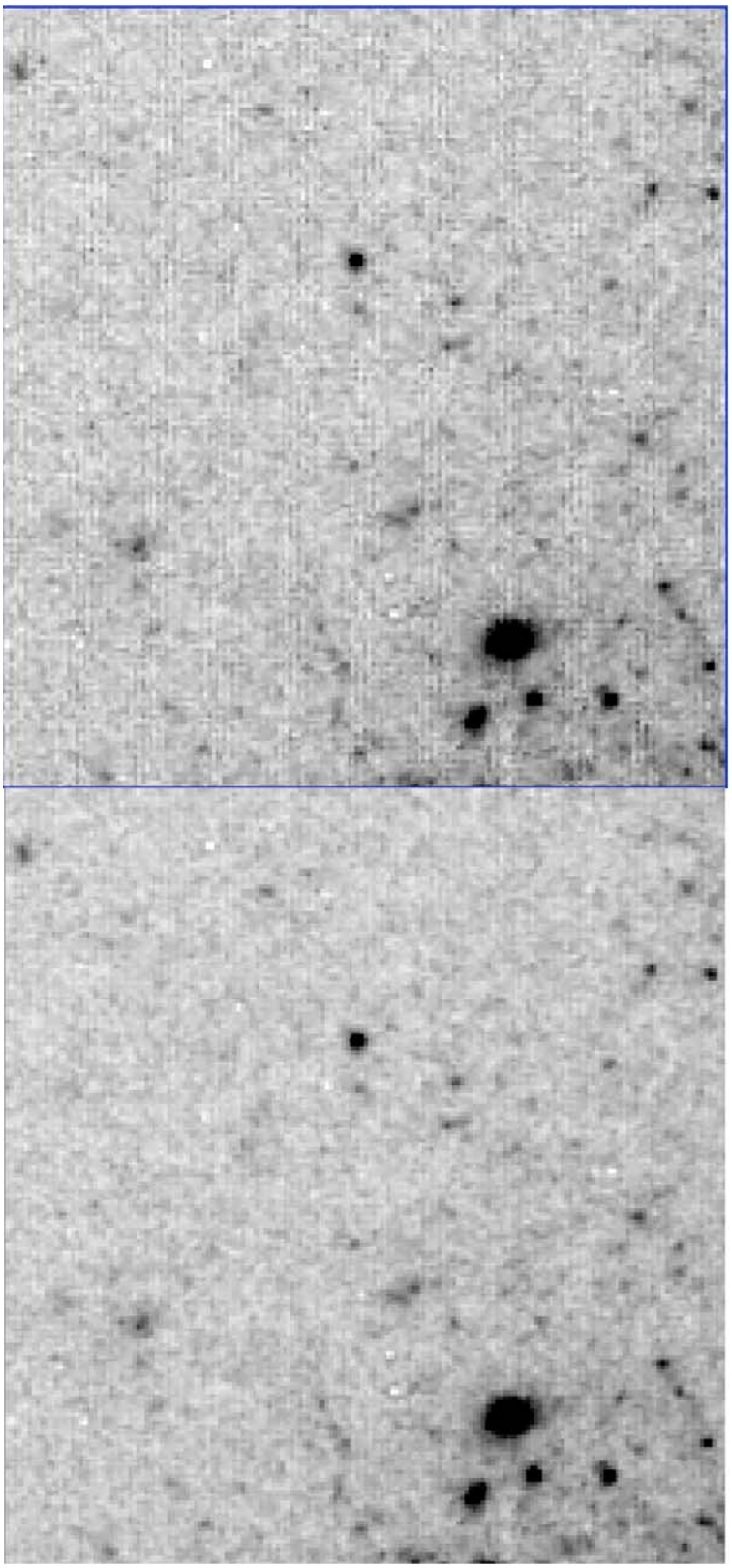
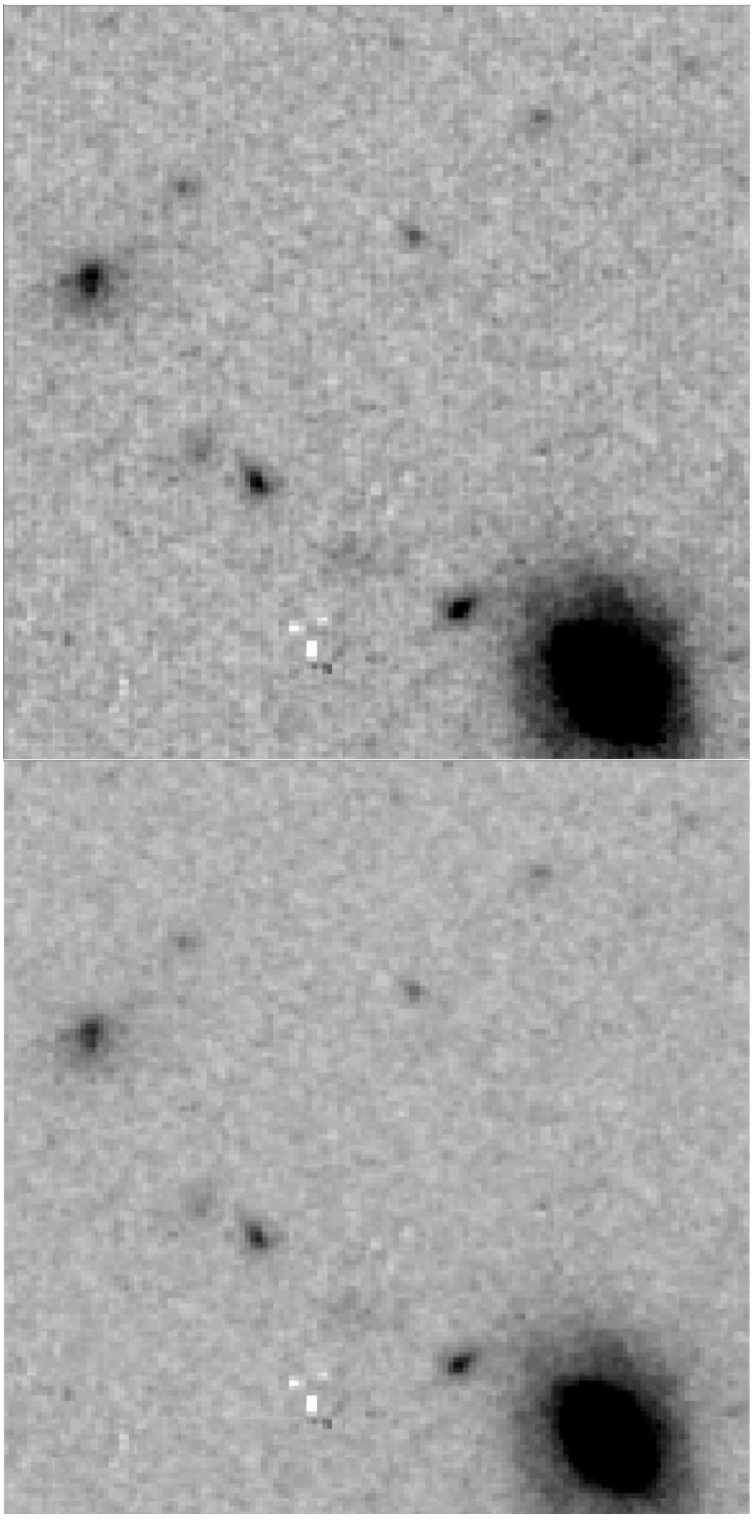
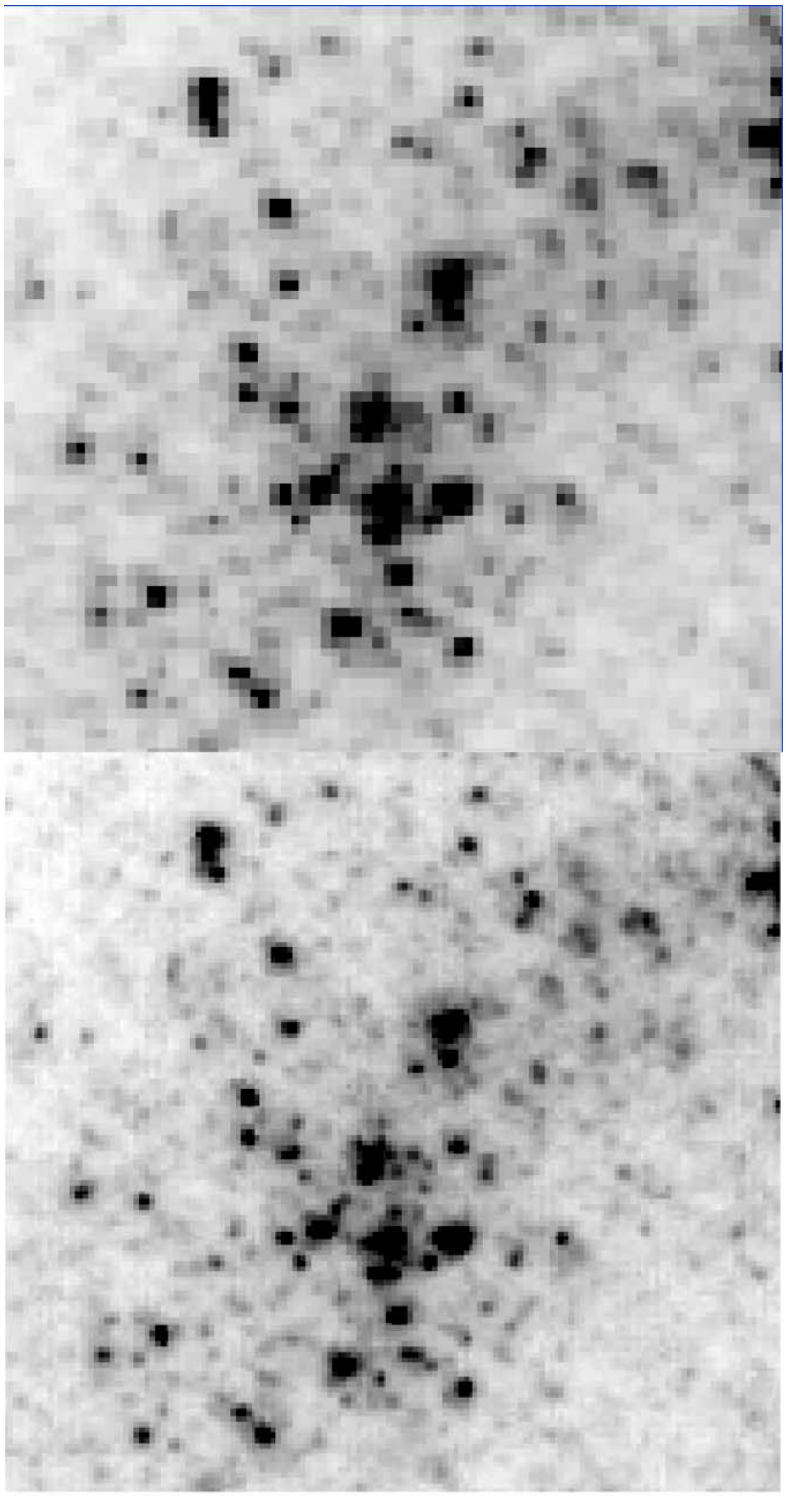


Figure 7.16: Sky Background in the Final Science Array With No Sky Subtraction (Top) and With Sky Subtraction (Bottom)



Note the additional noise in the top panel. The effect in this example is subtle, but will be more pronounced in images with larger sky variability between exposures.

Figure 7.17: Improvement in Resolution of the Pipeline Product (Top) Versus the Optimized Drizzled Product (Bottom) l



7.3 ACS/WFC: Optimizing the Image Alignment for Multiple Visits

Introduction

Three ACS/WFC images of a crowded stellar field, obtained at different epochs, with different `PA_V3` orientations, will be aligned to a common WCS using **tweakreg**, then combined using **astrodrizzle**.

Example Overview

1. Description of the data.
2. Align the images using TweakReg. Several runs of **tweakreg** are demonstrated, showing what bad results look like, and how to use them as clues to converging on good image offset determinations.
3. Combine the image using **astrodrizzle**.

7.3.1 Description of the Data

Images of the globular cluster NGC 104 (Program 10737) were obtained using ACS/WFC in the F606W filter. Three images, each taken in different visits, at different pointings and orientations, will be used to demonstrate the use of **tweakreg** and **astrodrizzle** to align and combined images. The input images for this example are flat-field calibrated images processed in the pipeline by **calacs** and corrected for CTE (indicated by the suffix `flc.fits`). A summary of the observations are provided in Table 7.5.

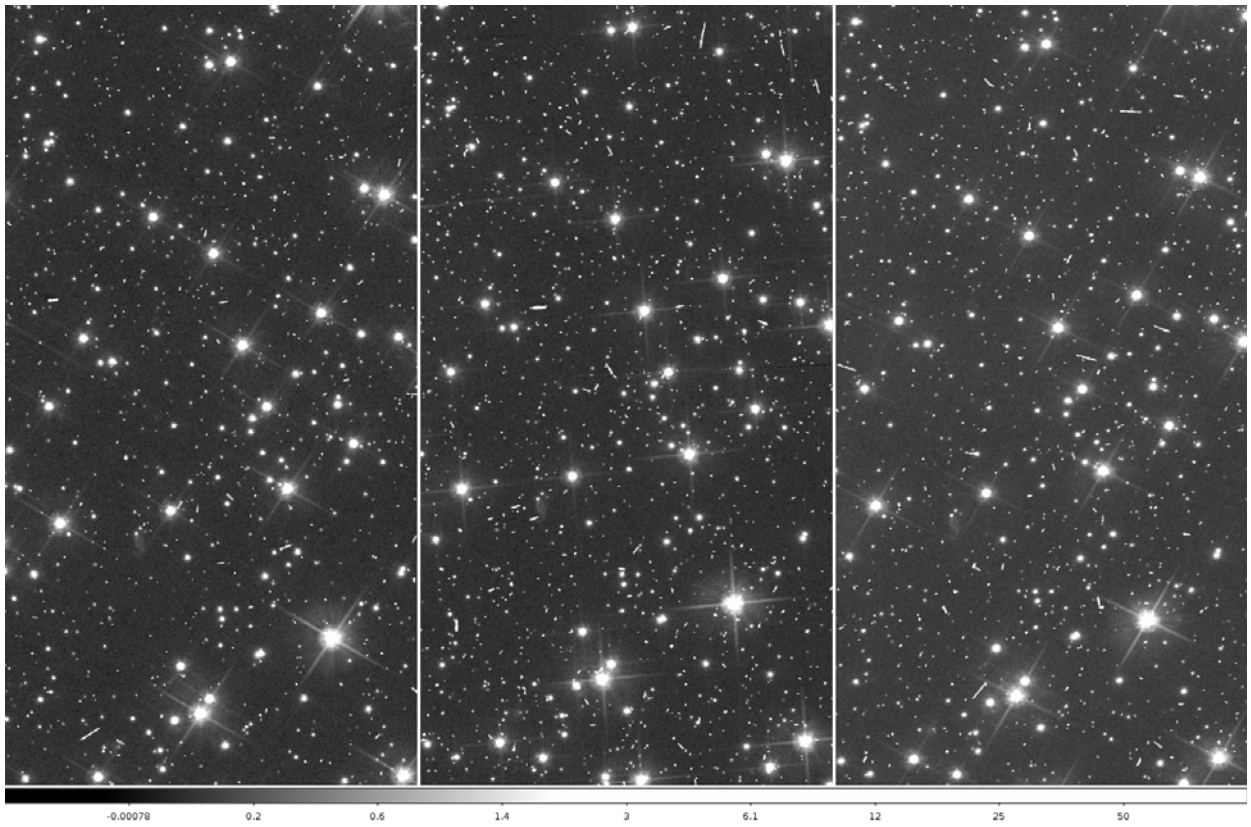
Table 7.5: Summary of Images in this Example

Image Name	Association ID	Proposal ID	Visit & Line Number	Pointing[1] (Degrees)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
j9irw3fwq_flc.fit	None	10737	W3.009	5.697616395115E+00 -7.205982433883E+01	57.3339	2006-05-30	339.00
j9irw4b1q_flc.fits	J9IRW4040	10737	W4.008	5.705688709088E+00 -7.206823392081E+01	93.8806	2006-07-08	339.00
j9irw5kaq_flc.fits	None	10737	W5.022	5.683740916018E+00 -7.207973425822E+01	150.7239	2006-08-31	339.00

1. CRVAL1, CRVAL2 at the reference pixel in [sci,1]

Within a single visit, images from small dithers are usually well-aligned. This is not the case for multiple visits which often utilize different guide star pairs, and could have WCS frames that are offset by as much as 0.5 arcseconds. This effect can be seen in Figure 7.18.

Figure 7.18: Offsets Are Visually Apparent in These Individually-Drizzled Images from Different Visits that Were Aligned Based on Original WCS Information



Stars do not fall in the same location when aligned in ds9, based on the WCS keyword values from each image's header.

7.3.2 Aligning the Images Using tweakreg and imagefindpars

The **tweakreg** task detects sources in multiple images, calculates the shift, rotation and scale between each image and a reference image, then updates the WCS information in the header of each image.

First, **tweakreg** finds sources in each image. This step, which usually involves experimenting with different values of source-finding parameters, varies depending on the type of image. It is often useful to plot source coordinates on the image to see if there are too many false detections, like cosmic rays, then modify the parameter values accordingly. (An example of how to do this can be found in Section 7.5)

Two useful source-finding parameters are the ***computesig*** and ***threshold*** parameters in the **imagefindpars** task (that is called by **tweakreg**). ***computesig*** is a switch for automatically calculating the sky standard deviation in the image, a value

that is important for object detection. The ***threshold*** parameter value is the detection threshold above the local background, in units of sigma. For this example, ***computesig*** retains the default value, ***yes***, and ***threshold*** is set to ***100*** to identify bright stars.

Diagnostic plots created by **tweakreg** are extremely useful for determining the quality of the offset fit solution:

- when ***use2dhist*** and ***see2dplot*** are set to ***yes*** (default settings) in **tweakreg**, a two-dimensional histogram showing the initially-computed offset is displayed.
- When the **tweakreg** parameter ***residplot*** is set to ***both***, two types of plots are displayed: (1) four panels showing fit residuals along the *x*- and *y*-axis; (2) a vector plot showing the size of the residuals across the image.

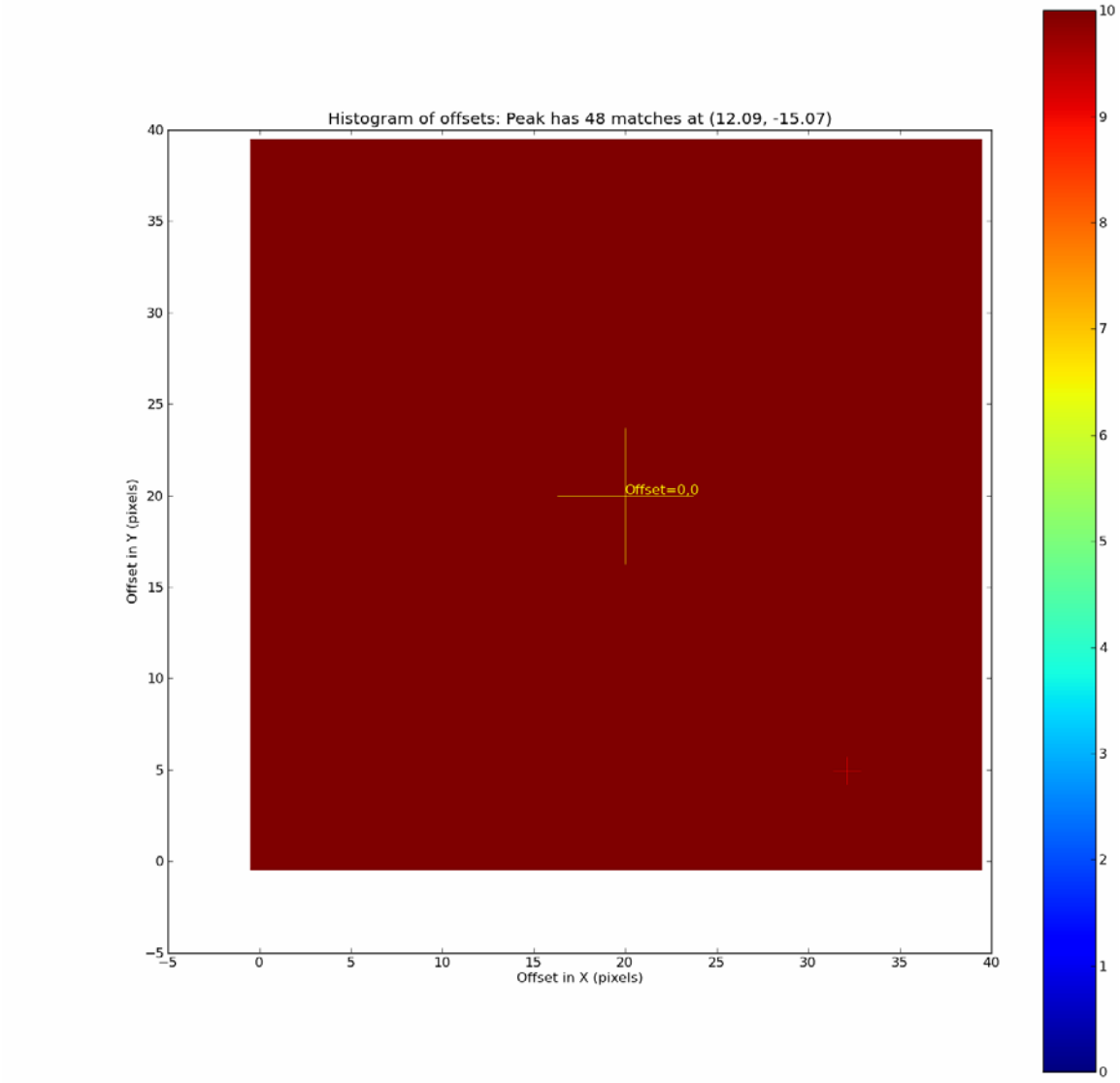
It is usually necessary to run **tweakreg** a few times, adjusting source detection parameters to obtain enough real sources to compute a good offset fit solution. During these tests, the **tweakreg** parameter ***updatehdr*** is set to ***no***. Once a good solution is found, **tweakreg** can be re-run with those same parameters with ***updatehdr*** set to ***yes*** so the WCS information in each image is updated to put them all in the same WCS frame as the reference image.

In this example, **tweakreg** is run from PyRAF as a command-line. Some users may prefer to use the **tweakreg** TEAL GUI that can be opened in PyRAF with the "`epar tweakreg`" command. In that interface, **tweakreg** should be set to its default values by clicking on the "Default" button at the upper right of the window. Then click on the "Edit imagefind parameters" button (third on the list of **tweakreg** parameters) to open **imagefindpars**. In the **imagefindpars** window, reset the parameter values to their defaults, set ***threshold*** to ***100***, then click "Save and Quit" to return to the **tweakreg** window to run the task.

Here, **tweakreg** is run at the command-line in PyRAF:

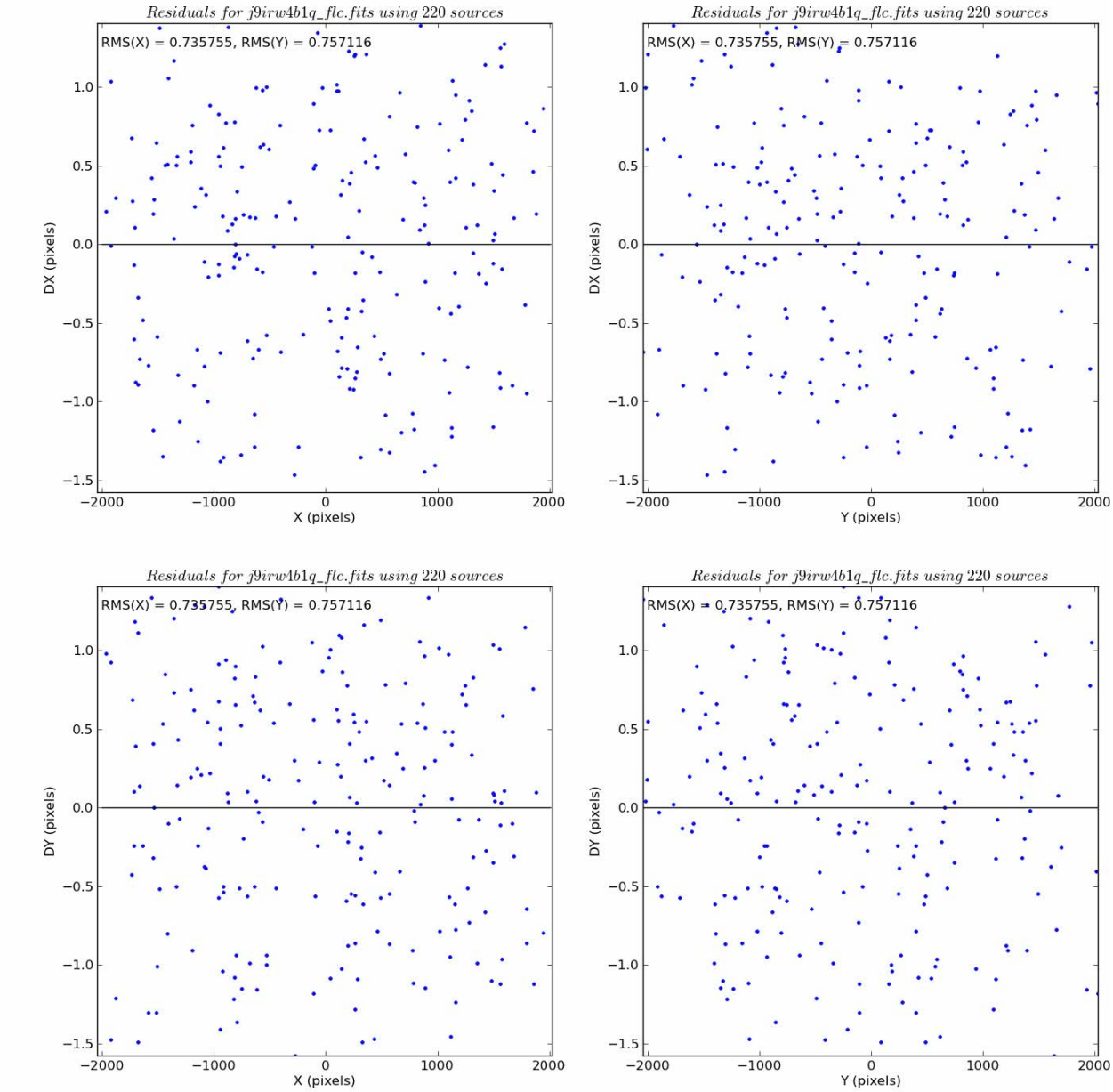
```
--> import drizzlepac
--> from drizzlepac import tweakreg
--> unlearn tweakreg
--> unlearn imagefindpars
--> tweakreg.TweakReg('*flc.fits',threshold=100)
```

Figure 7.19: tweakreg Two-Dimensional Histogram Showing Initial Offset Determination Between Two Images, Using Default Parameters Except *threshold=100*



This two-dimensional histogram shows the initial offset estimate between a reference image and an input image. The tweakreg task measures the *x,y* positions of each "matched" source in both images to determine the images' offsets in *x* and *y*. Not all matched sources are real; some may be cosmic rays or other nearby sources. For a well-defined offset between these two images, the histogram would show a very high number of sources tightly clustered around a specific *x,y* position on the histogram. Here, the default tweakreg settings with *threshold=100* produced a poor offset determination. Out of thousands of sources, offsets appear to be uniformly distributed across the search radius of one arcsecond (20 pixels). The color bar shows that red corresponds to 10 sources, so the uniform red color across the two-dimensional histogram indicates about 10 sources per *x,y* offset bin. The largest number of objects with the same offset, shown by the small cross at the lower right, is only 48 (out of thousands of sources) at x=12.09, y=-15.07 from the center.

Figure 7.20: Residuals Plots Produced by tweakreg, Using Default Parameters Except for *threshold=100*



Note the large RMS values. The RMS should be less than 0.1 pixels.

As **tweakreg** runs, status messages on the screen provide useful information about its progress. For instance, pay attention to the number of objects matched for each image, and the residual RMS for the shifts, scale and rotations. Diagnostic plots are also displayed for each pair of image being aligned and the reference image, as shown in Figures 7.19 and 7.20 that illustrate results for the first image-reference image pair.

In this first attempt, many sources were found in each image but the software was unable to find a good fit in trying to match sources in the image with their counterparts

in the reference image. The two-dimensional histogram in Figure 7.19 showed a random uniform distribution of offsets with a very small peak of only 48 sources having the same offset–out of several thousand objects, this is not statistically significant. An inspection of the images, by blinking the reference image and image indicates that the offset should be around -30 in *x*, and -40 in *y*.

The next two plots produced by **tweakreg** are a vector plot of the residuals (not shown), and a plot of the fit residuals (Figure 7.20). In the residuals plot, a large RMS in both *x* and *y* directions is clearly apparent. This is the result of using the default search radius value, ***searchrad*=*1.0*** arcseconds. For images taken during different visits at different orientations, it is possible that the software could not identify a matching source within a one arcsecond radius because the positional uncertainty of the source is larger than the search aperture. In addition, globular cluster observations have a slightly higher likelihood of locking on a "spoiler" guide star in a crowded field, which would throw off the WCS information into a greater mismatch with the reference image. So, it's worth trying to find matches by running **tweakreg** using incrementally larger ***searchrad*** values to see if the two-dimensional histogram begins to show a well-defined peak.

After trying several more settings, a well-defined offset in the two-dimensional histogram, shown in Figure 7.21, was obtained with ***searchrad=4.0*** arcseconds.

```
--> tweakreg.TweakReg('*flc.fits',threshold=100,searchrad=4.0)
```

This time, many more sources are matched in the images. The two-dimensional histogram of shifts shows a strong peak and the plot of residuals, seen in Figure 7.22, shows a better fit and RMS (a good RMS value is generally less than 0.1 pixels). Additional results are also reported on the terminal, as shown below.

```
Found 7380 matches for j9irw4b1q_flc.fits...
Computed  rscale  fit for  j9irw4b1q_flc.fits :
XSH: -30.0457  YSH: -40.1725   ROT: 0.00465531   SCALE: 1.00001
XRMS: 0.0444336    YRMS: 0.0479033

RMS_RA: 7.20492e-07 (deg)   RMS_DEC: 8.7999e-07 (deg)
```

A small time-dependent skew in the geometric distortion has been corrected for ACS data via improved distortion solutions, as discussed in Appendix A.5.1. While the effect has largely been corrected, a small residual skew (+/-0.05 pixels) may still remain and this may show up as a slope in the astrometric residual plots. An example of this skew is apparent in Figure 7.22. This is a known limitation in the distortion calibration and does not indicate a problem with **tweakreg**.

With this good offset fit, **tweakreg** can be run one last time with the parameter ***updatehdr=yes*** so that the header WCS information for all images are updated to match the WCS of the reference image. In addition, the ***wcsname*** parameter is set to ***TWEAK*** so that the updated set of new WCS header values are identified by the name

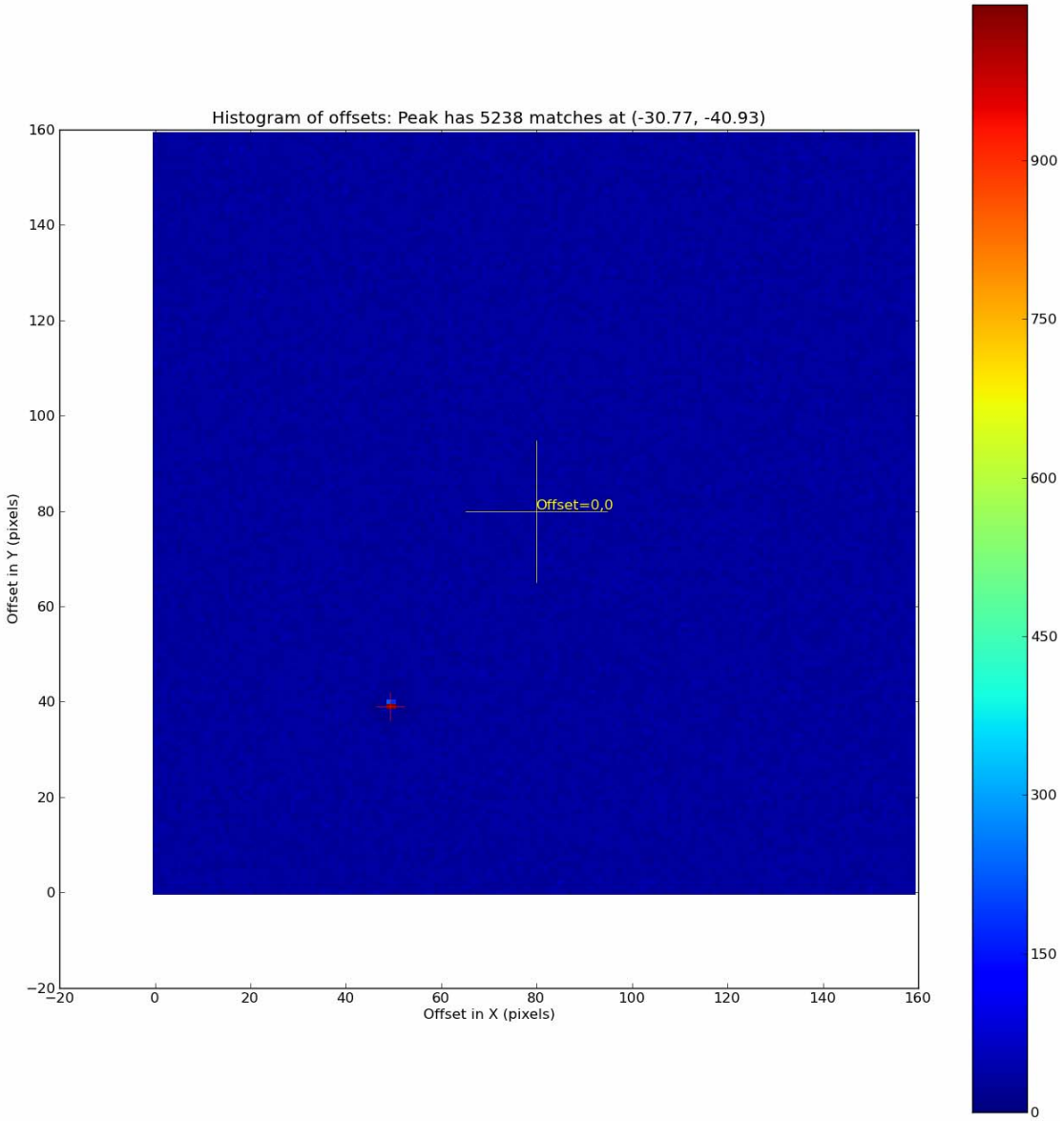
"TWEAK" (old values are still retained in the image header under different keyword names). Since this **tweakreg** run is only done to update the images' WCS, **tweakreg** parameters for displaying plots may be disabled.

```
--> tweakreg.TweakReg('*flc.fits',threshold=100,searchrad=4.0,\
updatehdr=yes,wcsname='TWEAK',see2dplot=False,residplot='None')
```

Once the images are aligned, they can be combined using **astrodrizzle**. For this example, only default values were used to get a quick look at the single-drizzled aligned science images (Figure 7.23) and the final combined image (Figure 7.24).

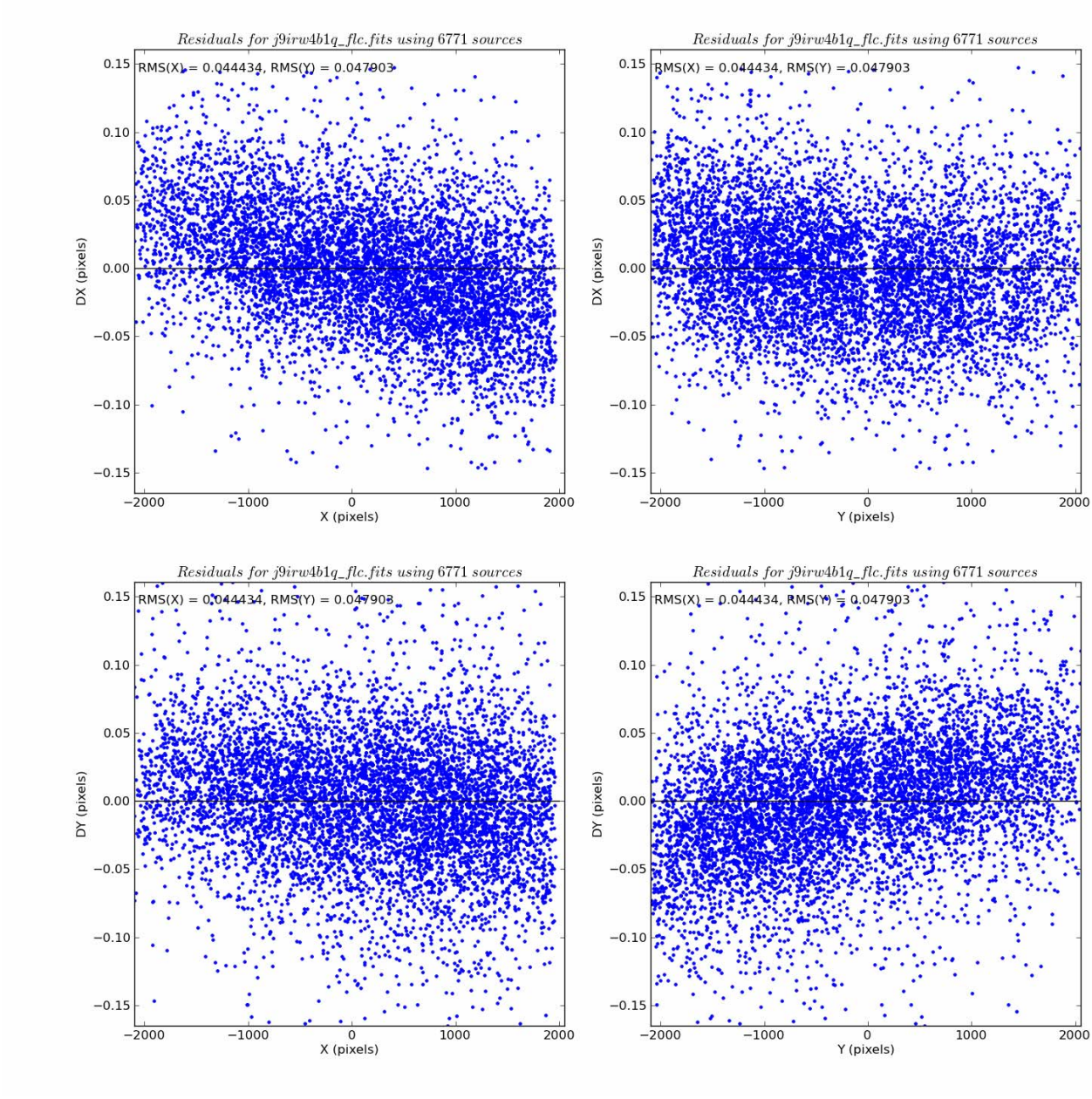
```
--> astrodrizzle.AstroDrizzle('*flc.fits')
```

Figure 7.21: Two-Dimensional Histogram With a Well-Defined Peak After Increasing the tweakreg Search Radius



The two-dimensional histogram of the *x* and *y* offsets between all the sources in the reference and input images. In this case the search radius has been expanded to 4 arcseconds (80 pixels, note the different axes from Figure 7.19). A large number of sources were found to have a similar offset, as shown by the red peak in the lower left, while the rest of the image contains random matches shown in blue. These offset values corresponding to the peak are a good first approximation of the offset between the two images.

Figure 7.22: Residuals Plot Corresponding to Figure 7.21



A solution to the shifts was clearly found now that stars have been matched in different images.

Figure 7.23: Section of Each Three Single-Drizzled Science Images Displayed in DS9 and Aligned by the New "TWEAK" WCS

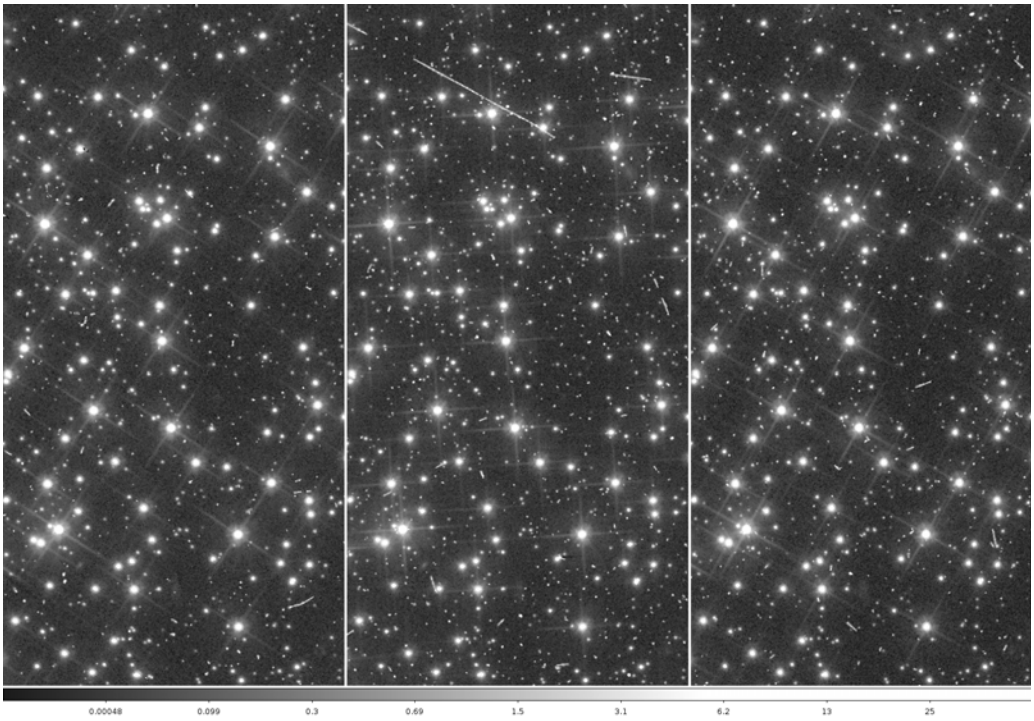
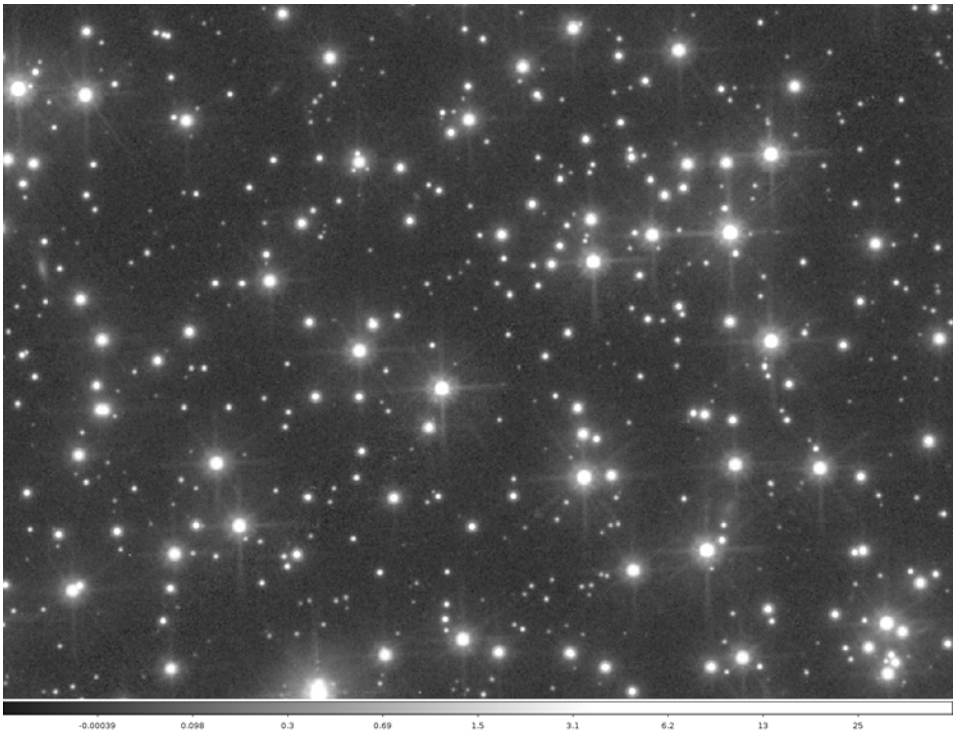


Figure 7.24: Portion of the Final Combined Image



The final combined image shows no evidence of misalignment. A more careful check of the alignment can be done by checking the PSF of a few stars across the field using a task like imexamine.

7.4 ACS/WFC: Optimizing the Image Sampling for a Single Visit

Introduction

This example describes the combination of four ACS/WFC images, obtained using the ACS-WFC-DITHER-LINE pattern, which includes subsampling in the *x* and *y* directions, and a large offset over the 50 pixel-wide WFC chip gap. Several **astrodrizzle** runs using different parameter values are used to determine the best settings for optimizing image sampling.

Summary of Steps

1. Description of the data.
2. Run **astrodrizzle** with ***final_scale*** of ***0.03*** arcseconds using three ***final_pixfrac*** setting: ***0.1***. ***0.6***, and ***1.0***.
3. Evaluate the results to determine which parameters provide improved image resolution while preserving signal-to-noise to an acceptable level.

7.4.1 Description of the Data

Data for this example are images of the galaxy NGC 4449 (Program 10585), obtained by the ACS/WFC using the F555W filter. These four images were obtained using primary and secondary two-point ACS-WFC-DITHER-LINE patterns. The primary dither pattern had a large offset to cover the 50 pixel-wide WFC chip, while the secondary dither pattern was a two-point subpixel dither at each point in the primary pattern. Offsets in the detector *x* and *y* direction are shown in Table 7.6 as POS TARG values in arcseconds.

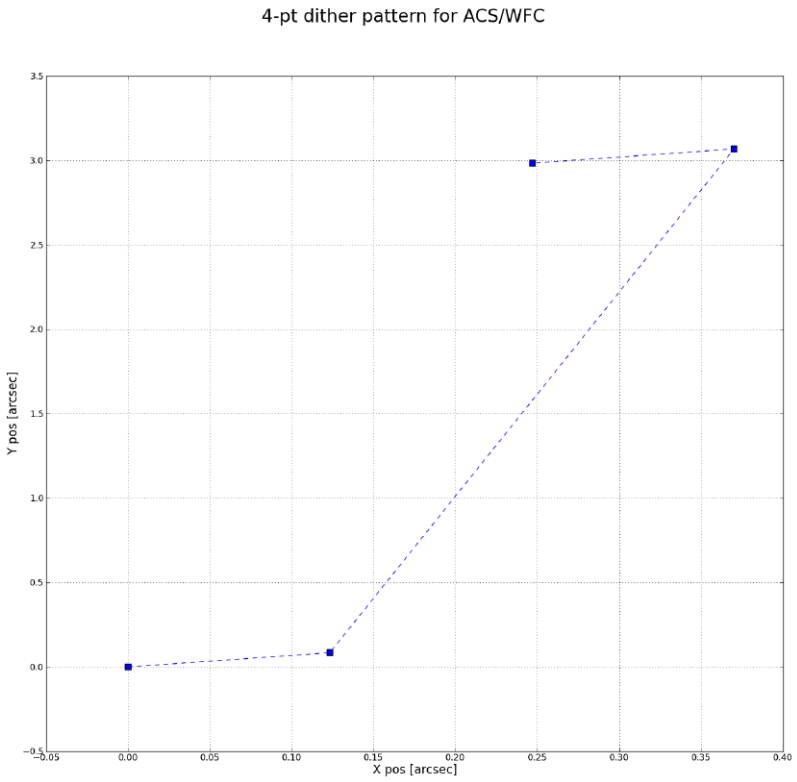
Table 7.6: Summary of Images in this Example

Image Name	Association ID	Proposal ID	Visit & Line Number	POS TARG[1] (x,y in arcsec.)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
j9cd01kqq_flc.fits	J9CD01020	10585	01.002	0.0000, 0.0000	141.2433	2005-11-10	608.00
j9cd01l5q_flc.fits	J9CD01020	10585	01.002	0.1232,0.0839	141.2433	2005-11-10	615.00
j9cd01mfq_flc.fits	J9CD01020	10585	01.002	0.2469,2.9838	141.2433	2005-11-10	616.00
j9cd01ldq_flc.fits	J9CD01020	10585	01.002	0.3700,3.0677	141.2433	2005-11-10	621.00

1. POS TARG are commanded offsets. The actual measured offsets would be slightly different.

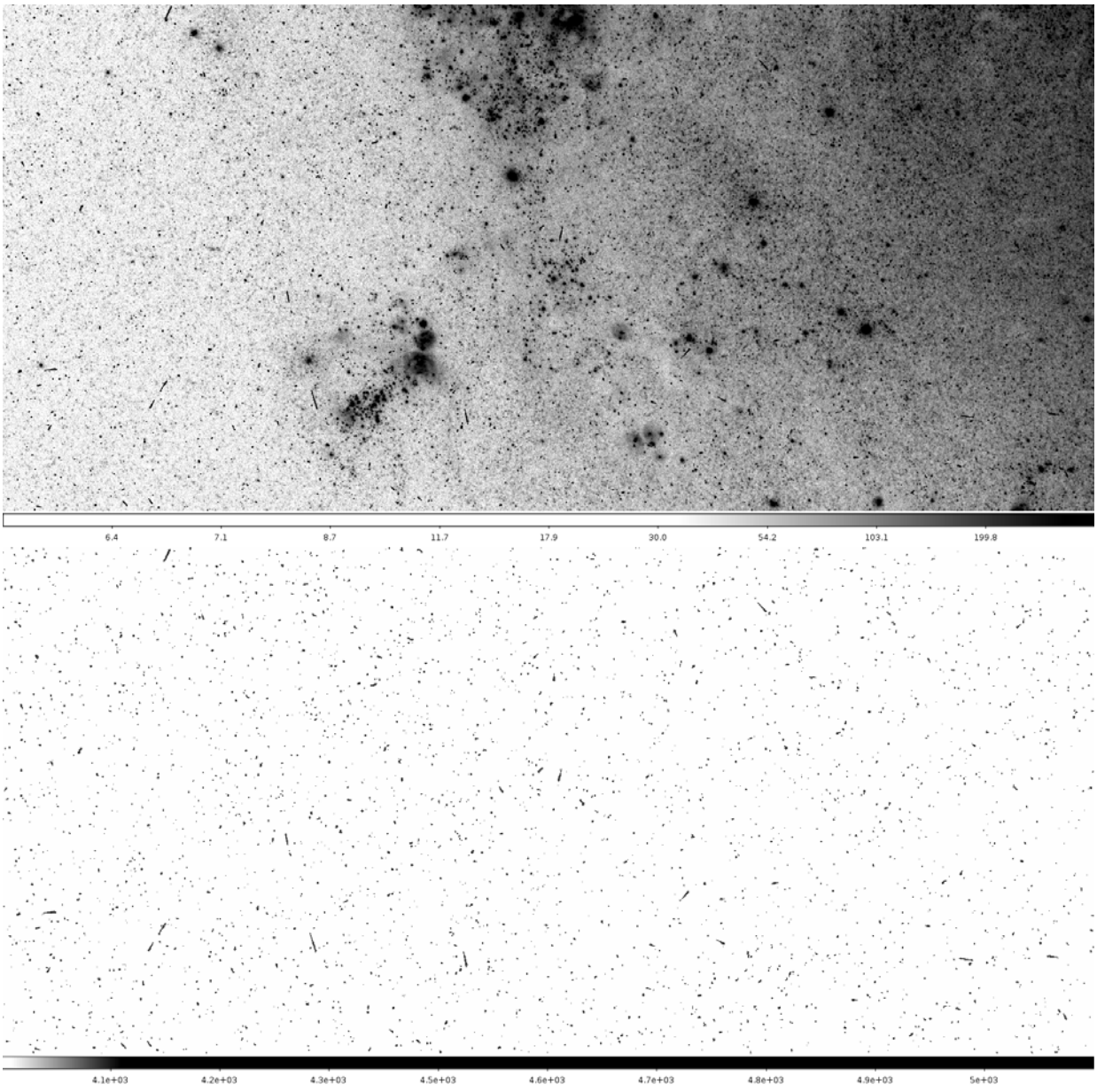
These images are flat-field calibrated by **calacs**, and the `flc.fits` extension indicates they were corrected for CTE.

Figure 7.25: Dither Pattern Used for these Observations



Pattern includes subpixel sampling and large offset in *y* for covering the CCD gap.

Figure 7.26: Science (Top) and Data Quality Arrays (Bottom) of the First Image in the Association



The bottom image has been scaled to only show cosmic ray bits.

7.4.2 Combining the Images using astrodrizzle

As described in Section 2.2, subpixel dithering has the potential to improve the resolution of the final combined image. Input pixels may be "shrunk" before being transformed to the subsampled output image. This is done to create a smaller input "footprint" in the output frame. The shrunken pixels, also called "drops," rain down on the subsampled output frame where each shrunken input pixel is averaged into an

output pixel with a weight proportional to the area of overlap between the "drop" and the output subsampled grid.

If the drop size is too small, not all output pixels will have data added to them from each input image. Care should be taken in selecting a drop size that is small enough to avoid degrading the image, but large enough to provide reasonably uniform coverage in the final output image. The drop size is controlled by the **astrodrizzle** parameter called ***final_pixfrac***, which is the ratio of the linear size of the drop to the input pixel (before geometric distortion corrections). The level of subsampling for the output image is specified by the ***final_scale*** parameter.

Values for ***final_pixfrac*** and ***final_scale*** depend on the number of input images and the size of the shifts. In this example, the dither pattern provides some subsampling so the resolution can be improved. Ideally, the best scale is half that of the native scale, but to avoid degrading the PSF, users should not excessively decrease the value of the ***final_scale*** parameter.

For this example, a ***final_scale*** value of ***0.03*** arseconds/pixel is adopted, which is 60% the linear size of the native plate scale (0.05 arcseconds/pixel.). A series of experiments will be run to determine the best ***final_pixfrac*** value for the selected output image scale.

As suggested in the *HST Dither Handbook*, statistics performed on the drizzled weight image should yield a RMS/median value less than 0.2. This threshold controls the trade-off between improving image resolution versus increasing background noise due to pixel resampling.

A series of **astrodrizzle** commands, executed in PyRAF, can be written to create a test grid of different ***final_pixfrac*** values. Default values have been used for the parameters, except for ***final_pixfrac*** and ***final_scale***. In the example below, three ***final_pixfrac*** values are used to illustrate its effect on the image quality: ***1.0***, ***0.6***, and ***0.1***.

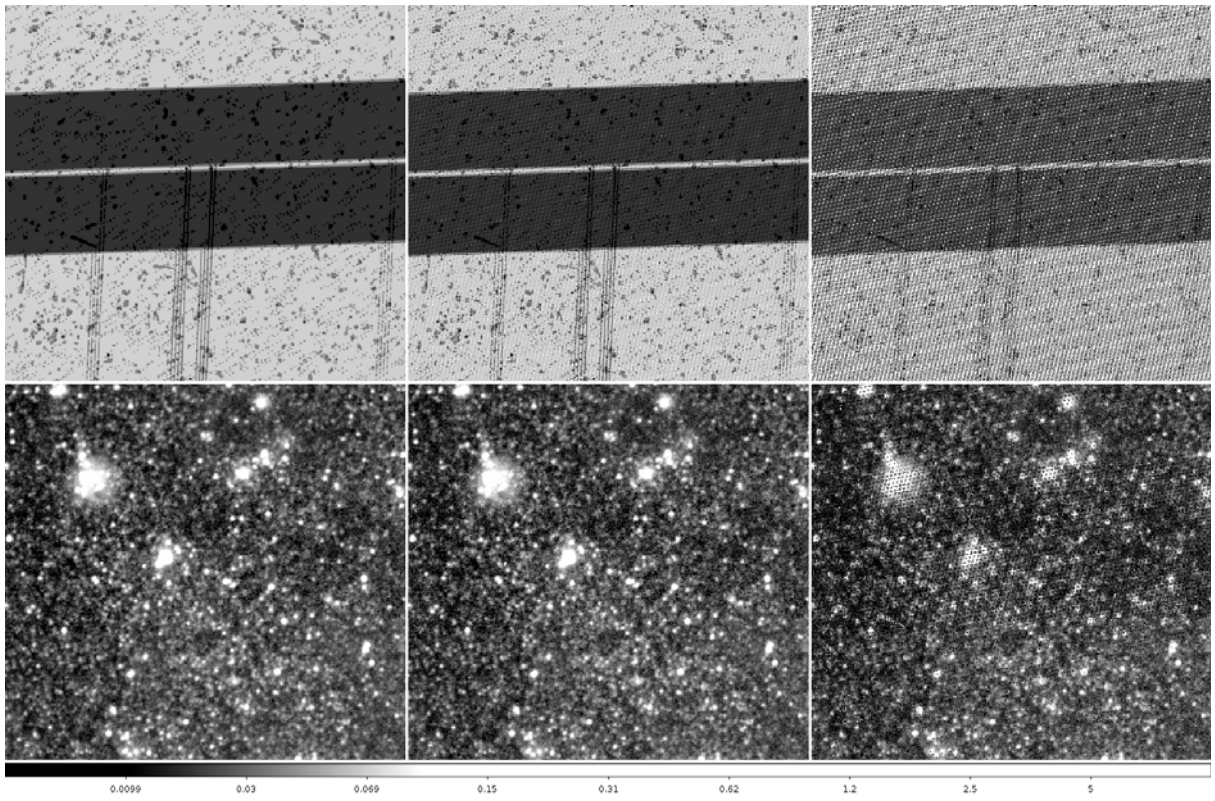
```
--> import drizzlepac
--> from drizzlepac import astrodrizzle
--> unlearn astrodrizzle
--> astrodrizzle.AstroDrizzle('*flc.fits',output='test1',final_scale=0.03,final_pixfrac=1.0)
--> astrodrizzle.AstroDrizzle('*flc.fits',output='test2',final_scale=0.03,final_pixfrac=0.6)
--> astrodrizzle.AstroDrizzle('*flc.fits',output='test3',final_scale=0.03,final_pixfrac=0.1)
```

Each **astrodrizzle** run produces three images; the science file (suffix `drc_sci.fits`), the weight map (`drc_wht.fits`), and the context image (`drc_ctx.fits`).

Figure 7.27 shows the central region of the science and weight images produced by ***final_pixfrac*** values of ***1.0***, ***0.6***, and ***0.1***. The science image with the smallest ***final_pixfrac***, ***0.1***, shows a noisy background with hot pixels improperly removed. There are "holes" in the images where no input pixels fall into the output grid because they have been shrunk too small. Inspection of the weight map corresponding to the smallest ***final_pixfrac*** value shows many places with weights of zero, indicating that a

final_pixfrac value of 0.1 is clearly too small, and was only included in this example for illustrative purposes.

Figure 7.27: Results for Three Tests with final_pixfrac Values of 1.0, 0.6, and 0.1

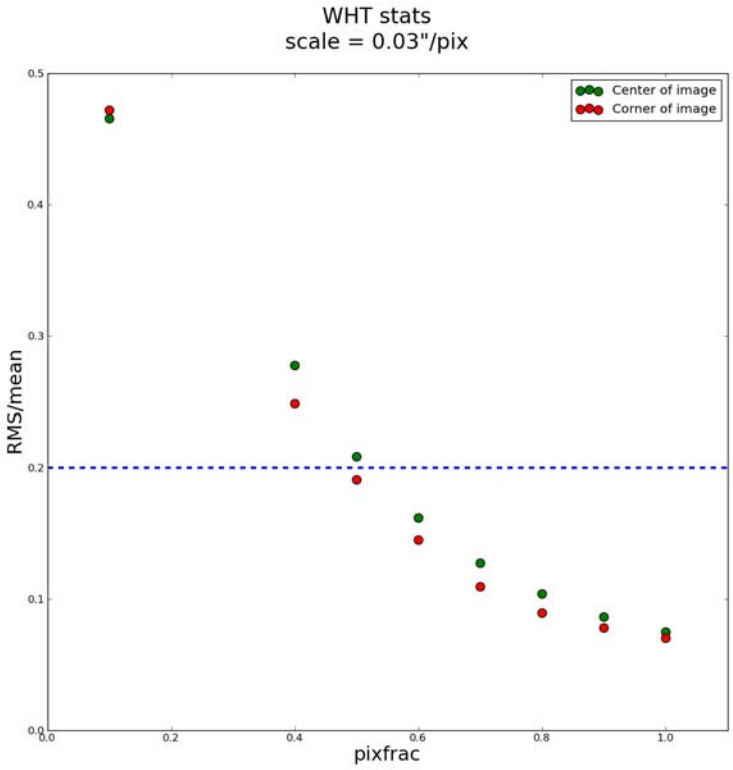


Left column corresponds to *final_pixfrac =1.0***, the middle column corresponds to a** *final_pixfrac = 0.6***, and the right column was created using** *final_pixfrac = 0.1***. The top row shows the weight maps, and the bottom row shows the science frames.**

Weight statistics, (RMS/median) for each trial are obtained for a 500x500 pixel box in the center and lower right corner of each weight image. Statistics for the weight image created with ***final_pixfrac*** of ***0.6*** meet the general requirement of RMS/median < 0.2. A visual inspection of the weight image shows how the RMS varies over the field of view, due to changes in geometric distortion over the chip where input pixels cover different areas on the sky. (For more information, refer to the discussion on Pixel Area Maps at the ACS website.)

Determining the best solution for drizzle-combining images will also depend on the position of the target on the detector. For a compact source in the lower right corner of the image, one could set ***final_pixfrac*** to 0.5. For this example, however, objects are distributed over the entire field so a ***final_pixfrac*** of ***0.6*** is found to provide the best result, where the RMS/median does not exceed 0.2 on any portion of the detector.

Figure 7.28: Weight Map Statistics in a 500x500 Pixel Box at the Center and Lower Right Corner of the Image



7.5 WFC3/UVIS: Aligning Images by Defining an Output Reference Frame

Introduction

The following example describes the alignment of images in two filters obtained with the WFC3/UVIS channel. While **tweakreg** aligns the image World Coordinate System (WCS) header information, this example demonstrates how to define the output frame reference image for aligning the two filters in pixel space. This example can be generalized for aligning images obtained in different cameras, for example WFC3/UVIS to ACS/WFC.

Step-by-step instructions are provided below, in a cookbook fashion, with minimal discussion of parameters. Commands are given using Python syntax executed in PyRAF. Since the tasks have been restored to default values prior to execution, mostly non-default parameters are specified in the command-line examples.

Summary of Steps

1. Description of the data.
2. Use **tweakreg** to align images in a given filter.
3. For each filter, combine the images using **astrodrizzle** to produce high signal-to-noise drizzled products, free from cosmic rays and detector artifacts.
4. Improve the alignment between the two *drizzled* images, each representing a filter, using **tweakreg**.
5. Using **tweakback**, propagate the new "tweaked" solutions obtained from aligning the two drizzled images back to their respective original `flt.fits` images.
6. Drizzle-combine the updated `flt.fits` images for each filter, using the improved WCS offsets, to create WCS-aligned drizzled images for each filter.
7. Demonstrate how defining a reference frame in **astrodrizzle** can create drizzled images for each filter that are aligned, in pixel space, to a common "output grid."

7.5.1 Description of the Data

In August 2009, the first epoch of M83 observations were obtained as part of the WFC3 Early Release Science program (Proposal ID 11360). Three images with small dithers (`POS TARG`s) were obtained, in each filter, to fill the gap between the two UVIS chips and to allow for rejection of cosmic rays and detector artifacts. This example will focus on aligning two sets of images, one in the F814W filter and the other in F438W, each obtained in different visits.

The datasets used in this example are listed in Table 7.7. Note that the images have been given different names for clarity. While the `flt.fits` suffix has been dropped in the new name, users should keep in mind that these are calibrated `flt.fits` products from the pipeline.

Table 7.7: Summary of Images for this Example

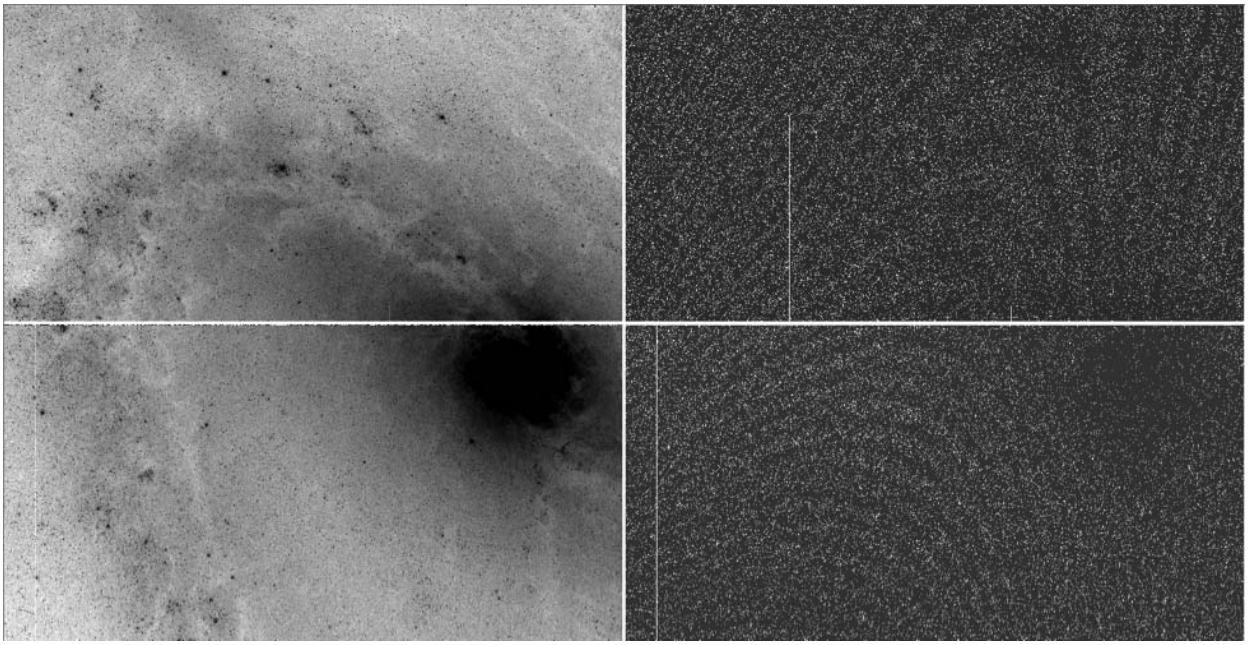
Unique & Orig. Image Name	Association ID	Proposal ID	Visit & Line Number	POS TARG (x,y in arcsec.)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
f814w_pos1_01.fits ib6w62tvq_flt.fits	IB6W62060	11360	62.022	0.0000,0.0000	305.0004	2009-08-26	401.00
f814w_pos1_02.fits ib6w62txq_flt.fits	IB6W62060	11360	62.023	1.4460,2.9260	305.0001	2009-08-26	401.00
f814w_pos1_03.fits ib6w62tzq_flt.fits	IB6W62060	11360	62.024	-1.4460,-2.9260	305.0007	2009-08-26	401.00
f438w_pos1_01.fits ib6w61uaq_flt.fits	IB6W61050	11360	61.001	0.0000,0.0000	305.0004	2009-08-26	600.00

Table 7.7: Summary of Images for this Example (Continued)

Unique & Orig. Image Name	Association ID	Proposal ID	Visit & Line Number	POS TARG (x,y in arcsec.)	PA_V3 Orientation (degrees)	Observation Date	Exposure Time (sec.)
f438w_pos1_02.fits ib6w61umq_flt.fit	IB6W61050	11360	61.008	1.4460,2.9260	305.0001	2009-08-26	640.00
f438w_pos1_03.fits ib6w61uzq_flt.fits	IB6W61050	11360	61.015	-1.4460,-2.9260	305.0007	2009-08-26	640.00

Science and data quality extensions for one of the calibrated pipeline products, the first F814W image, are shown in Figure 7.29. Note that the pipeline was overly aggressive at flagging cosmic rays, as reflected in the large number of white pixels in the data quality (DQ) array; this is likely due to a poor estimate of the sky background. These cosmic ray flags, with the value of 4096 in the `flt.fits` DQ array, will be, by default, automatically reset to be treated as "good" pixels by **astrodrizzle** (using ***resetbits=4096***). Reprocessing the dataset with different parameter values will then allow the user to manually optimize cosmic ray rejection.

Figure 7.29: Science and Data Quality Extensions for the First F814W Calibrated (`flt.fits`) Image



The science extensions, [sci,2] and [sci,1], on the left, with their corresponding data quality extensions [dq,2] and [dq,1], on the right. Note that the pipeline did a poor job in flagging cosmic rays, resulting in a pattern which reflects the noise in the sky background. Manual reprocessing with astrodrizzle will allow these flags to be reset and optimized by the user.

7.5.2 Align Images for Each Filter Set Using tweakreg

Before running **tweakreg**, the **drizzlepac** package has to be loaded and the required tasks for this example have to be imported and reset to default values.

```
--> import drizzlepac
--> from drizzlepac import tweakreg
--> from drizzlepac import astrodrizzle
--> from drizzlepac import tweakback

--> unlearn tweakreg force=yes
--> unlearn imagefindpars force=yes
--> unlearn astrodrizzle force=yes
--> unlearn tweakback force=yes
```

This target has few actual point sources, so the alignment is primarily based on the positions of H II regions. The commands below reflect parameters that have been optimized after several interactive trials.

The parameter value for ***conv_width*** is recommended to be approximately twice the FWHM of the PSF. By default, **imagefindpars** will automatically compute the sigma value (standard deviation) of the sky background, so it is necessary to adjust the ***threshold*** parameter in **imagefindpars** to ensure an adequate number of sources are found to compute the fit. (For images containing a large number of saturated objects, the **imagefindpars** parameter ***peakmax*** can be set to exclude those sources from the fit.)

Using the **tweakreg** parameters shown below (that also includes parameters from the **imagefindpar** task that is used by **tweakreg** for object detection settings), several thousand sources are initially detected with several hundred matches, providing solutions accurate to much better than 0.1 pixels r.ms. Users are strongly encouraged to inspect the four-panel residual plot for signs of remaining systematics in the final residuals, like, for example, a slope which indicates that a residual rotation or scale still remains in the data. Instructions for examining the source catalogs are given in Section 7.5.3.

```
--> tweakreg.TweakReg('f814w_pos1_??.fits,conv_width=3.5,\
threshold=200,shiftfile=True,outshifts='shift_814_pos1.txt',updatehdr=False)

-->tweakreg.TweakReg('f438w_pos1_??.fits,conv_width=3.5,\
threshold=200,shiftfile=True,outshifts='shift_438_pos1.txt',updatehdr=False)
```

In these **tweakreg** line-commands in PyRaF (that can also be run in the Python interface), the following settings were used:

- ***conv_width=3.5***, the convolution kernel which is twice the PSF FWHM of sources. It identifies potential sources which look closest to a gaussian with the FWHM equal to this parameter's value.
- ***threshold=200*** sigma above the local background for object detection.
- ***updatehdr=False*** so that the offsets are not used to modify the WCS information in each image.
- ***shiftfile=True*** and ***outshifts='shift_438_pos1.txt'*** so that a shift file containing the computed offsets are written to a named file. The contents of the two output shift files are shown below. Shift files are only created for record-keeping purposes. Unlike **multidrizzle, astrodrizzle** does not accept a shift file as input because offsets are directly applied to the WCS in the image headers.

F814W and F435W shift files:

```
Image               dX         dY         drot        scale     xfit_rms  yfit_rms
f814w_pos1_01.fits  0.000000   0.000000   0.000000    1.000000  0.000000  0.000000
f814w_pos1_02.fits  -0.055635  -0.079298  0.000071    1.000000  0.045810  0.052170
f814w_pos1_03.fits  -0.045426  -0.022232  359.999817  0.999998  0.044504  0.054453

Image               dX         dY         drot        scale     xfit_rms  yfit_rms
f438w_pos1_01.fits  0.000000   0.000000   0.000000    1.000000  0.000000  0.000000
f438w_pos1_02.fits  -0.098221  0.042909   0.000385    1.000000  0.069631  0.066566
f438w_pos1_03.fits  0.046986   0.160370   359.999612  1.000001  0.067033  0.061197
```

When **tweakreg** is run with its default setting ***writecat=True***, it creates source catalogs with extension "`.coo`" for each science image extension.

tweakreg also automatically generates an ASCII table listing the names of the catalogs generated for each image. The name of the table can be specified by the parameter ***catfile***, but if no name is provided, the default name is <*reference_file*>`_xy_catfile.list`. For this set of data, the default name of the ***catfile*** is `f814w_pos1_01_xy_catfile.list`, and its contents are shown below:

```
f814w_pos1_01.fits f814w_pos1_01_sci1_xy_catalog.coo f814w_pos1_01_sci2_xy_catalog.coo
f814w_pos1_02.fits f814w_pos1_02_sci1_xy_catalog.coo f814w_pos1_02_sci2_xy_catalog.coo
f814w_pos1_03.fits f814w_pos1_03_sci1_xy_catalog.coo f814w_pos1_03_sci2_xy_catalog.coo
```

When the user is satisfied with the results, **tweakreg** should be run one last time with ***updatehdr=True***, to update the WCS of each `flt.fits` image with the final solution. At this point, it is no longer necessary to run the task in interactive mode. To update the WCS information in the image headers as a batch, plotting parameters can

be turned off (as shown below) and the task will run without prompting the user to verify the solutions.

```
--> tweakreg.TweakReg('f814w_pos1_??.fits,writecat=False,\
catfile='f814w_pos1_01_xy_catfile.list',residplot='NoPlot',\
see2dplot=no,updatehdr=True,wcsname='TWEAK_814')

--> tweakreg.TweakReg('f438w_pos1_??.fits,writecat=False,\
catfile='f438w_pos1_01_xy_catfile.list',residplot='NoPlot',\
see2dplot=no,updatehdr=True,wcsname='TWEAK_438')
```

This final **tweakreg** run uses the same settings as the previous run, except that the WCS information in the image headers are updated (***updatehdr=True***). In addition, ***writecat*** is set to ***False*** and the ***catfile*** parameter points to a previously-made listing of catalogs so that tweakreg makes use of existing catalogs to save processing time.

When ***updatehdr*** is set to ***True***, the updated WCS solutions can be tracked via a unique name, given by the **tweakreg** parameter ***wcsname***. (The default value is the string `TWEAK`). It is useful to choose a meaningful name to keep track of it when querying the `flt.fits` headers.

The example below shows the query results for an image header with several different WCSs, each identified by a unique name that reflects the history of the dataset.

- `WCSNAMEO=OPUS`: the original WCS name from OPUS pipeline processing
- `WCSNAMEA=IDC_v5r1512gi:` the WCS corresponding to the distortion reference file (IDCTAB) from running the task **updatewcs**
- `WCSNAMEB=TWEAK_438` is a user-generated solution from this example
- `WCSNAME =TWEAK_438` is also the most current WCS

To query the `WCSNAME` keywords in the image header, the following IRAF command may be used:

```
--> hedit f???w_pos?_??.fits[1] wcsnam* .
f814w_pos1_01.fits[1],WCSNAMEO = OPUS
f814w_pos1_01.fits[1],WCSNAMEA = IDC_v5r1512gi
f814w_pos1_01.fits[1],WCSNAMEB = TWEAK_814
f814w_pos1_01.fits[1],WCSNAME = TWEAK_814
f438w_pos1_01.fits[1],WCSNAMEO = OPUS
f438w_pos1_01.fits[1],WCSNAMEA = IDC_v5r1512gi
f438w_pos1_01.fits[1],WCSNAMEB = TWEAK_438
f438w_pos1_01.fits[1],WCSNAME = TWEAK_438
```

7.5.3 Overplot Matched Sources Onto the Original flt.fits Image

While this step is optional, it is highly recommended.

The commands below use PyRAF tasks to display a chip in a `flt.fits` image, overplots the initial source catalog "`image_sci?_xy_catalog.coo`"[8] (in red) on the image, then overplots the matched catalog "`image_catalog_fit.match`" (in green). The matched catalog shows which sources were used to compute the **tweakreg** solution, allowing the user to verify that actual sources (not cosmic rays or hot pixels) were used to determine this solution (See Figure 7.30).

Matched catalog files (with suffix "`catalog_fit.match`") are created for all images except the first in the list, which serves, by default, as the reference. Columns 11 and 12 give the original position in the `flt.fits` images, and column 15 (labeled "`EXTVER ID`" in `*catalog_fit.match` files) specifies the chip that corresponds to the coordinates.

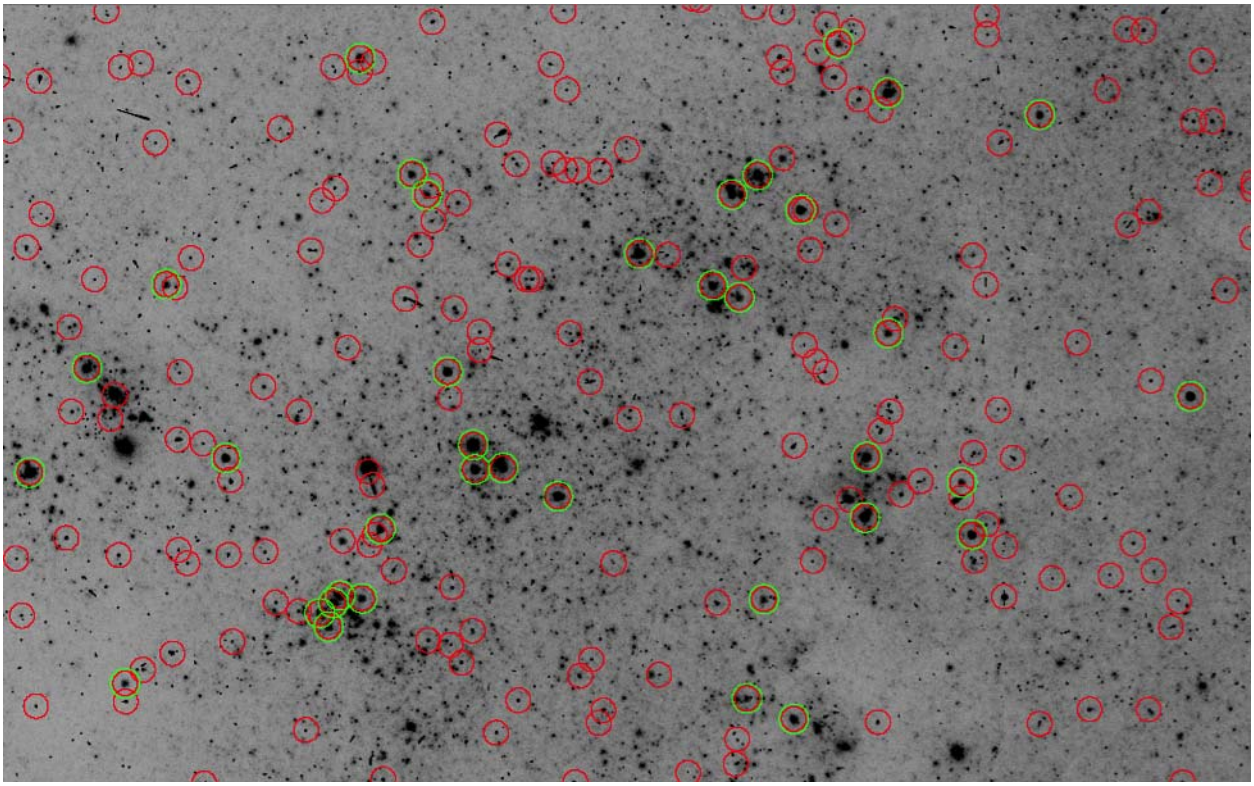
```
-> tselect f814w_pos1_02_catalog_fit.match \
f814w_pos1_02_catalog_fit_sci1.match "c15 .eq. 1"
--> tselect f814w_pos1_02_catalog_fit.match \
f814w_pos1_02_catalog_fit_sci2.match "c15 .eq. 2"

--> tproject f814w_pos1_02_catalog_fit_sci1.match \
f814w_pos1_02_catalog_fit_sci1.xyflt "c11,c12"
--> tproject f814w_pos1_02_catalog_fit_sci2.match \
f814w_pos1_02_catalog_fit_sci2.xyflt "c11,c12"

--> display f814w_pos1_02.fits[sci,2] 1
--> tvmark 1 f814w_pos1_02_sci2_xy_catalog.coo mark=circle radii=5 color=204
--> tvmark 1 f814w_pos1_02_catalog_fit_sci2.xyflt mark=circle \
radii=7 color=205
```

8. "?" is a single-character wildcard for the extension number, either 1 or 2 for UVIS. The names in quotes are files suffixes for the catalogs, as illustrated in the command examples. In this example, chip 1 of the first F814W image is displayed, which corresponds to science extension 2.

Figure 7.30: Source Catalogs Over-Plotted on a `flt.fits` Image



This is a useful way to inspect objects used for computing the offset fit. Matched sources are shown in green.

Create a Drizzle-combined Image for Each Filter

Note this common "gotcha:"
for WFC3/UVIS and ACS/WFC images,
EXTVER ID = 2 = [sci,2] = chip1 = flt.fits[4]
EXTVER ID = 1 = [sci,1] = chip2 = flt.fits[1]

For each filter, images are drizzle-combined to create a high signal-to-noise image, free from cosmic rays or detector artifacts. These two drizzled image will, themselves, be aligned in a later step.

```
--> astrodrizzle.AstroDrizzle('f814w_pos1_??.fits',\
output='f814w_pos1',driz_sep_bits=64,32',driz_cr_corr=yes,\
final_bits='64,32',final_wcs=yes,final_scale=0.0396,final_rot=0.)

--> astrodrizzle.AstroDrizzle('f438w_pos1_??.fits,\
output='f438w_pos1',driz_sep_bits='64,32',driz_cr_corr=yes,\
final_bits='64,32',final_wcs=yes,final_scale=0.0396,final_rot=0.)
```

In the **astrodrizzle** commands shown above,

- ***driz_sep_bits*** and ***final_bits*** set to "***64,32***" tells **astrodrizzle** to treat input pixels flagged in the `flt.fits` data quality extensions with the value 64 (CTE tails) and 32 (warm pixels) as "good" pixels.
- ***driz_cr_corr =yes*** is a switch to tell the software to create a cosmic ray mask for each input image.
- f***inal_wcs= yes*** because there are two "custom" output image parameter settings in that category:
  - ***final_scale***, the final output scale, is set to ***0.0396*** arcseconds/pixel
  - ***final_rot=0***[9] orients the output image such that its *y*-axis is relative to the north (at the top of the image).

Note:

Information about the primary WCS solution for each input image that was used to create the drizzled image is stored in a new drizzled file extension of type `HDRTAB`. The `WCSNAME` in the drizzled image header itself is set to the value `DRZWCZ`.

```
--> hedit *drz_sci.fits wcsnam* .
f814w_pos1_drz_sci.fits,WCSNAME = DRZWCS
f438w_pos1_drz_sci.fits,WCSNAME = DRZWCS
```

Next, the data products are inspected to look for any offsets. In **ds9**, the images may be matched by the WCS or in physical (or pixel) units, and then blinked. In this case, the positions of stars in each science array appear to be offset by approximately (+1,+6) pixels.

```
--> displ f438w_pos1_drz_sci.fits 1 zs-zr-z1=-0.2 z2=0.5 \
ztran=log fill+
--> displ f814w_pos1_drz_sci.fits 2 zs-zr-z1=-0.2 z2=0.5 \
ztran=log fill+
```

The weight images should also be inspected. In this case, the final weight image parameter was set to ***EXP***, so the `WHT` images for each filter resemble an exposure time map of the combined dataset. For UVIS data, it is important to verify that an imprint of sources is not apparent in the weight image, which could imply that one or more frames were being excluded from the final product.

```
--> displ f438w_pos1_drz_wht.fits 3 fill+
--> displ f814w_pos1_drz_wht.fits 4 fill+
```

9. Setting ***final_rot*** to the default value, ***none*** or ***INDEF***, will orient the output image to the same orientation as the reference image.

Figure 7.31: Science Extension of the F814W Drizzled Product Oriented North-Up

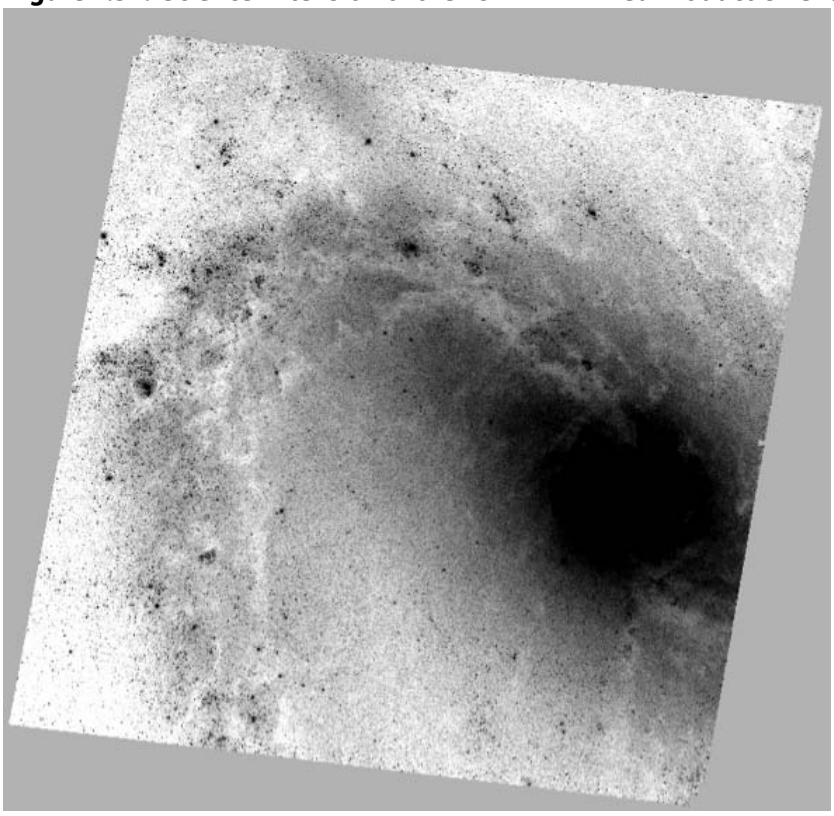
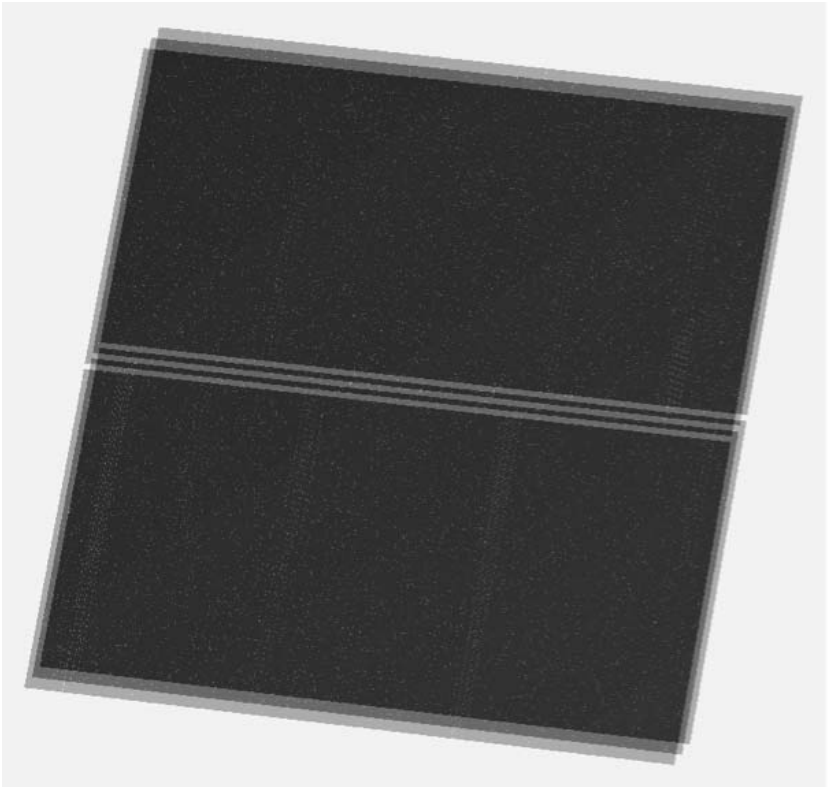


Figure 7.32: Weight Extension of the F814W Drizzled Product



The gap dither pattern can be seen, as well as pixels with DQ flags and cosmic rays which were treated as "bad" in the final drizzle step.

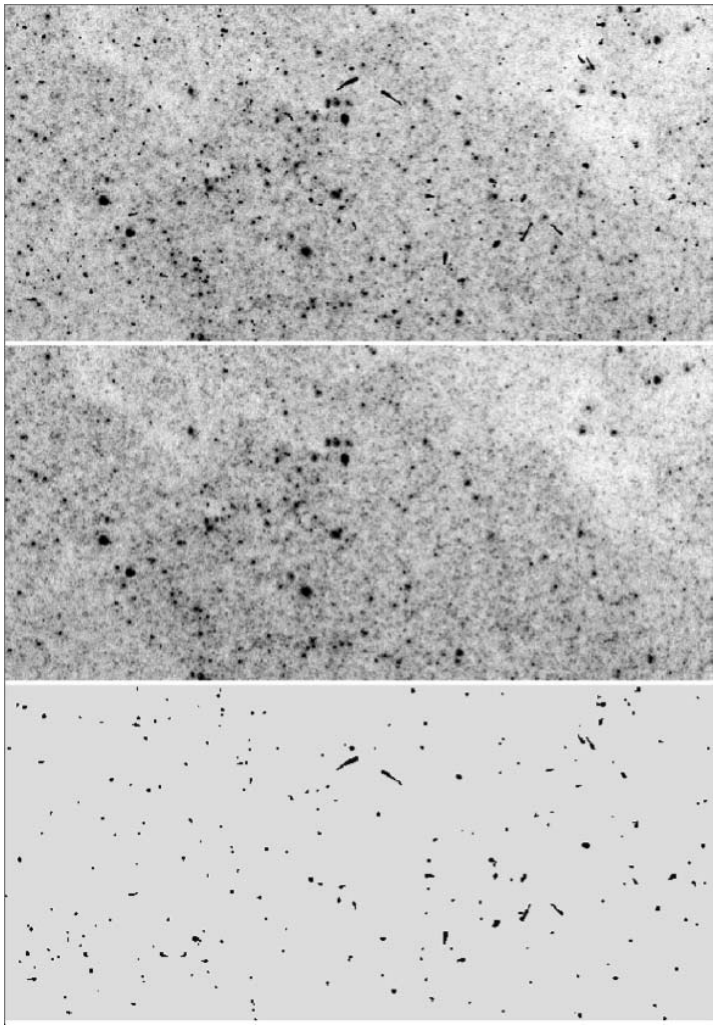
The quality of the cosmic ray masks should be verified by blinking the original `flt.fits` image with both the cosmic ray-cleaned image (suffix `cr_clean.fits`) and the cosmic ray mask (suffix `cr_mask.fits`). When testing various parameter values, the user may find that the cores of stars have been improperly masked, and this indicates that either the ***driz_cr_scale*** parameter should be increased or that the alignment computed by **tweakreg** was not optimal. If not enough cosmic rays were detected, the ***driz_cr_snr*** parameter should be lowered to detect fainter objects. The example below is for a single chip, but this should be repeated for both chips.

```
--> displ f814w_pos1_01.fits[sci,2] 1 zs-zr-z1=-0.2 z2=0.5 ztran=log fill+

--> displ f814w_pos1_01_crclean.fits[sci,2] 2 zs-zr-z1=-0.2 z2=0.5 ztran=log fill+

--> displ f814w_pos1_01_sci2_crmask.fits 3 zs+ zr+ fill+
```

Figure 7.33: Close-up of the Same Region in Three Types of Input Products



The top image is a "zoomed-in" region of a flt.fits image, a cosmic ray-cleaned version of that image (middle), and the corresponding CR-mask image (bottom). Blinking these products is recommended to ensure that the cores of stars are not flagged as cosmic rays.

7.5.4 Align the Header WCS of the Two Filter Images

tweakreg is run on each drizzle-combined filter image to compute the offsets that were previously seen in the **ds9** display:

```
--> tweakreg.TweakReg('f438w_pos1_drz_sci.fits',\
refimage='f814w_pos1_drz_sci.fits',conv_width=3.5,threshold=5,\
shiftfile=True,outshifts='shift_438ref_pos1.txt',nclip=10,\
updatehdr=False)
```

Note that the ***threshold*** parameter was selected to produce a matched catalog with a few thousand (3,900) sources. Before matching, the lists contain 6,900 sources in F438W and 51,100 sources in F814W. When the input image and the reference image have very different signal-to-noise values, it may be useful to create a reference catalog a priori and provide this to ***tweakreg*** via the ***refcat*** parameter. This will help ensure that the two filters (or detectors) have similar numbers of objects for matching. The example below shows the syntax for providing a prepared reference catalog:

```
-> tweakreg.TweakReg('f438w_pos1_drz_sci.fits',\
refimage='f814w_pos1_drz_sci.fits',\
refcat='f814w_pos1_drz_sci_sky_catalog.ref',conv_width=3.5,\
threshold=5,shiftfile=True,outshifts='shift_438ref_pos1.txt',\
nclip=10,updatehdr=False)
```

The contents of the resulting shift file are given below. Notice that the fit produces an offset similar to the one predicted from the earlier visual inspection of the drizzled products in **ds9**.

```
Image              dX       dY       drot       scale     xfit_rms  yfit_rms
f438w_pos1_drz_sci  0.973094  5.995664  359.999049  0.999970  0.105581
0.110062
```

To update the image headers with the new solution, run **tweakreg** with ***updatehdr=True***.

```
--> tweakreg.TweakReg('f438w_pos1_drz_sci.fits',\
refimage='f814w_pos1_drz_sci.fits',conv_width=3.5,threshold=5,\
nclip=10,updatehdr=True,wcsname='TWEAK_814DRZ')
```

IRAF's **hedit** command shows that the keyword `WCSNAME` has now been updated to reflect the new solution:

```
-> hedit f438w*drz_sci.fits wcsnam* .
f438w_pos1_drz_sci.fits,WCSNAMEA = DRZWCS
f438w_pos1_drz_sci.fits,WCSNAMEB = TWEAK_814DRZ
f438w_pos1_drz_sci.fits,WCSNAME = TWEAK_814DRZ
```

The WCS of the F814W and F438W drizzled images are now aligned. No further processing is necessary, as long as the user defines photometric source catalogs in R.A. and Dec. coordinates.

If, however, the user requires the physical pixels of the two frames to align (for example, to subtract a broad-band image from a narrowband image in order to remove the continuum), the images must be drizzled one more time with a pre-defined output frame of reference.

7.5.5 Propagate Improved Solution to Original flt.fits Images with tweakback

After **tweakreg** has been used to align drizzled products[10], the **tweakback** task can be used to propagate the updated WCS back to the original `flt.fits` images. AstroDrizzle may then be used to re-drizzle the `flt.fits` images frames; the drizzle products will be aligned, and can be verified by displaying and blinking them in **ds9**. Note that **tweakreg** only aligns the WCS in the image headers. To ensure that the actual pixels are aligned in the final drizzled product, a output frame of reference must be defined, as described in the next section.

```
--> tweakback.tweakback('f438w_pos1_drz_sci.fits',\
input='f438w_pos1_??.fits',verbose=True)
 Processing f438w_pos1_01.fits[('SCI', 1)]
Updating header for f438w_pos1_01.fits[1]
WCS Keywords
CD_11 CD_12: -1.09949173435E-05 -1.97018332034E-06
CD_21 CD_22: -1.27983613307E-06 1.08908389688E-05
CRVAL : 204.271406112 -29.8720133266
CRPIX : 2048.0 1026.0
NAXIS : 4096 2051
Plate Scale : 0.039851380502520579
ORIENTAT : -10.25406939504851

--> hedit f438w_pos?_??.fits[1] wcsnam* .
f438w_pos1_01.fits[1],WCSNAMEO = OPUS
f438w_pos1_01.fits[1],WCSNAMEA = IDC_v5r1512gi
f438w_pos1_01.fits[1],WCSNAMEB = TWEAK_TILE1
f438w_pos1_01.fits[1],WCSNAMEC = TWEAK_814DRZ
f438w_pos1_01.fits[1],WCSNAME = TWEAK_814DRZ
```

10. In this example, **tweakreg** was used to align two drizzled images, each with a different filter. The same concept can be applied to aligning drizzled images of different detectors and mosaic pointings.

As a sanity check, one can rerun **tweakreg** on the original list to ensure that the intra-visit alignment is still optimal. This step is not necessary, but it is shown for completeness, along with the output from the shift file.

```
--> tweakreg.TweakReg('f438w_pos1_??.fits,updatehdr=False,\
conv_width=3.5,threshold=200,shiftfile=True,\
outshifts='shift_438_pos1_iter2.txt')
```

Image	dX	dY	drot	scale	xfit_rms	yfit_rms
f438w_pos1_01.fits	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
f438w_pos1_02.fits	0.000023	-0.000038	0.000024	1.000000	0.067504	0.066542
f438w_pos1_03.fits	-0.000098	0.000042	0.000015	1.000000	0.067033	0.061176

7.5.6 Drizzle the Images to a Common Reference Frame

This section demonstrates how the drizzled images for the two filters can be aligned in pixel space. Some reasons for doing this include creating color images, subtracting one image from another, or combining them to create an object detection "white light" image (i.e., adding images taken with different filters).

This maybe done in one of two ways:

Option A:

Set the parameter ***final_refimage*** to point to the F814W drizzled product. This will automatically define the size, orientation, scale, and R.A./Dec. of the central pixel. To retain the drizzled product from Section 7.5.5 for comparison, it can be renamed as a "version 1" image prior to reprocessing.

Parameters specifying the final output grid are shown in bold in the examples below.

```
--> imrename f438w_pos1_drz_sci.fits f438w_pos1_drz_sci_v1.fits
--> imrename f438w_pos1_drz_wht.fits f438w_pos1_drz_wht_v1.fits

--> astrodrizzle.AstroDrizzle('f438w_pos1_??.fits,\
output='f438w_pos1',driz_sep_bits='64,32',driz_cr_corr=yes,\
final_bits='64,32',final_wcs=yes,\
final_refimage='f814w_pos1_drz_sci.fits')
```

If ***final_refimage*** is not defined, the F438W drizzled product will not be aligned with the F814W image in *pixel space* because **tweakreg** operates on the images' header WCS, not the physical pixels.

AstroDrizzle automatically computes the image reference frame for each set of input images based on the orientation of the reference image (`f814w_pos1_drz_sci.fits` in the example above) and the distortion solution

of each filter. The product dimensions will be the smallest-sized output image which incorporates the full set of input frames.

Option B:

Instead of defining a ***final_refimage***, set up a pre-defined output grid based on keywords in the original F814W drizzled product. The **hselect** command, shown below, queries the value of the reference position in pixels and in sky coordinates (R.A./Dec.), and the **imheader** command returns the size of the array. The user may wish to allow for additional "padding" in the final drizzled product to account for offsets in pointing or differences in geometric distortion with respect to the reference image. This may be done by specifying the ***final_outnx*** and ***final_outny*** parameters in AstroDrizzle. Note that the output products have been given unique rootnames, `f814w_pos1r` and `f438w_pos1r`, to reflect the fact that they were drizzled to a pre-defined reference grid.

```
-> hselect f814w_pos1_drz_sci.fits \
$I,crpix1,crpix2,crval1,crval2 yes
# output:
f814w_pos1_drz_sci.fits 2462.24 2597.76 204.268999 -29.860656

--> imheader *drz_sci.fits
# output:
f814w_pos1_drz_sci.fits[4924,5195][real]:
f814w_pos2_drz_sci.fits[4581,4885][real]:

--> astrodrizzle.AstroDrizzle('f814w_pos1_??.fits,\
output='f814w_pos1r',driz_sep_bits='64,32',driz_cr_corr=yes,\
final_bits='64,32',final_wcs=yes,final_scale=0.0396,final_rot=0,\
final_outnx=5000,final_outny=5200,final_ra=204.268999,\
final_dec=-29.860656)

--> astrodrizzle.AstroDrizzle('f438w_pos1_??.fits,\
output='f438w_pos1r',driz_sep_bits='64,32',driz_cr_corr=yes,\
final_bits='64,32',final_wcs=yes,final_scale=0.0396,final_rot=0,\
final_outnx=5000,final_outny=5200,final_ra=204.268999,\
final_dec=-29.860656
```

Following the same logic, an ACS/WFC image may be aligned to a WFC3/UVIS image, though one may need to think about how to set the ***final_scale*** parameter since the native ACS/WFC plate scale, at 0.05 arcseconds/pixel, is larger than the WFC3/UVIS plate scale of 0.0396 arcseconds/pixel. If the frames have been dithered, it may be possible to set the ***final_scale*** to a value smaller than the native plate scale. For more information, refer to the examples on optimizing image sampling in Sections 7.2 and 7.4.

The drizzled products are now aligned to the same reference pixel, as shown in the result below. These products can be compared to those from Section 7.5.5 which used a different reference pixel.

```
--> hselect *drz_sci*.fits $I,crpix1,crpix2,crval1,crval2 yes
#
f438w_pos1_drz_sci_v1.fits 2461.82 2597.75 204.269016 -29.860724
f814w_pos1_drz_sci.fits 2462.24 2597.76 204.268999 -29.860656

# Option A
f438w_pos1_drz_sci.fits 2462.24 2597.76 204.268999 -29.860656
f814w_pos1_drz_sci.fits 2462.24 2597.76 204.268999 -29.860656

# Option B
f438w_pos1r_drz_sci.fits 2500.00 2600.00 204.268999 -29.860656
f814w_pos1r_drz_sci.fits 2500.00 2600.00 204.268999 -29.860656
```

In **ds9**, the frames may now be aligned by physical pixels rather than by WCS. Comparing the position of a few stars in each image using **imexamine** will verify the alignment.

```
--> displ f438w_pos1r_drz_sci.fits 1 zs-zr-z1=-0.2 z2=0.5 \
ztran=log fill+
--> displ f814w_pos1r_drz_sci.fits 2 zs-zr-z1=-0.2 z2=0.5 \
ztran=log fill+
```

An optional step: to verify that there are no remaining residuals in the two aligned filter images, **tweakreg** can be used to prove the integrity of the results:

```
--> tweakreg.TweakReg('f438w_pos1r_drz_sci.fits',\
refimage='f814w_pos1r_drz_sci.fits',updatehdr=False,\
conv_width=3.5,threshold=5,shiftfile=True,\
outshifts='shift_438ref_pos1r_iter2.txt',nclip=10)
```

The drizzled science products are shown for comparison in Figure 7.34 and the 4-panel fit residual plot is shown in Figure 7.35. The images, as shown in the shift file below, are now aligned to ~0.1 pixel accuracy,

```
Image                dX         dY         drot      scale     xfit_rms  yfit_rms
f438w_pos1r_drz_sci  0.001162   -0.003424  0.000099  1.000000  0.105570  0.110072
```

Figure 7.34: A Comparison of the F438W and F814W Products

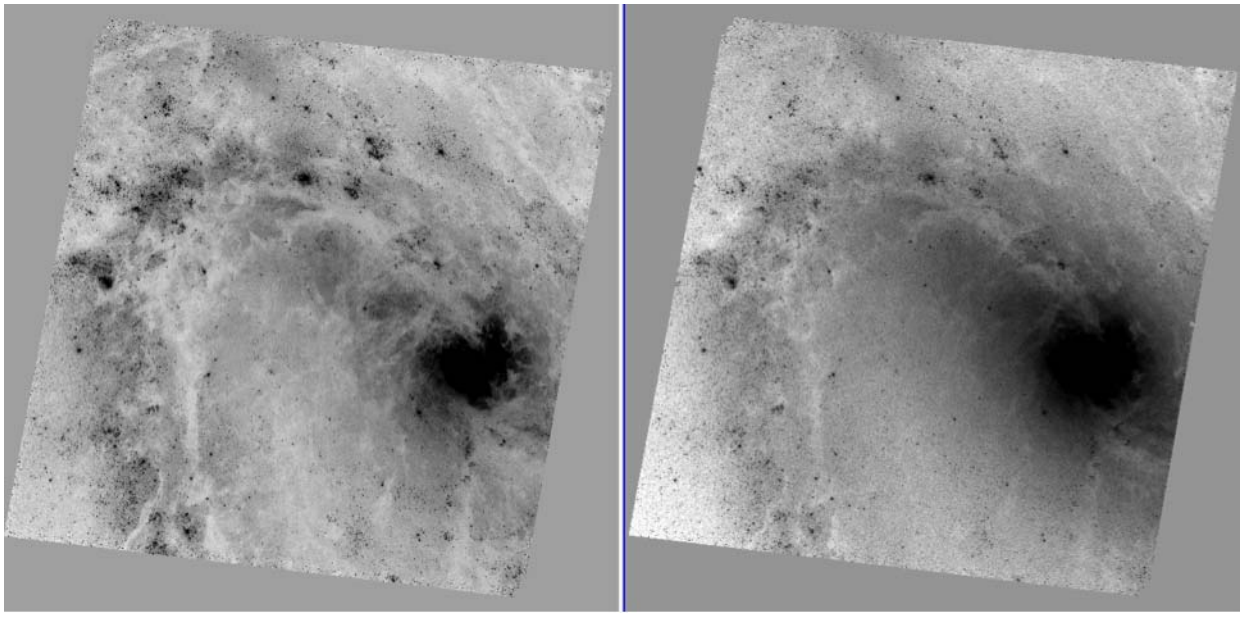
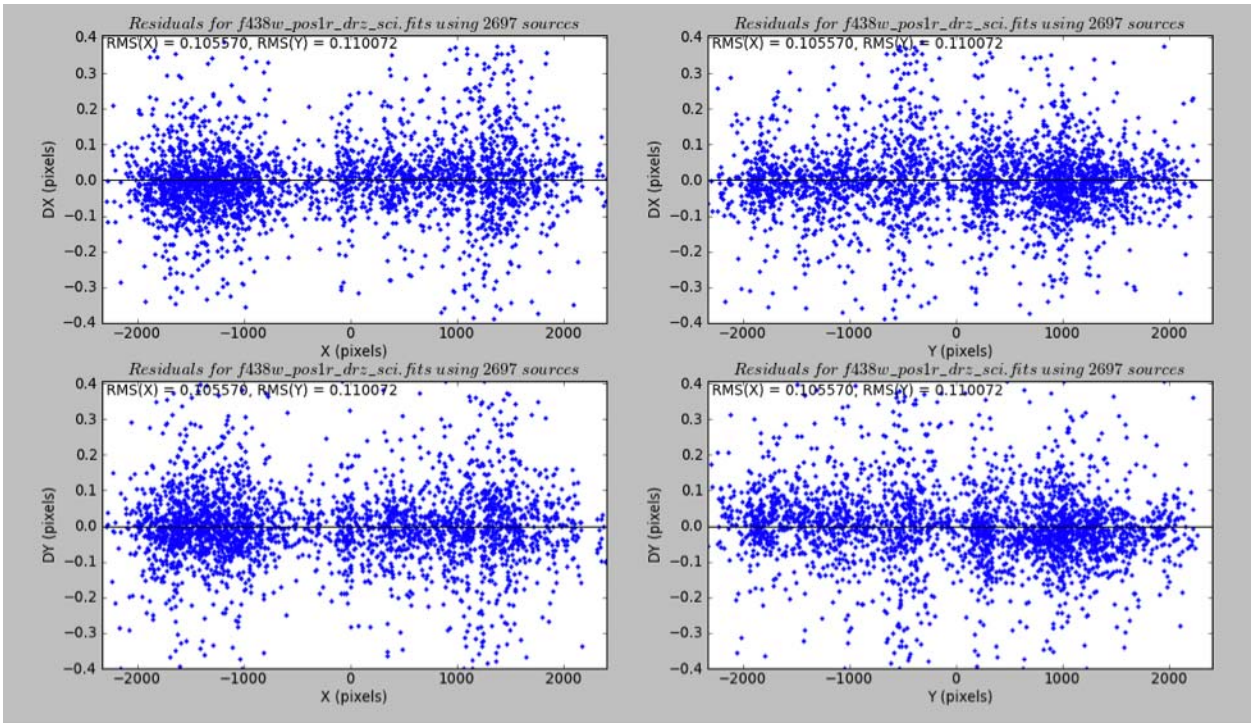


Figure 7.35: Astrometric Residuals for F438W with Respect to F814W



Relative alignment is accurate to ~0.1 pixels.

CHAPTER 8:

Data Quality Checks and Troubleshooting Problems

In this chapter…

8.1 Inspecting the Drizzled Products from MAST

Should the images be reprocessed? Are the pipeline drizzled products adequate for the science goals?

In general, pipeline drizzled products should only be used as "quick-look" products. Reprocessing is highly recommended to achieve the most scientifically accurate data products. Four main areas for improvement include: (1) image alignment, (2) sky subtraction, (3) cosmic ray rejection, and (4) final image resolution.

AstroDrizzle ties together a substantial set of algorithms, each designed to accomplish a different task, and as such has a large parameter set. Pipeline products have been created using a default set of parameters that will work for a wide range of data. These default settings, however, will not produce the optimum science data quality for most programs, and those images will require post-pipeline processing. Additionally, MAST only creates drizzled products for images obtained in a single visit, so additional visits may only be combined together by reprocessing.

While single visit data with small `POS TARG` dithers (like the 4-point dither box) are usually aligned to better than 0.1 pixels, the drizzled products are created using the native detector plate scale, and with a drop size or "pixfrac" of 1.0. In these cases, the

resolution of the drizzled products can be improved by fine-tuning the ***final_scale*** and ***final_pixfrac*** parameters.

Single visit data with large `POS TARG` dithers (for example, to create mosaics), on the other hand, usually have residual offsets of a few tenths of a pixel (and small rotations of a few thousandths of a degree.) When combining data from different visits (via manual reprocessing), tweaks to the image alignment are usually necessary since different sets of guide star pairs may have been used. Offsets of the same target at different rolls are typically ~0.3 to 0.5 arcseconds. For more information, see Appendix B.

Poor alignment can lead to poor cosmic ray rejection, and when astronomical sources are flagged as cosmic rays, photometric accuracy of the final data products will be compromised. Additionally, a poor estimate of the sky background, for example in images where a bright target fills the frame, may also impact the accuracy of cosmic ray rejection, and in turn, the resulting photometry.

Listed below are some problems that users may encounter, and suggestions on how to address these issues during reprocessing. For more discussion, please also refer to Section 8.3.

8.1.1 Examine the Drizzled Science Image

Do the drizzled products look "clean?"

The science extension "*image*`_drz.fits[sci,1]`" from the drizzled MAST data products should be inspected for any obvious anomalies and compared with the calibrated data products, "`image_flt.fits`" which are used as input to the **astrodrizzle** task. For any anomalies which are not addressed in this chapter, please contact the STScI Help Desk (help@stsci.edu).

Are there any irregularities (or discontinuities) in the sky background?

A discontinuity in the sky could be caused by improper sky subtraction near a bright target when a large dither has been obtained as part of an associated data product. Alternately, when the target fills the field of view and there is no "blank" sky to use for determining the sky background, the sky may be over-subtracted. In such instances, it may be necessary to adjust the sky determination parameters during reprocessing, or to instruct **astrodrizzle** to use a keyword containing an independently-determined sky value. For more details, please refer to Section 6.3.1.

Are the PSFs "round" and "narrow," as expected?

Elongated PSFs in drizzled images may indicate that the alignment needs to be improved. A powerful new task called **tweakreg** (part of the **drizzlepac** package) can be used to improve the image alignment for data taken within a single visit or across multiple visits. Alternately, slightly elongated PSFs may be a result of the actual telescope focus at the time of observation. To view the focus history for your observations, users are encouraged to use the HST Focus Model Tool.

More information about the WFC3 PSF may be found in Sections 6.6.1 and 7.6 of the *Wide Field Camera 3 Instrument Handbook for Cycle 20*. Information about the ACS PSF may be found in Section 5.6 of the *Advanced Camera for Surveys Instrument Handbook for Cycle 20*.

Are there unusual patterns or clusters of bright pixels repeated across the image?

These are usually due to hot pixels which are not properly rejected. When the observer has made use of a standard dither pattern (i.e., a 4-point dither box), these artifacts will show up in the same pattern. To remedy this, it may be necessary to reprocess the data, changing the ***driz_sep_bits*** and ***final_bits*** parameter settings. For more details, please refer to Section 6.3.4.

Were the observations dithered?

Archival drizzled products are created using a conservatively-selected set of parameters, which include drizzling to the detector's native plate scale and using a drop size where ***final_pixfrac=1.0***.

When dithering was part of the observation strategy, the resolution of the drizzled products can often be improved by fine-tuning the ***final_scale*** and ***final_pixfrac*** parameters in the final ***astrodrizzle*** step. This is especially true when one of the standard HST pixel dither patterns was chosen to subsample the PSF. For more information, please see Section 6.3.4 and the examples in Sections 7.2 and 7.4 which describe how to improve image sampling for dithered observations.

Were observations obtained in multiple visits?

Tweaks to the image alignment are usually necessary when combining data from different visits (via manual reprocessing). Offsets of the same target at different rolls, and using different guide stars have typically been ~0.3 to 0.5 arcseconds. For more information on relative WCS alignment between visits, see "Updating the WCS Header Information" on page 27.

Using **tweakreg** to improve the image alignment will result in much better cosmic ray rejection. Even data taken in the same visit may often be improved by using the **tweakreg** task to sharpen the alignment since small changes in pointing can occur with guide star reacquisitions from orbit to orbit. In other less common instances, a loss of lock on a guide star(s) may introduce a small drift or roll which will need to be corrected.

Why is there a Moiré pattern in the sky background?

Correlated noise patterns are most often seen when just a few images are being drizzled or a small "pixfrac" has been used. These patterns are seen when the amount of correlated noise varies strongly between pixels. (Pixels with highly correlated noise tend to look less noisy than uncorrelated pixels.) For more details, refer to Section 2.3.

The best way to mitigate this effect is to acquire the observations using a dither pattern, or to use **astrodrizzle** to combine more frames from overlapping observations so that the correlated noise becomes out-of-phase and cancels out in the combined output frame.?

If the data is not dithered and more frames are not available for combination, users can try the *lanczos3* kernel in the final drizzle step, which can help suppress correlated noise. Note that this option does not perform well in the presence of artifacts such as hot pixels and cosmic rays. If too few images are available for combining the frames to reject such artifacts, some "ringing" (a halo of negative pixels) may be seen around the sharp edges of these artifacts.

8.1.2 Verify the Image Header Sky Keyword

Does the MDRIZSKY header keyword seem correct when compared to an independent estimate using other software tools?

During pipeline processing, the sky background is estimated using a default set of parameters and is written to the image header in the science extensions [1] and [4] of the `flt.fits` image in the header keyword `MDRIZSKY`.

Many astronomical fields of view cover portions of the sky devoid of large objects, and as a result, the default sky subtraction parameters are sufficient. For observations of targets that fill the field of view, however, sky background may be overestimated. An inaccurate sky subtraction may compromise the accuracy of the cosmic ray rejection, and this in turn may impact the resulting photometry.

Two important parameters to consider when reprocessing are the lower and upper values for pixels that will be used to estimate the sky value. These should be set large enough to include the majority of pixels in the sky (substantially more than the FWHM of the sky distribution) but not so large as to include substantial signal from objects or cosmic rays.

If the user instead wishes to compute the sky using an alternate method outside of **astrodrizzle**, the ***skyuser*** parameter can be set to point to a keyword in the image header which gives a user-defined sky value. **astrodrizzle** will then assume that this sky value has already been removed from the `flt.fits` images prior to processing.

8.1.3 Examine the Data Quality Array

Did the pipeline flag excessive numbers of pixels as 4096 (cosmic ray DQ flag)?

Users are encouraged to blink the science array of the calibrated `flt.fits` images with its corresponding data quality array. Cosmic rays flagged during pipeline processing are assigned a flag value of 4096, and the accuracy of these flags should be inspected. For more details, refer to the example in Section 7.1. If primarily astronomical sources are being flagged, this could indicate a slight misalignment of the images. The WCS information can be corrected to account for these small offsets by processing the images with **tweakreg**.

If a regular "pattern" of cosmic ray flags is apparent over the entire image, this could be caused by improper sky subtraction. Instead of cosmic rays, the algorithm may be picking up noise in the sky background. For an example of this effect, see the example in Section 7.5.

Reprocessing will correct the problem of excessive or imperfect cosmic ray flagging when the ***resetbits*** parameter is set to ***4096*** (default). This parameter will reset those flags in the DQ array so that cosmic ray flags can be recomputed by the user during reprocessing.

8.1.4 Examine the Drizzled Weight Image

Is the mean value of the weight image approximately equal to the total exposure time?

The weight image in pipeline-drizzled products (i.e., *image*`_drz.fits[wht,1]`) are weighted by the total exposure time of each pixel (***final_wht_type=EXP***), and the weight extension of the drizzled image can be considered an effective exposure time map. Regions of the image with drastically different weight values may indicate that only a fraction of the original images contributed to the final product. In this case, reprocessing is highly recommended. Other weighting methods (`ERR`, `IVM`) may be preferable for reprocessed data, depending on the science goals. For more information, see Section 4.2.12.

Does there appear to be an imprint of the target in the weight image?

In general, the weight extension of the drizzled product should contain a random distribution of pixels with lower weight, reflecting detector artifacts and cosmic rays which were excluded. When the sources themselves are flagged as cosmic rays, this often indicates a problem with the cosmic ray rejection, due either to an improper sky subtraction or to misalignment between input frames.

WFC3/IR images are the exception to this rule. IR data has an additional 5th extension (`TIME`) that contains the effective integration time associated with each corresponding science image pixel value. For calibrated datasets, the `TIME` array contains the combined exposure time of the valid readouts that were used to compute the final science image pixel value, after rejection of cosmic rays and saturated pixels from the intermediate data. When drizzled images are weighted by the total exposure time, the weight image will reflect the reduced exposure time in pixels which saturated in one or more samples. These often fall in the cores of bright stars and galaxies, resulting in an imprint of the target in the final weight image. This effect does not mean that reprocessing is required; it is simply a feature of WFC3/IR data, showing pixels where the detector was saturated.

8.2 Verifying tweakreg Solutions

How can the quality of **tweakreg** image alignment results be assessed?

In general, the first step in manual reprocessing is running **tweakreg** to improve the relative alignment of the WCS in the image headers. By default, this task runs in an interactive mode by producing a number of plots and figures for inspection. Users are strongly encouraged to verify the quality of the solutions as described below.

8.2.1 Examine the Fit Residuals

While running **tweakreg**, the results of the fit (residual shift, rotation, and scale), and the fit RMS are printed to STDOUT (computer monitor) or to an optional "shift file." These solutions should be inspected for accuracy before proceeding to drizzling the images.

Does the solution make sense?

Observations obtained within a single visit should, in general, require no residual rotation (less than 0.001 degrees) and a very small shift (less than a few tenths of a pixel.) The exception to this rule is when very large `POS TARG`s are specified to offset the telescope from the commanded R.A. & Dec. In these cases, small residual rotations on the order of a few thousands of a degree are expected. On the other hand, when images are obtained in separate visits, the target may have been reacquired with a different set of guide stars, and this will limit the accuracy of the relative alignment to about 0.3 to 0.5 arcseconds. For more information on pointing repeatability, see Appendix B.

To estimate the residual offsets before running **tweakreg**, users may display the images in **ds9**, align by WCS, and then blink the three image frames. For more details on inspecting the WCS alignment, refer to the example Section 7.3.

By default, **tweakreg** will solve for a residual shift, rotation, and scale between images (***fitgeometry='rscale'***). When the solution suggests a very small residual rotation, users are encouraged to rerun the task with ***fitgeometry*** set to ***shift***. If the RMS of the fit is roughly the same, then the residual rotation is likely not real and the shift-only solution is recommended. (In other words, the simplest solution is usually the best one, and the rotation is likely not significant.) Additionally, if **tweakreg** computes a residual shift that is smaller than the fit RMS, the results are not significant (compared to the errors), and the user may proceed to drizzling without updating the image headers.

For images with long exposure times and low signal-to-noise sources (for example, narrow-band images), **tweakreg** may detect more cosmic rays than actual sources. When this happens, the fit may give a result that makes no sense. Since **imagefindpars** (used by **tweakreg**) is not able to select sources based on their "sharpness," it may be necessary for the user to import his or her own custom cosmic ray-free catalogs generated using tasks like **daofind** or **sextractor**. These catalogs may be given to **tweakreg** as input via the ***catfile*** parameter. For more information on the format of this input table, see Table 4.21 and the example in Section 7.5.

Is the fit RMS small?

For a good solution with a large number of sources (a few hundred), the RMS of the fit is generally better than ~0.1 pixels (and as good as 0.03 pixels). When fewer astronomical sources (stars, compact objects like HII regions, or small galaxies) are available for centering, the RMS of the fit may be slightly larger. Users are encouraged to inspect the three **tweakreg** plots described below to verify the quality of the solutions.

8.2.2 Examine the Astrometric Residuals

Is there any remaining slope in residuals plot?

The four-panel residuals plot gives the *x* and *y* components of the residuals vs. *x*- and *y*-axis position, and this plot is useful for spotting subtle and/or large scale alignment issues. A good fit gives flat residuals with typical RMS values less than 0.1 pixels. A residuals slope in the astrometric residuals plot may indicate that the true rotation or scale between images has not been adequately fit. Users can try changing the parameter ***fitgeometry*** from ***rscale*** to ***shift*** to see if this improves the solution (for example, a solution with a smaller fit RMS or a residual plot with a less obvious slope).

For ACS data, a small time-dependent skew in the geometric distortion has been corrected via improved distortion solutions, as discussed in Appendix A.5.1. While the effect has largely been corrected, a small residual skew (+/- 0.05 pixels) may still remain and this may show up as a slope in the astrometric residual plots. An example of this skew is apparent in Figure 7.22, for the example in Section 7.3. This is a known limitation in the distortion calibration and does not indicate a problem with **tweakreg**.

Users may also experiment with the clipping parameters ***nclip*** and ***sigma*** to see if removing outliers allows **tweakreg** to compute a more accurate solution (with a fit RMS ~0.1 pixels). Determining the optimal level of clipping is a judgment call, and users are advised to use care to not clip the residuals too aggressively. While more aggressive clipping may give a better fit RMS, an examination of the plots may show that the solution has been artificially over-constrained, showing a hard edge in the distribution and no outliers. A slightly larger fit RMS is better than a small fit RMS and an unintentionally biased solution.

8.2.3 Examine the Vector Plot

Are there any obvious "flow" patterns in the vectors?

The vector plot is an alternate way of viewing the four-panel astrometric residuals plot. In the vector plot, source magnitude and direction of residuals are plotted as a function of location. This is useful for spotting localized systematic deviations in image alignment. A good image alignment produces a vector plot that appears as small randomly-oriented vectors with no clear organized flows or structures. Any obvious "flow" patterns may indicate that the true rotation or scale between images has not been accurately fit.

Are sources detected over the entire field of view (for sparse fields or star clusters)? If not, are they detected in the regions expected (for example, a small galaxy which does not fill the image)?

To verify whether the fit is being based on "real" objects and not artifacts, users are encouraged to display the calibrated (`flt.fits`) images and overplot the matched source lists. For details on how to do this, refer to the example in Section 7.5. If sources are only matched in one portion of the detector, this may indicate that the sigma clipping parameters ***nclip*** and ***sigma*** were too aggressive. Alternately, if sources

were only detected in the sky surrounding the target (and not in the target itself), it may be necessary to adjust the **imagefindpars** parameters ***threshold*** and ***skysigma***.

Are there "clumps" of sources detected near very bright stars?

Saturated (or very bright stars) may cause **imagefindpars** to detect a large number of sources in the halo of the star, in the diffraction spikes, or in bleeding pixels. To avoid biasing the **tweakreg** solution, users may define one or more "exclusion" regions such that sources within that area are not included in the fit. For more details on defining Exclusion Catalogs, see Section 4.4.2.

8.2.4 Examine the 2-Dimensional Histogram

Is the peak clearly defined?

tweakreg will display a two-dimensional histogram with an initial guess of the offsets between each image. As shown in examples in Sections 7.1 and 7.3, a good fit will produce a 2D histogram with a single bright peak. When this fails, the user should examine the four-panel residual plot. If the fit does not appear to include the majority of sources (clustered tightly around zero residual), then it is possible that the fit has not been based on the position of astronomical objects, but instead on cosmic rays or detector artifacts, like hot pixels. To verify this, users are encouraged to display the calibrated (`flt.fits`) images and overplot the matched source lists to ensure that the fit is based on "real" objects and not artifacts. For details on over-plotting catalogs generated by **tweakreg**, refer to the example in Section 7.5.

Alternately, a poorly-defined peak may suggest that the user needs to be more (or less aggressive) in clipping sources using the parameters ***nclip*** and ***sigma***.

Is the fit crosshair centered on the peak?

When the fit crosshair is offset from the brightest pixel in the two-dimensional histogram, this may indicate a need to increase the search radius of the fit. By default, ***searchrad=1.0*** arcsecond, but when images are obtained in different visits, especially for older data where the guide star catalogs have larger positional uncertainty, offsets of several arcseconds may be found.

8.2.5 Verify the Number of Matches Used to Compute the Solution

Does tweakreg crash with the message "not enough matches found?"

When **tweakreg** finds less than 15 objects in the final matched source catalog, it will report an error. When this happens, the user is advised to either decrease the ***minobj*** parameter, to increase the ***searchrad*** parameter, or to modify the clipping parameters ***nclip*** and ***sigma***.

Users are also advised to inspect the number of objects in the initial (unmatched) source catalogs (with file naming convention `*sci?_xy_catalog.coo?`), one of several intermediate processing files created by **tweakreg**. If the catalogs do not contain a sufficient number of sources (ideally a few hundred), the **imagefindpars** parameter ***threshold*** may be decreased to look for fainter sources. Since

imagefindpars is not able to select for "sharpness," many of these sources will be cosmic rays or detector artifacts, so creating catalogs that are slightly larger than necessary for the final match will ensure that a sufficient number of true sources remain for computing the residual offsets between images. For details on over-plotting the original and matched source lists, refer to the example in Section 7.5.

8.3 Inspecting Drizzled Products after Reprocessing

Are the reprocessing task parameters optimal?

The same set of checks discussed for Archival data should be performed for reprocessed data. For more details, refer to Section 8.1. These checks include examining the quality of the sky subtraction, inspecting the cosmic ray masks, and looking for unusual patterns of artifacts or correlated noise in the science array. The PSF should be inspected (using a task like IRAF's **imexamine**) over the entire field of view. If dithering was part of the observation strategy, the resolution of the drizzled products can usually be improved by experimenting with the final drizzle parameters, especially for HST detectors which are significantly undersampled.

The weight images should be carefully examined to get an idea of how many input pixels contributed to each output pixel. When the weight image has a significant number of "holes" where no valid input pixel was available, the data quality array of the input frames should be inspected and the value for the ***final_bits*** parameter adjusted. Alternately, this may indicate that the ***final_pixfrac*** parameter was too small for the dataset, based on the number of images and the dithering pattern used. (More information about this will be covered in Sections 8.3.1 and 8.3.2, below.)

A detailed discussion of running **astrodrizzle** with HST data is presented in Chapter 7. Instructions for inspecting the intermediate data products and for optimizing various parameters are addressed in these examples, as well as suggestions for troubleshooting.

8.3.1 Examine the Drizzled Science Image

Drizzled products from MAST are single multi-extension FITS (MEF) files with the science image, the weight image, and the context image in extensions one, two, and three, respectively. During reprocessing, the parameter ***build*** is set to ***no*** by default so the science, weight, and context images are written to separate output files. The choice of this parameter is purely a matter of convenience; reprocessed images can be generated using ***build=yes***, if that option is preferred. (Separate data products may be more convenient for users working with already-large mosaics, where smaller files may be preferable for electronically sharing with collaborators, or when users have disk space limitations and need to move various sets of data products to alternate locations.)

Compare the science array of the drizzled pipeline product (i.e., *image_drz*.fits[sci,1]) with the science array derived during reprocessing (i.e., *image_drz_sci*.fits).

- Do the drizzled products look "clean?"
- Are there any irregularities (or discontinuities) in the sky background?
- Are the PSFs "bound" and "narrow", as expected?
- Are there unusual patterns or clusters of bright pixels repeated across the image?
- Does the MDRIZSKY header keyword seem correct?
- Did **astrodrizzle** flag excessive numbers of pixels as cosmic rays?

The questions above are addressed in Section 8.1. Additionally,

- Is there a correlated noise pattern in the sky background that resembles a "screen door" pattern? This type of correlated noise can be caused in two ways: by shrinking the "pixfrac" too small, or by failing to subtract the sky background. For details on each of these effects, please refer to the example in Section 7.2.

Maintaining a larger ***final_pixfrac*** ensures overlap between pixels and less correlated noise in the drizzled science array. When ***final_pixfrac*** has been shrunk too far, a "beating pattern" can be seen in the sky. While this pattern may look alarming to the eye, it does not significantly impact the photometric integrity of the drizzled products. In general, ***final_pixfrac*** values in the range 0.7 to 0.9 are usually optimal when the observations have been dithered (see Section 8.3.2 for details.). If a gain in resolution is not important for the program's science goals, then ***final_pixfrac=1*** will suffice.

As a general guideline, the sky subtraction step should be turned on. This step is necessary for optimal flagging of cosmic rays. Additionally, failure to remove the sky will lead to correlated noise in the drizzled images. (The size of this effect depends primarily on the variation in sky levels from one exposure to the next.) For details, please refer to the example Section 7.2. While some external software packages (such as **daophot**) may expect the sky level to be present, it should be removed for drizzle processing to avoid correlated noise and then (optionally) added back later, if so desired.

8.3.2 Examine the Drizzled Weight Image

- Is the mean value of the weight image roughly equal to the total exposure time?
- Does there appear to be an imprint of the target in the weight image?

The questions above are addressed in Section 8.1 Additionally,

- Is the RMS of the weight image (near the target of interest) less than 20% of the mean (or mode)?

As a rule of thumb, statistics performed on the drizzled weight image in the region of interest should yield an RMS value (standard deviation) that is less than 20% of the median value. This threshold is a balance between the benefits of improving the image resolution at the expense of increasing noise in the background. The ***final_pixfrac*** value should be small enough to avoid degrading the final drizzle-combined image, but large enough that when all images are "dropped" onto the final frame, coverage of the output frame is fairly uniform. In general, ***final_pixfrac*** should be slightly larger than the final output scale to allow some "spillover" to adjacent pixels. This will help avoid "holes" in the final product when a given pixel has been flagged as bad in several frames.

Are there "holes" in the final weight image?

"Holes" in the weight image, regions with no valid input pixels, may indicate that the user should rethink which `flt.fits` DQ flags should be treated as good pixels. Because **astrodrizzle** was designed for reprocessing with multiple instruments, the default value for the "bits" parameter is set to "0" in ***driz_sep_bits*** and ***final_bits***. This is generally an over-aggressive approach if only a few input images are being combined. The "bits" parameters indicates which suspect pixels to keep, and the user can decide which strategy is best based on the number of input frames and the dither pattern used. Ultimately, one wants to avoid having too many pixels with no good input pixel in the stack. For more information on selecting the appropriate DQ bits values, refer to Section 6.3.3.

On the other hand, "holes" may indicate that the user has chosen too aggressive a value for ***final_pixfrac***. For routine observations containing several dithered images, the "pixfrac," or "drop" size, should be between 0.5 and 1.0, depending on the number of input images and the dither pattern. In general, values in the range 0.7 to 1.0 are optimal, but the user should experiment to see what is best for the combination of data in hand and the desired science to be obtained from it. In some cases, pushing the envelope a bit further may yield more beneficial results. In rare cases such as the HUDF, an extremely large number of images were very well-dithered in subpixel space, and this allowed the use of a point kernel (***final_pixfrac=0***), but this is an extremely rare case. Most observers will have far fewer images than this, however, and a more routine and conservative use of "pixfrac" and "scale" is usually in order. For more information, see Section 6.3.3, as well as the examples in Sections 7.2 and 7.4.

APPENDIX A:

Plate Scales and Polynomial Distortions

In this appendix . . .

A.1 Introduction

Apart from uncertainties in telescope pointing, astrometric accuracy is also affected by varying plate scale across detectors as a result of geometric distortion produced by the telescope and instrument optics.

For instance, distortion across the WF chips in WFPC2 amounts to ~2% at the chip corners. If the telescope was commanded to move exactly 2.5", that shift would correspond to a shift of 25 pixels at the detector center where the plate scale is 0.1 arcseconds/pixel. At the corners, however, where the plate scale is ~2% smaller, 0.098 arcseconds/pixel, that shift would have spanned over 25.5 pixels. In other words, objects near the edge of the chip would be subsampled by half a pixel, while those near the center would not be subsampled at all. Therefore, for any given shift in units of pixels, there is a continuously varying amount of subsampling across the image. This illustrates how detector distortion leads to non-uniform pixel subsampling when dithering.

The size of the additional offset introduced by geometric distortion towards the edges of the chip scales linearly with the size of the commanded telescope offset. For example, for WFPC2/WFC, a dither of five pixels at the center (instead of 25 pixels)

would produce a much less severe error of only 0.1 pixels in additional shift at the edge of the chip. As such, the sampling would remain nearly constant across the detector. *This is a reason for why large dithers are discouraged; although the Drizzle software is capable of dealing with changing pixel shifts and different degrees of subsampling across the image, the scientific interpretation of the data could still be impacted because not all the objects are equally subsampled.* Image analysis would be greatly simplified if small dithers were used, thereby keeping such sampling differences relatively small across the chip (e.g., < 0.1 pixel). However, for the ACS, the two chips that compose the WFC are separated by a gap of about 2.5 arcseconds (~50 WFC pixels). Therefore, many ACS dither strategies involve the use of offsets sufficiently large to allow the detectors to cover this gap; the consequence is a varying degrees of subsampling across the detector.

A.2 Detector Plate Scales

HST instruments have multiple folds in the light path to direct light to selected optical elements, and to the detector(s), in a way that achieves the desired focal ratios within the confines of the optical bench envelope, while minimizing throughput loss and image degradation.

In some instruments, the focal surfaces are far from being normal to the principal rays. This results in an image of the sky that's distorted in a large, but predictable manner, depending on the tilt of the detector plane and curvature of the folding optics. To first order, the tilt causes a square pixel on the detector to be projected as a rectangle (for rotation of the detector about one of its axes), as a rhombus (for rotation of a square detector about one of its diagonals), or more generally as a parallelogram. The curvature of the folding optics introduces higher order terms in the distortion, causing the projected dimensions of a pixel to vary somewhat with location on the detector.

The current best-determined values of the projected pixel dimensions are stored in the Science Instrument Aperture File (SIAF), specifically defined for a single pixel at specified reference locations on the detector. The geometric distortion model appropriate to a given detector then allows the calculation of projected pixel dimensions at other locations on the detector.

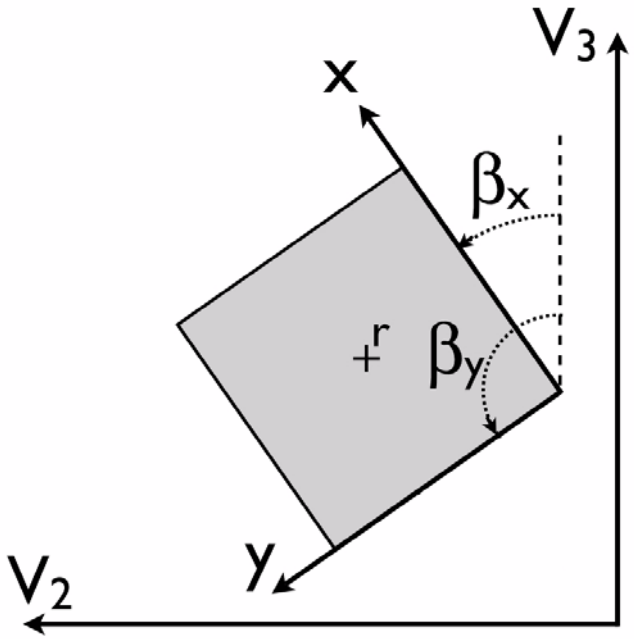
The Instrument Aperture tables contain parameters relating the instrument coordinates, pixels, or deflection offsets to the telescope coordinate frame. This frame, designated *V2, V3*, represents the projected position on the sky. If the telescope roll is zero, then *V3* points North, and *V2* points East. The reason for the unconventional axis order is that *V2* and *V3* are part of a three-dimensional coordinate system with *V1* being approximately along the optical axis. The *x,y* coordinate system relates to each instrument aperture and the angles β_x and β_y are measured counter-clockwise from the *V3* axis, to the *x* and *y* axes. A reference point is chosen near the center of the aperture, which has the coordinates (x_r, y_r) in the instrument frame, and $V2_r$, $V3_r$ in the telescope frame. The reference point is the fiducial in the focal plane on which, HST

will endeavor to position the target specified in the proposal. The scales s_x and s_y, in x and y directions, are not necessarily equal, but are each tabulated. For most HST imaging instruments, the x and y axes are flipped with respect to the *V2* and *V3* axes ($\beta_y > \beta_x$), as shown in Figure A.1, such that their projection onto the *V2, V3* frame may not be orthogonal. This can be expressed by the β_x and β_y angles, such that the transformation formula remain valid. The transformation from any point *(x, y)*, to *V2, V3,* is determined to first order by:

$$V2 = V2_r + s_x \sin\beta_x (x - x_r) + s_y \sin\beta_y (y - y_r)$$

$$V3 = V3_r + s_x \sin\beta_x (x - x_r) + s_y \sin\beta_y (y - y_r)$$

Figure A.1: Visual Representation of the Frame of Reference for Angles in the SIAF Instrument File ($\beta_y > \beta_x$) for ACS, NICMOS, STIS, WFC3, WFPC, and WFPC2



The x and y dimensions of the reference pixels in the *V*3 coordinate system for each of the commonly used imaging detectors are tabulated in Table A.1. Also listed in the table are the angles of the x and y axes for each detector, corresponding to the CCD rows and columns, measured counter-clockwise from the *+V3* axis of the spacecraft, (β_x and β_y). These dimensions are measured for the reference pixel. The orientation and scale of the pixel axes may change somewhat across the detector as a result of geometric distortion. The last column in the table lists the approximate, maximum change in scale from the center pixel to any other position on the detector (the largest changes are generally seen between the centers of the chips and the corners). While the values in this table are intended to be current, the latest positional information can

always be obtained directly through the Observatory Support Group's webpages on apertures and pointing.

All of the numbers detailed in the table below are a snapshot of the current SIAF information as of the printing of this handbook, or values from the latest calibrations reported in Instrument Science Reports. The SIAF file itself is an internal table, which is used in the operational science database at STScI. Information that users would be the most interested in is available from the SIAF Aperture file. If this information is insufficient to meet your needs, please contact the Space Telescope Science Institute help desk.

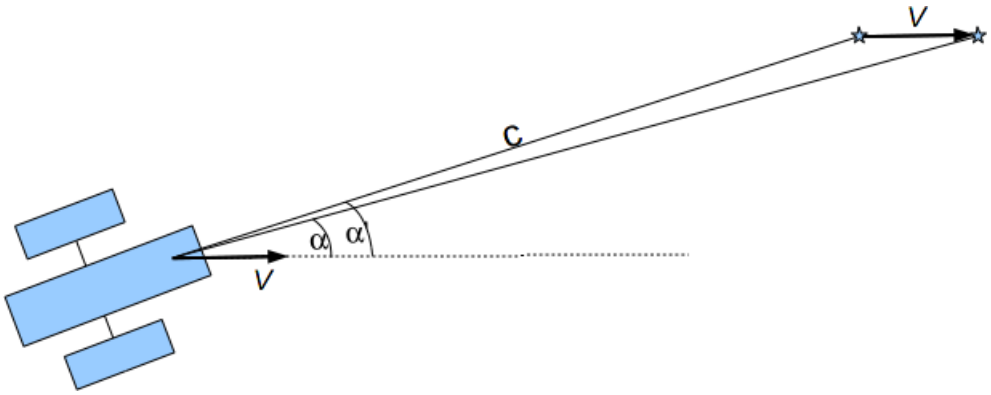
Table A.1: Pixel Scales of the Primary HST Instruments

Instr.	Aperture (pix/side)	Field of View	x-scale ("/pix)	y-scale ("/pix)	β_x	β_y	Detector Tilt (deg)	Distortion Center- corner
WFPC2	PC1 (800^2)	36"	0.045528	0.045507	134.908	224.908	-	1%
	WF2 (800^2)	80"	0.099500	0.099596	224.388	314.388	-	2%
	WF3 (800^2)	80"	0.099573	0.099480	314.698	44.698	-	2%
	WF4 (800^2)	80"	0.099539	0.099635	45.258	135.258	-	2%
ACS	WFC (4096^2)	200"	0.0496	0.0495	92.06	177.53	22	8%
	HRC (1024^2)	29"	0.0284	0.0248	-84.12	0.08	25	11%
	SBC (1024^2)	30"	0.0337	0.0300	-84.56	-0.11	25	11%
NICMOS	NIC1 (256^2)	11"	0.043200	0.043026	225.312	315.312	-	0.7%
	NIC2 (256^2)	19"	0.076002	0.075320	224.507	314.507	-	0.2%
	NIC3 (256^2)	52"	0.203758	0.203012	224.851	314.851	-	0.6%
STIS	CCD (1024^2)	52"	0.050763	0.050777	45.056	135.056	-	0.3%
	NUV (1024^2)	25"	0.024750	0.024750	45	135	-	1%
	FUV (1024^2)	25"	0.024742	0.024742	45	135	-	0.7%
WFC3	UVIS (4096^2)	159"	0.0396	0.0393	-41.121	44.957	21	7.0%
	IR (1014^2)	136"	0.1354	0.1209	-44.8161	45.0099	24	8.0%
COS	LFPSA	2.5" diam	0.0226	0.0943	135	45	-	-
	LNPSA	2.5" diam.	0.0258	0.0258	135	45	-	-
	LFBOA	2.5" diam.	0.0226	0.0943	135	45	-	-
	LNBOA	2.5" diam.	0.0258	0.0258	135	45	-	-

A.3 Effects of Velocity Aberration

Plate scales can vary as a result of the telescope's orbit through velocity aberration. The Hubble Space Telescope has an orbital speed about the Earth of about 7 km/sec., and the Earth's orbital velocity about the Sun is approximately 30 km/sec.

Figure A.2: Velocity Aberration Angular Change



The net velocity causes stellar image displacements of some tens of arcseconds. In Figure A.1, α represents the angle between the telescope direction of motion relative to the Sun and the direction of a star in barycentric coordinates. The term α' is the angle measured towards the instantaneous apparent direction and is given by

$$\tan\alpha' = \frac{\sqrt{1-\beta^2}\sin\alpha}{\cos\alpha+\beta}$$

Differentiating this expression gives us the change in scale due to velocity aberration:

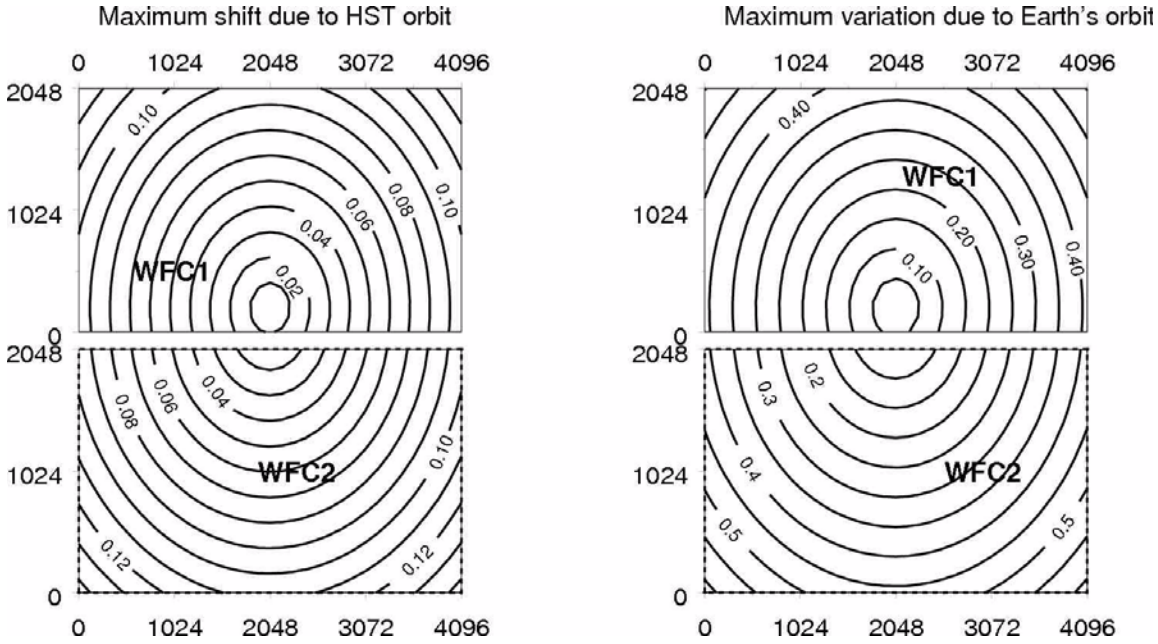
$$\frac{\delta\alpha'}{\delta\alpha} = \frac{\sqrt{1-\beta^2}}{1+\beta\cos\alpha}$$

This scale change turns out to be isotropic over the field of view, allowing it to be corrected for through updates to the CD matrix. The plate scale increases by the reciprocal of $\frac{\delta\alpha'}{\delta\alpha}$. The magnitude of this effect is of order 1 part in 10^4 and can vary by 5 parts in 10^5 within an orbit. More details on this calibration for ACS can be found in the 2002 HST Calibration Workshop proceedings; specifically in the paper "The Effect of Velocity Aberration on ACS Image Processing".

A.4 Velocity Aberration for ACS

The effects of velocity aberration can cause a difference of order one pixel across the diagonal of the ACS Wide Field Camera (ACS/WFC) with its 4096 x 4096 pixel field of view. The isotropic nature of these changes can be seen in Figure A.3.

Figure A.3: Scale Changes for ACS/WFC due to Velocity Aberration



A.5 Detector Distortion Models

Detector distortion models can generally be characterized in terms of the dependence of the true location *(X, Y)* of an object, (e.g., in arcseconds), as a function of the measured pixel position *(x, y)* of the object on the detector relative to some reference pixel (x_0, y_0), usually as a polynomial of order *m* in *x*, in *y*, and including all their cross-term

$$X = \sum_{i=0}^{m} \sum_{j=0}^{i} a_{ij} (x - x_0)^j (y - y_0)^{i-j}$$

$$Y = \sum_{i=0}^{m} \sum_{j=0}^{i} b_{ij} (x - x_0)^j (y - y_0)^{i-j}$$

Explicitly, these distortion polynomials expand out to:

$$X = a_{00} + a_{10}y + a_{11}x + a_{20}y^2 + a_{21}xy + a_{22}x^2 + a_{30}y^3 + a_{31}xy^2 + a_{32}x^2y + a_{33}x^3 + \ldots$$

$$Y = b_{00} + b_{10}y + b_{11}x + b_{20}y^2 + b_{21}xy + b_{22}x^2 + b_{30}y^3 + b_{31}xy^2 + b_{32}x^2y + b_{33}x^3 + \ldots$$

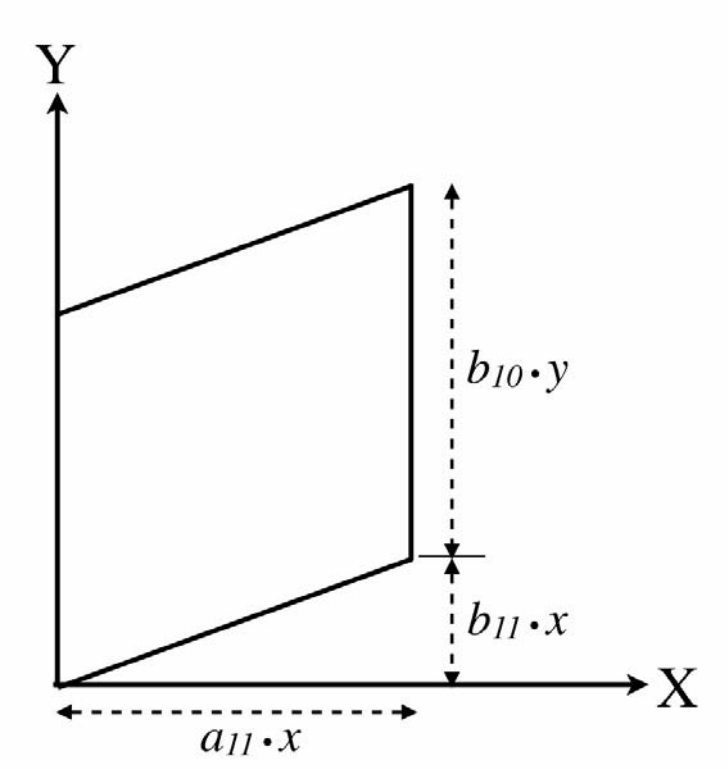
The distortion coefficients characterizing each instrument are stored in the form of tables called `IDCTAB` reference tables that are maintained by STScI, and are incorporated into the IRAF/STSDAS DrizzlePac software. The linear component of the distortion generally
reduces to:

$$X = a_{11}x$$

$$Y = b_{10}y + b_{11}x$$

a_{00} and b_{00} are zero since the two coordinate systems have the same origin. a_{10} is zero when the *Y* axis is defined to lie along the *y* axis of the detector, which is generally the case. If the projected *x* and *y* axes are not perpendicular (b_{11} is non-zero), a displacement in *x* on the detector has a *X* component and a *Y* component, while a displacement in *y* on the detector has only a *Y* component, as shown in Figure A.4:

Figure A.4: Projection of *x, y* Detector Coordinates onto the *X, Y* Frame Using First Order Distortion Terms

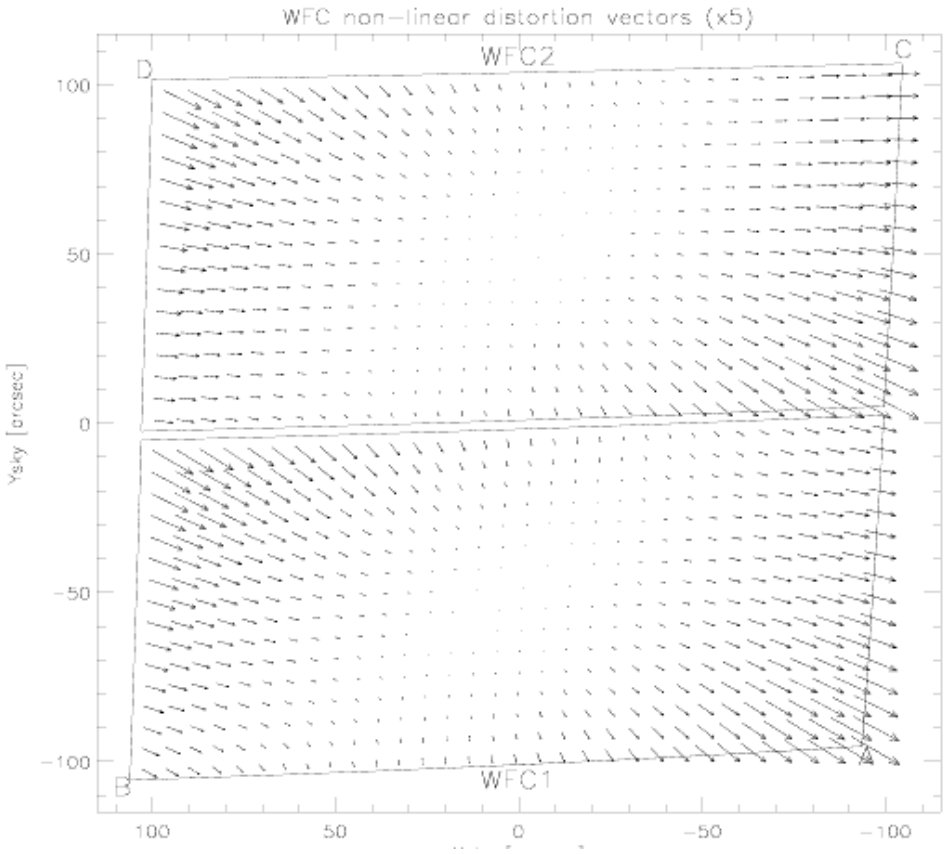


A.5.1 ACS

The ACS/WFC is significantly more distorted than the WFPC2, STIS or NICMOS cameras. The magnitude of the distortion is predominantly a result of the large format of the detector, together with its off-axis location in the HST focal plane. The distortion of ACS/WFC is a combination of an 8% elongation along the detector diagonal, resulting from the detector inclination with respect to the optical axis, together with an increasing radial distortion away from the center of the WFC. The ACS distortion, characterized in terms of polynomials (Hack & Cox 2000), has been described in the *ACS Data Handbook*.

In particular, the WFC distortion is illustrated in this figure, a vector displacement diagram which shows the contribution of the non-linear part of a quartic fit to the data. The vectors represent the degree of distortion to be expected in the WFC beyond the directional dependence of the plate scale. For display, the vectors are magnified by a factor of five compared to the scale of the *x* and *y* axes. The largest displacement indicated at the top left corner of the figure is ~82 pixels or about four arcseconds.

Figure A.5: Non-linear Component of the ACS Distortion for the WFC Detector using a F475W Quadratic Fit



Note that this figure is rotated 180 degrees with respect to the default orientation of the drizzled product, where WFC2 would be the lower half of the detector

This distortion model has been stored in `IDCTAB` reference files for use by the AstroDrizzle software to correct the distortion in ACS observations and allow for the combination of dithered images.

The strong distortion implies that the degree of subsampling will vary across the image even for small dithers. For example, an offset of five pixels at the center of the WFC already introduces an additional shift of 0.4 pixels near the edge of the detector. A larger dither, e.g., 12 pixels at the center, will correspond to an integral shift near the edge (one entire additional pixel), but will provide half-pixel subsampling midway between the center and the edge. Thus, varying degrees of subsampling across the image will be unavoidable with any single pair of dither offsets. Obtaining a larger number of offsets can help produce more uniform sub-sampling across the entire ACS field of view.

Time-dependent Distortion for ACS

A small change in the skew of the ACS images has been detected over time, as have small distortions, which are of higher frequency than can be corrected by the low-order polynomial presently incorporated into the IDCTAB reference files. This time-dependent distortion can result in residual alignment errors between two images taken in different visits of up to 0.3 pixels, and therefore needs to be corrected. Full details of the calibration of this effect were documented in the ACS ISR 2007-08 on the "Variation of the Distortion Solution of the WFC."

This distortion correction term has been included in the latest versions of the IDCTAB reference table as additional keywords in the Primary header of the IDCTAB file itself. The polynomial distortion coefficients get read from the IDCTAB by **updatewcs** when running AstroDrizzle (or **tweakreg** or when running **updatewcs** by itself) and these time-dependent terms get computed and used to adjust the polynomial distortion terms before they get written out as the linear WCS (CD* keywords) and SIP keywords. Thus, the distortion model represented in the keywords in each science image reflects the distortion specific to that image and should correct the image with residual alignment errors of one ACS image to another of much less than 0.1 pixels (with RMS of 0.02 being noted with some datasets).

A.5.2 WFC3

Wide Field Camera 3 was installed during Hubble's fourth servicing mission. The UVIS detector is tilted about one of its diagonals with respect to the light path. This tilt produces a projected rhombus with a diagonal elongation of ~7%, reminiscent of the projected parallelogram of the ACS/WFC detector. The IR detector is tilted about its *x* axis, creating a projected rectangle of elongation of ~8%. Both WFC3 detectors have substantial non-linear geometric distortion, in addition to these projected elongations. The maximum displacement from the rhomboidal projection of the UVIS detector, occurring at two opposite corners, is about 62 pixels (2.5 arcseconds) along the diagonal. The maximum displacement from the rectangular projection of the IR detector, occurring at all four corners, is about 20 pixels (2.4 arcseconds) along the diagonals. These effects make it critical to calibrate the distortion model and remove it in order to be able to combine WFC3 data using AstroDrizzle.

The distortion model has been calibrated to an RMS of roughly 0.1 pixels for both the UVIS and IR channels. Full details on how these models were calibrated can be found in the WFC3 Instrument Science Reports for the WFC3 ISR 2009-33 and WFC3 ISR 2009-34. These calibrations indicated that a simple polynomial solution provided by the IDCTAB reference table would be sufficient for use with AstroDrizzle.

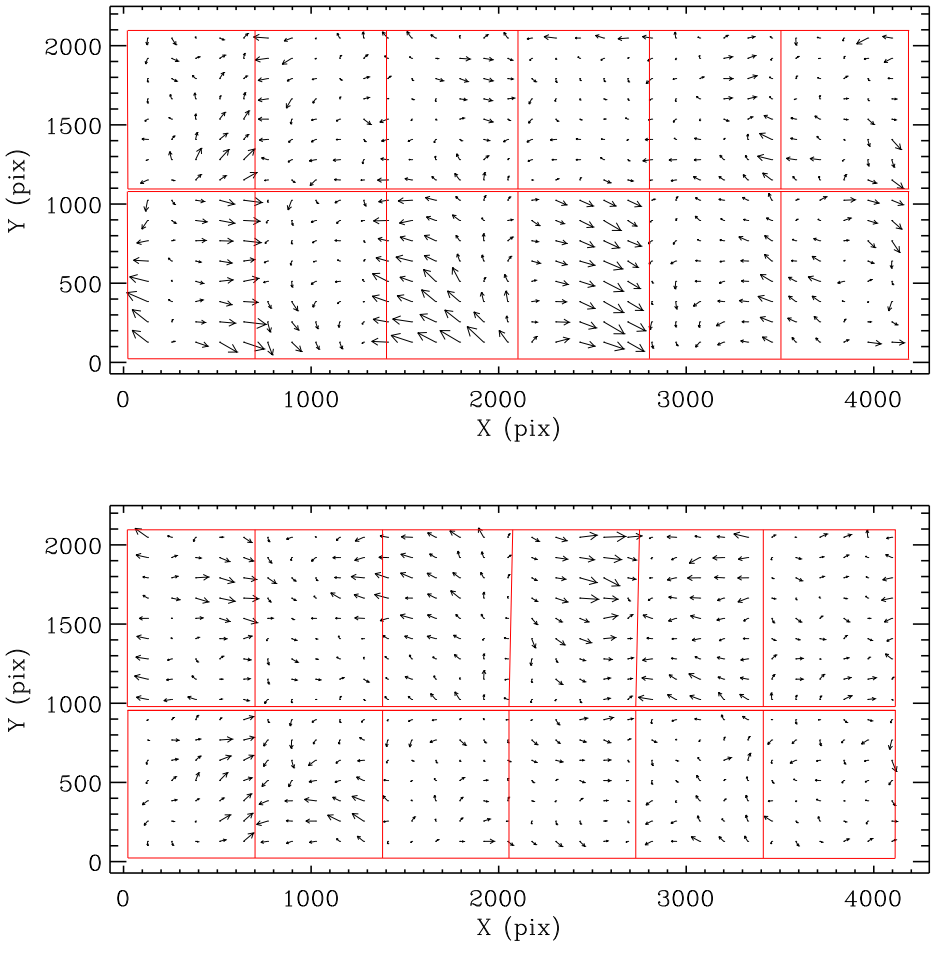
As previously described, the ACS/WFC geometric distortion varies with time. WFC3/UVIS and IR geometric distortion, on the other hand, appears to be stable, with no evidence of secular changes. However, during each interval of the orbital target visibility, there is a linear trend in the UVIS and IR *X* and *Y* scale at the level of 0.05 UVIS and IR pixels. These linear changes are related to focus variations over an HST

orbital time scale, known as orbital breathing, and will not effect the quality of the drizzled products (Kozhurina-Platais and Petro, WFC3 ISR 2012-03).

An additional feature of the WFC3/UVIS geometric distortion is the fine-scale systematic pattern in the residuals from the best-fitting polynomial solutions. These systematic residuals are typically 0.15 pixels in amplitude and depend on positions of UVIS CCD chips and UVIS passbands. A complicated and correlated residuals pattern is due to the photo-lithographic pattern with size of 700 x 1000 pixels, imprinted onto detector itself during the manufacturing process (Kozhurina-Platais et.al., 2010 in "HST 2010 Calibration Workshop").

This complicated structure of residuals cannot be removed by a polynomial model. The simple way to remove the fine scale variations is to model the fine scale structure with a look-up table, which can be linearly interpolated at any point in the image. This look-up table would be analogous to ACS/WFC and ACS/HRC (Anderson & King, 2002) as a reference file `NPOLFILE` in AstroDrizzle.

Figure A.6: Lithographic Pattern of Residuals Seen After Applying WFC3/UVIS Polynomial Distortion Model



The top panel shows the X,Y residuals for the WFC3/UVIS1 CCD chip and the bottom panel, XY residuals for WFC3/UVIS2 CCD chip, after geometric distortion have been removed. The largest vector is ~0.15 pixels, magnified by 2500. The red lines indicate the boundaries of the lithographic pattern The units are WFC3/UVIS pixels.

APPENDIX B:

HST Pointing Accuracy and Stability

In this appendix . . .

B.1 HST pointing stability

An understanding of HST's pointing stability and offsetting accuracy is essential for planning dither observing strategies, regardless of planning for integer or subpixel offsets. Multiple, dithered exposures of the same target with HST could have three types of observing scenarios.

- Within a single orbit:
  - The pointing stability of HST during the orbit, specifically when pointing at a single location
  - The precision with which HST can be offset to different dither locations during an orbit (i.e., a comparison between the commanded and actual telescope offsets)
- Within a single visit (i.e., multiple contiguous orbits):
  - The pointing repeatability after the guide stars are re-acquired at the start of each new orbit
- Across multiple visits:
  - Whether or not the same guide stars are used
  - Repeatability of pointing and roll angle after a full guide star acquisition

Astrodrizzle-combined images from the pipeline are observations taken within a visit (the first and second scenario described above), as specified in the exposure logsheet either using the `CR-SPLIT`, or special requirements, or dither patterns. Such exposure groupings are known as associations. In the pipeline, exposures in an association are aligned based on the WCS of each image, and drizzle-combined to create a combined distortion-free image with suffix `drz.fits`. In most of these cases, the image alignment based on WCS, performed by AstroDrizzle in the pipeline, produces satisfactory results. On some occasions, the alignment may not be optimal due to pointing anomalies. These associations will have to be re-processed by the observer who will need to change the default AstroDrizzle processing for an image association to produce better results.

In post-pipeline image processing, AstroDrizzle and other associated tasks can be used for more complex image combination, such as combining observations taken across several visit or taken at different orientations, as described above.

Statistics on HST pointing behavior have been continually improving thanks to extensive use of dithering to optimize scientific output in several large observing campaigns. These include the Hubble Deep Field North and South (Williams et al. 1996, 2000; Casertano et al. 2000; Gardner et al. 2000), the Hubble Ultra-Deep Field (Beckwith et al, 2006), and long-term monitoring campaigns of the globular clusters M22 and 47 Tuc (programs 7615 and 8267 respectively; Sahu et al. 2001; Gilliland et al. 2000). These observations have provided an excellent body of information regarding precision and repeatability of HST offsets, as well as tracking stability of the telescope when no offsets are commanded (e.g., multiple exposures at the same location). Drawing on experience with these observing programs, more details about HST pointing and stability characteristics for each of the observing scenarios listed above can be described, particularly in terms of positional accuracy of the spacecraft when performing offsets for dithered observational programs. Gilliland (2005) contains a thorough analysis of the datasets, which established the values given in Table B.1.

Table B.1: Typical HST pointing and stability characteristics

Observing Scenario (with fine lock on two guide stars)	Type of Program RMS Precision	RMS Precision
Single pointing	Small programs (no dithering)	< 2 - 5 mas
Offsets within an orbit (recommend < 1 arcsec)	Small programs (with dithering)	~ 2 - 5 mas
Re-acquisition for contiguous orbits in the same visit	Medium-sized programs (e.g., < 5 orbits per target)	5 - 20 mas
Repeatability for different visits, same guide stars and same ORIENT	Large/deep programs (e.g., > 5 orbits per target)	~ 50 - 100 mas
Pointing repeatability with different guide stars	Not recommended unless unavoidable, e.g., due to scheduling constraints	0.2 - 0.5 arcsec
Guiding with a single guide star	*Unavoidable in many crowded fields such as those along the galactic plane*	*1.5 mas/sec roll For ACS: 0.052 arcsec/orbit in WFC*

B.2 HST Tracking Stability at a Single Location

During each orbit, thermal variations in the telescope cause structural variations known as *breathing*, which leads to changes not only in the optical telescope assembly (OTA), but also in the way that the Fine Guidance Sensors (FGS) track guide stars. The breathing manifests itself as time-dependent changes in the shape and centroid of the PSF across the image due to the changing focus.

Changes related to the FGS, on the other hand, depend largely on whether fine lock has been achieved on one or two guide stars. Most observations are obtained with successful fine lock on two guide stars. In those cases, positional drifts would mainly be related to thermal variations that predominantly manifests as positional translations. A small amount of rotation may also occur during an orbit, typically less than a few hundredths of a pixel across the science instrument. Typical RMS tracking accuracy is generally on the order of two to five mas or less throughout an orbit, and can be verified post-facto by examining the jitter files that are part of the archival dataset for a particular observation.

In some observations, however, fine lock is successfully achieved on only one guide star. In this case, a steady roll angle drift is present as a result of gyro drift. The telescope will rotate about the guide star, typically at a rotation rate of ~1.5 mas/sec but rates of up to five mas/sec have occurred on rare occasions. This manifests itself primarily as a translation of the science instrument, but some slight rotation may also be evident. The actual amount of translation of the science instrument on the sky depends on its location in the focal plane relative to the guide star. For example, STIS and NICMOS are located approximately midway between the optical axis and the FGS apertures, so their distance from a guide star could range from 6 to 20 arcminutes. For these instruments, the maximal scenario of a rotational drift of five mas/sec would produce a total translation during one orbit ranging ~25 to 85 mas. For WFPC2 this maximum shift could be ~50 mas.

Thus, before proceeding with the analysis of dithered data, it is always advisable to examine either the `EXPFLAG` keyword value or the jitter data products to confirm whether a two-FGS fine lock was successfully achieved during the observation. If a two-FGS fine lock was achieved, the expected translational shifts due to FGS drift should be less than three mas during the orbit, and any apparent rotation should be less than a few hundredths of a pixel across the detector. The *HST Data Handbook* contains details on how to extract the relevant information from jitter files.

B.3 Precision of Commanded Offsets

If the primary reason for dithering is to avoid bad pixels or improve PSF sampling, then dither offsets less than about one arcsecond are recommended. Examination of HST behavior in previous dither campaigns reveals that for offsets of this size, the actual measured dither offset will agree with the commanded offset to an RMS within about two to five mas during a single orbit with good lock on both guide stars. The RMS of this offset typically increases to a range of up to ~10 to 15 mas when comparing one visit to another over several days. Occasionally, the actual offsets can differ substantially from the commanded offsets by ~0.1 to 0.2 arcseconds or more, with field rotations of up to 0.1°. This is generally the result of FGS false lock on a secondary null, or other FGS interferometric peculiarities. This behavior was observed in two out of nine pointings during the HDF-N campaign.

In some cases, larger dither offsets of up to a few arcseconds are required to bridge inter-chip gaps between detectors, as in WFPC2's four CCDs and the two detectors in ACS/WFC. Offsets of this size are unlikely to present any problems with pointing precision but observers should be aware that such offsets may introduce more non-uniform subsampling across the field as a result of the geometric distortion inherent in the instruments.

Offsets larger than several tens of arcseconds may result in guide stars moving out of the FGS apertures, depending on the exact configuration of the primary and secondary guide stars. This would necessitate a full target acquisition using new guide stars, with substantial associated overhead, as well as a loss of pointing repeatability due to the relative positional uncertainties in the guide star catalog (~0.2 to 0.5 arcseconds). Such large offsets are more appropriate for mosaicing programs where large areas are being mapped, and would therefore involve a fundamentally different proposal design than those programs involving small dither offsets.

B.3.1 Pointing Repeatability After Guide Star Re-acquisition

For many HST programs, dithered observations of a target are obtained during a number of separate orbits, often contiguous, which are in turn grouped into one or more visits.

The first orbit in a visit begins with a full guide star acquisition. For each subsequent orbit in the same visit, HST will reacquire the same guide stars upon exit from occultation. In post-occultation guide star reacquisitions, the instrument pointing is typically within ~5 to 20 mas of its location in the previous orbit.

The precision of HST's guide star reacquisition is based on its ability to force the post-slew position of the guide stars to reside in the exact same location in the guide star acquisition field-of-view (i.e., pickles), as in the previous orbit. This is generally sufficient to reliably perform subpixel dithers for most HST instruments that have pixel sizes of the order ~0.05 to 0.1 arcseconds. Therefore, it's recommended that, whenever possible, the observing proposal should be designed to fit all dithered observations of a given target into a contiguous set of orbits within a single visit to provide improved relative image registration.

B.3.2 Roll Angle Repeatability Over Multiple Visits

Some observing programs are sufficiently large enough to necessitate dithered observations of the same target over many orbits. In such cases, it is necessary to break the observations into several visits because the length of a single visit is constrained by available scheduling windows depending on the target's position in the sky. For all targets outside the CVZ, single visits are usually constrained by scheduling limitations to contain no more than five orbits. If multiple visits of the same target were scheduled across different dates, images in one visit may have small offsets relative to images from other visits, even if the same pointing, same guide stars, and same `ORIENT` were specified for the visits.

At the start of a new visit, HST sets up the specified roll for the observation using the gyros, and carries out a full acquisition of the dominant guide star. This is followed by the acquisition of the sub-dominant guide star, which enables the telescope to track in fine lock. The pointing control system (PCS) then preserves this roll angle for the remainder of the visit.

In most cases, the difference between the desired roll angle, and the actual roll angle, will be less than ~0.003°. This corresponds to a positional shift of approximately 73 mas at the sub-dominant guide star, assuming a separation of 1,400 arcseconds between the two guide stars. For WFPC2, this shift is 38 mas, i.e., just less than the size of a WFPC2/PC pixel. Therefore, multiple visits at the same specified roll, target, and guide stars will, under nominal circumstances, show repeatability to this level. It is not uncommon for scheduling constraints to affect the time between updates to the Fixed Head Star Trackers (FHSTs) and FGS acquisitions, in which case roll angle deviations of 0.01° and greater can occur (i.e., translational shifts above 100 mas). For visits with the same guide stars and requested roll angle, the actual roll changes incurred between visits can be accurately determined from the locations of guide stars in the FGS as recorded in the datasets' jitter files.

APPENDIX C:

Observational Dithering Options for Drizzling Data

In this appendix . . .

C.1 Dithering Strategies

C.1.1 What is Dithering?

An increasingly popular technique in UV, optical, and IR imaging observations involves the use of dithering, that is, spatially offsetting the telescope by shifts that are generally small relative to the detector size and therefore moving the target to a number of different locations on the detector.

Two of the main strategies involve,

1. Offsets by an integer number of pixels to facilitate the removal of bad pixels
2. Offsets by sub-integer pixels to improve spatial sampling of the point spread function (PSF)

The latter application is particularly important in the case of HST because the PSF is so small that it is significantly undersampled by most primary science instruments.

3. A third spatial offsetting technique involves the use of large shifts, comparable to the scale of the detector, to fully map areas of the sky that are several

times larger than the detector area. This is generally referred to as *mosaicing*, the technique used for observations like the Hubble Deep Field and Ultra-deep field. In this context, these refer to large mosaics created from data taken over multiple visits using different guide stars, and should not be confused with mosaic-type dither patterns. Large multi-visit mosaics involve observational considerations and methods of data analysis that are beyond the scope of this document. However, the techniques covered in this document are essential to mosaicing with HST.

C.1.2 Benefits of Dithering

Dithering of HST observations is hardly new; the primary data acquisition modes of the GHRS and FOS involved both sub- and multi-diode offsets to obtain well-sampled data along the spectral dimension without gaps resulting from the presence of a few dead diodes. However, dithered observations in imaging mode became routine only after the dramatic improvement in HST optics with the installation of COSTAR and WFPC2 in 1993.

Dithering often provides considerable benefits to a science program, specifically in the following ways:

1. Dithering reduces the effects of pixel-to-pixel errors in the flat field or in spatially varying detector sensitivity.
2. Integer shifts of a few pixels allow the removal of small-scale detector defects such as hot pixels, bad columns, and charge traps from the image.
3. Non-integral (subpixel) dithers allow, when correctly implemented, the recovery of some information lost to undersampling by pixels that are *not* small in comparison to the point spread function.

The third point is of particular importance for HST imaging; almost all the imaging instruments aren't able to fully take advantage of the resolving power of HST optics. This is because instrument designers had to decide between fully sampling a small field of view, or using coarser sampling on a larger field.

Dithering was particularly important when WFPC2 and NICMOS were HST's primary imagers. The width of a WFPC2 WF pixel, about 0.1 arcsec, was already comparable in size to the optics full-width-at-half-maximum (FWHM) in the *I*-band and substantially larger in the *blue* band. Images from the NICMOS camera-3 detector were similarly undersampled over much of its spectral range. While the ACS/HRC provided an adequately sampled PSF at optical wavelengths, it came at the cost of a drastically reduced field of view–it was just 1/50th the area of ACS/WFC.

HST's active imagers, WFC3/UVIS, WFC3/IR, and ACS/WFC, have detector pixel widths comparable to the FWHM of the point spread function (PSF). In theory, a minimum of two samples per FWHM would be required for full recovery of the image resolution. While the “missing” samples could be recovered with dithering, it's not possible to completely undo the low-level blurring produced by a larger pixel. Still, dithering provides substantial improvements to the final image quality, including better removal of detector defects.

C.1.3 Costs and Drawbacks of Dithering

While dithering provides substantial benefits, there are a number of trade-offs that must be understood and considered when deciding whether or not to obtain dithered data. Additional details are described later in the document, but summarized below.

- Post-pipeline reprocessing of data with the **drizzlepac** package requires special reductions that require more work for the observer.
- Additional spacecraft overhead time is needed for small angle maneuvers between dither points. Users will have to make some judgment calls about what they can do within their allotted orbits by testing different observing scenarios using the Astronomer's Proposal Tool (APT).
- A long exposure broken into shorter exposures at each dither point will increase the amount of read noise in the final combined image.
- If the primary science goal is to measure differential changes over time, as in time-series photometry, dithering may complicate data analysis due to detector intra-pixel sensitivity variations. While this is a concern for NIC3 data analysis, it should not be an issue for WFC3/IR since there was no indication of intra-pixel sensitivity during ground testing.
- Cosmic ray elimination may be more difficult for subpixel dithers with one image per pointing, if only two pointings are being taken.

For most HST observing programs, potential drawbacks of dithering are outweighed by the scientific benefits. There are, however, some situations where the drawbacks could outweigh the benefits, such as in programs with very limited observing time.

For questions related to how your observing program could be affected by dithering, please consult the Phase 2 Proposal Instructions and use the Astronomer's Proposal Tool. If you need additional assistance, please get in touch with your Contact Scientist or send an e-mail message to the STScI Helpdesk.

C.2 Selecting the Right Dither Strategy

The dithering strategies outlined in this section are guidelines, not solid rules. There will likely be science programs that do not neatly fit into any one of these dithering categories. If answers to your question are not found in this document, the Phase 2 Proposal Instructions, instrument web pages, and the Astronomer's Proposal Tool (APT), please get in touch with your Contact Scientist or send an e-mail message to the STScI Helpdesk.

C.2.1 Dealing with Cosmic Rays, Hot Pixels, Undersampling, and Photometric Accuracy

During the Phase II proposal writing stage, users are faced with the challenge of crafting the best possible observing program within the allocated telescope time. For instance, a long observation broken into multiple dithered exposures comes at a cost: increased readout noise and less total science exposure time (due to observational overheads). How much of that cost can be incurred without compromising science goals? That depends on the purpose of the observations: is it to detect an unknown underlying structure in the field? Is high spatial resolution a priority? Is there a requirement for highly accurate photometry?

Designing an observing program to get the best quality data depend on how to deal with cosmic rays, hot pixels, spatial sampling, and signal-to-noise.

- **Cosmic rays**: a minimum of two exposures, preferably three or more, is the most effective way to reduce the number of cosmic ray hits at the same detector location in each exposure (according to the binomial distribution). Even with two exposures, it's possible to get overlapping cosmic ray hits in the two images. In the example below, generated using the WFPC2 Exposure Time Calculator, a 3000 second exposure for a 24 magnitude point source in F122M with gain of 7 is broken into several sub-exposures to show how the number of overlapping cosmic ray pixels is reduced as more sub-exposures are used for the combined image, but at the cost of decreased signal-to-noise. (Items in parenthesis are additional comments). A 3000 second exposure would lose about 61,000 pixels per chip to cosmic ray hits. If two 1500 second images were combined, the number of cosmic ray-affected pixels drops dramatically to 1400 per chip. If three 1000 second images were combined, only 21 pixels per chip would be affected by cosmic rays.

Table C.1: WFPC2 Exposure Time Calculator Shows Changes in SNR and Cosmic Rays Based on Number of Split Exposures

# of sub-exposures	Total SNR	Pixels Lost	Comments
(No Split)	3.7	9.606250%	(about 61,000 pixels)
2	2.9	0.230700%	(about 1400 pixels)
3	2.5	0.003283%	(21 pixels)
4	2.3	0.000033%	
6	1.9	0.000000%	

- **Hot Pixels**: flat-field calibrated images (`flt.fits`) from the Archive are processed with dark reference files that contain hot pixel information for a time period during which the observation was obtained. (It usually takes a few weeks for the most up-to-date dark reference files to catch up with the science observations; if you retrieve images taken within a month to six weeks of execution, chances are that the dark reference file used to calibrate the data does not match the observation date.) Since some hot pixels are variable on very short timescales, they're not flagged in dark reference files. Therefore, the easiest way to remove hot pixels is to dither the images. A two-point dither with small integer shifts is enough to remove most hot pixels.
- **Undersampling**: HST cameras, with exception of the ACS/HRC, have detector pixel widths comparable to the FWHM of the point spread function (PSF). Drizzle-combining images that are shifted by subpixel amounts can improve PSF sampling, (in other words, increase spatial resolution). Generally, subsampling by a small shift with a 1/2-pixel offset provides the best improvements over non-dithered images. In some cases, observers may wish to further explore the limits of the instrument and spacecraft pointing accuracy by considering small shifts with 1/3-pixel offsets. The extent to which such refinements can be explored depends primarily upon the number of orbits available and the instrument being used.
- **Photometric Accuracy**: HST instruments can vary in sensitivity across each individual pixel; this is referred to as intra-pixel sensitivity. For an undersampled PSF, this can complicate photometric analysis of dithered images. Therefore, programs requiring highly accurate photometry measurements may not always benefit from dithering.

C.2.2 A Top-level View of Dithering Strategies

Decisions on how to implement dithering in your observing proposal depends on many factors, some of which were discussed in the previous section, others that will be explained later in this chapter. With the exception of mosaic dithers, dither step sizes are kept small to minimize differential geometric distortion between the images

but also need to be large enough to remove chip gaps for detectors like WFC3, WFPC2, and ACS/WFC. This keeps the step size at each position in each image more nearly the same so that every pixel gets as close to the same level as possible of subpixel sampling as intended by the dither pattern.

At a top level, there are several dithering categories:

- Some types of observations may be unsuitable for dithering
  - Very Short Exposures: if each target is observed for less than a few minutes, extra overhead from dithering could significantly impact the overall signal-to-noise (S/N) that may offset advantages gained from dithering.
  - Critical photometric measurements: for high-precision time-dependent photometric monitoring, dithering may introduce additional complications due to intra-pixel sensitivity variations. Therefore, some observers may prefer to obtain all the images at a single pointing location.
- Simple Dithering: dithering each exposure by integer pixel shifts reduces the impact of hot pixels in the final combined image. Furthermore, spatial sampling can be improved with two- or three-point subpixel dithering. It is possible to do cosmic ray rejection with a single image at each dither point but for a more robust rejection, two or more exposures at each pointing is recommended. For programs allocated about one orbit per target, at least two to three exposures should be obtained to facilitate cosmic ray rejection. If one is interested in targets throughout the field, rather than one single small target, cosmic ray removal will need to be more rigorous, and a larger number of exposures will be required. The instrument handbooks give expected cosmic ray rates for each of the imaging instruments.
- Full Dithering: for improved spatial sampling, a “full” four-point dither, with 1/2 pixel subsampling along both detector axes, is recommended. Most of the subpixel information in an image can be recovered by a four-point dither. Deep programs may benefit from an even larger numbers of dithers. Obtaining a four-point dither across the field of view limits the user to small dithers because of the distortion of many HST cameras. At the same time, the user may want to remove features such as the slit between the two chips on ACS with a large dither. The user may want to combine several sets of four-point dithers in this case. In addition, in cases where there are small objects with high signal-to-noise, image quality can be improved by using dithering patterns sampled finer than four points.
- Dithering for Parallel Images: it is not always possible to obtain optimal dithers simultaneously for primary and parallel instruments due to the large separation between detectors, and different pixel scales. Uniformly-spaced dithers for the primary instrument generally yield non-uniform dithers for the parallel instrument. In most cases, we recommend that users select their dither pattern to obtain the best possible data from their primary instrument.
- Dithering in WFPC2: the Planetary Camera (PC) and Wide Field Cameras (WFC) have different scales; therefore, a dither pattern was developed to implement subpixel dithering in both camera types. (WFPC2 has been decommissioned.)

C.2.3 Selecting the Number of Dither Pointings and Step Sizes

Dithering requires a noticeable amount of spacecraft overhead, with each dither offset typically adding about two to three minutes of overhead to the total observing plan. Outlined below are recommendations for various observing goals.

Integer-Spaced Dither Steps

Two to three integer-spaced dither steps will, in most cases, correct the effects of hot pixels; if the flux from an object fell on a hot pixel in one image, chances are good that it will fall on a normal pixel in the other dithered image.

Subpixel Dithering

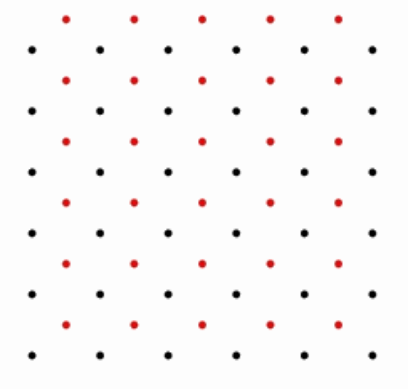
Strategies and issues for subpixel dithering are covered in the remainder of this section. The number of subpixel dithers for an observation depends on the amount of available observing time and project goals.

- The simplest type of subpixel dither is a two-point dither offset along only one axis; this is used in STIS long-slit spectroscopy for subsampling along the (spatial) slit direction. For example, one exposure would be obtained at the original pixel position of *(0, 0)* and a second obtained at *(0, n+1/2)* pixels where *n* is an arbitrary integer value.
- For imaging, a two-point subpixel dither starting at the original pixel position of *(0,0),* followed by a second image shifted by *(n+1/2, m+1/2)*, where *n* and *m* are arbitrary integer values, will provide a substantial increase in information over non-dithered data. For square detector pixels, this dither pattern results in sampling that would be produced by an array with a pixel size that's $\sqrt{2}$ smaller than the original array, rotated by a $45°$ angle from the original orientation. Setting *n* and *m* to a small integer value, around 5 to 10, will also allow the removal of hot pixels. Figure C.1 shows the sampling by the WFC3 IR detector on the sky (note the slightly rectangular pixels[1]), and Figure C.2 shows the sampling produced by introducing a two-point dither. The original placement is shown in black, the second dither is in red.

1. WFC3/IR pixels are not square pixels, but slightly elongated in the *x* direction.

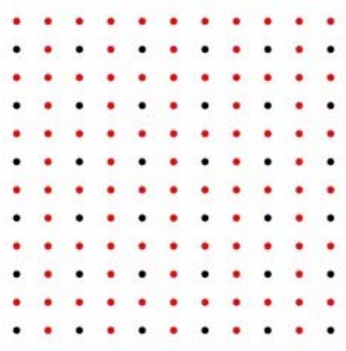
Figure C.1: The Sampling of the WFC3 IR Detector on the Sky

Figure C.2: The Sampling Produced by Introducing a Two-point Dither Using the WFC3 IR Detector.



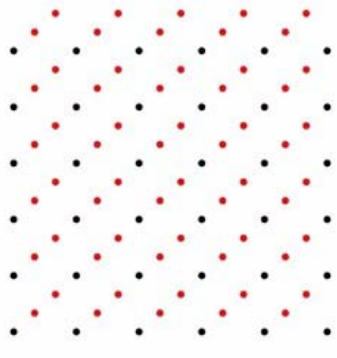
- A four-point dither yields a total of 4 images that have 1/2-pixel offsets in *x* and *y*, as well as small integer shifts (*n, m*) to reduce the effect of hot pixels: *(0,0), (0,m+1/2), (n+1/2,m+1/2), (n+1/2,0)*. This yields uniform tiling along the x- and y-axis with half-pixel offsets, providing a more robust and powerful sub-sampling of the PSF. In fact, given the native sampling of HST instruments, an accurate four-point dither recovers nearly all of the information available in an image (see Figure C.3).

Figure C.3: A Four-point Dither



- Use of a three-point dither may arise for cases where the available observing time breaks down more naturally into blocks of three exposures, instead of two or four. However, the best placement for a three-point dither is not obvious because there is no natural way to tile the plane using three placements of a rectangular CCD grid. Two- and four-point dither patterns described earlier minimize the largest distance of any point on the image plane to the nearest dither location. Therefore, what kind of three point dither pattern would have the same characteristics? Based on a (computer) calculation, this can be done with offsets along the diagonal of *(0,0), (1/3,1/3), and (2/3,2/3)* pixels. Again, additional integer offsets of a few pixels should be added to help remove detector defects. Figure C.4 shows a three-point dither applied to the WFC3 NIR detector.

Figure C.4: A Three-point Dither Applied to the WFC3 NIR Detector

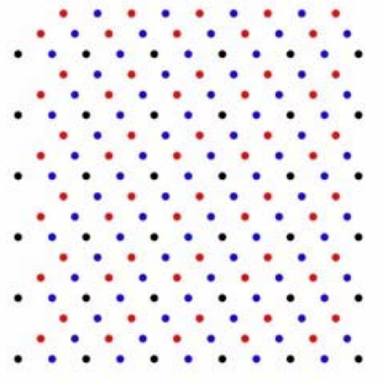


- If the goal is to obtain extremely accurate PSFs from observations spanning several orbits, users may consider an even finer subsampling of the pixel. An eight-point dither could be performed by crossing a four-point dither with a two-point dither; in other words, a secondary dither pattern at the location of each point in a primary dither pattern. That secondary dither should be a two-point dither of the form *(m + 1/4,n + 1/4)* which would place a point in the center of each "square" created by the primary four-point dither pattern. Note, however, that differential distortion across the field can mean that unless the integer offsets are small, a well-planned dither strategy for the center of the chip will perform worse near the edges where the distortion will result in a dither pattern that varies significantly from the center of the chip. The four-point dither, if performed accurately with the loss of few pixels to cosmic rays or other defects, recovers nearly all the spatial information in an HST image. Therefore, users of instruments like ACS/WFC may prefer to cross a small four point dither with a larger two or three point dither that will cover the gap between the chips. The four point dithers will insure good subsampling in the final combined image.

- A number of WFC3 users have inquired about dithers in multiples of three because they find that three exposures fit well into a single orbit. One option[2] is to create a nine-point dither by dividing the original pixel with a 3x3 grid.

Forming a six-point dither[3], however, is less clear; a calculation suggests that crossing the linear three point dither, described above, with a (1/2,0) two-point dither is the optimal strategy. For square pixels, the half-pixel dither could be taken in either direction. But WFC3 IR pixels are slightly longer in the x-direction, so the dither should be performed along the x-axis. In Image:6pt_small.jpg, the black points show a single WFC3 image; the red points show the two additional dithers to form a single three-point dither; the blue points show the additional three-point dither to form the six-point dither.

Figure C.5: A Six-point Dither



C.2.4 Data with Inaccurate Offsets in Position or Roll Angle

On rare occasions, pointing errors may occur during guide star re-acquisitions within a visit. As a result, images in the same visit cannot be aligned based on their WCS information. This will be evident in pipeline drizzle-combined images that may show double objects, elongated psfs indicating sub-pix misalignments, and artifacts like “chopped” psfs. Tasks such as **tweakreg** can be used to measure and correct the offsets between images, so they can be properly aligned and reprocessed with **astrodrizzle**.

2. A three-point dither with another three-point secondary dither at each point:*(0,0), (0,a+1/3), (0,a+2/3), (m,m), (m+a+1/3), (m+a+2/3), (n,n), (n+a+1/3), (n+a+2/3),* where *(0,0), (m,m), (n,m)* is a three-line dither with integer steps, and *a* is a small integer added to the fractional shift.

3. The six-point dither: *(0,0), (0,m+a+1/2), (m,m), (m+a+1/2), (n,n), (n+a+1/2)* where *(0,0), (m,m), (n,m)* is a three-line dither with integer steps, and "a" is a small integer added to the fractional shift.

C.2.5 How Many Images to Obtain at Each Dither Location

In general, most cosmic rays can be removed with one image at each dither pointing. It's the best approach for small programs (one orbit per target) that require subpixel dithering, and for programs that require low read noise (such as narrow-band imaging of extremely faint sources). For larger programs, or for programs where read noise is not a serious issue, users can opt for slightly improved sampling by executing a small secondary subpixel dither at each pointing of a larger primary dither pattern or they could choose to obtain multiple exposures at each primary dither pointing. Implementing multiple exposures at each dither position insures that cosmic ray rejection can be performed in all pixels of each image, whereas dithered observations will result in only one image on the edges of the combined image and making identification of cosmic rays by AstroDrizzle impossible in those regions.

C.2.6 Specific Instrument-related Issues

WFC3

Summarized below are WFC3 dither strategies for UVIS and IR channels. Additional information is available in the Phase II Proposal Instructions, Section 8.4.4, as well as Appendix C in the *WFC3 Instrument Handbook*.

WFC3 dither patterns designed to subsample pixels can be used as secondary patterns to complement primary WFC3 patterns with larger steps. It is also possible to use WFC3 patterns as secondary patterns within a primary pattern based on generic `BOX`, `LINE`, or `SPIRAL` pattern types. When combining patterns, the smaller dither pattern should be the secondary pattern to minimize time spent in moving the telescope.

WFC3/UVIS

WFC3/UVIS images are in many ways similar to ACS/WFC images. The detector is comprised of two rectangular CCD chips separated by a gap approximately 35 pixels wide, so that a gap-stepping dither is needed to avoid having a gap across the center of the field of view. The projection of the pixels on the sky is in the shape of a rhombus, with an angle between the *X* and *Y* axes of 86°. As with the ACS/WFC, a `POS TARG` in *Y* is along the *Y*-axis of the aperture (along columns), and a `POS TARG` in *X* is perpendicular to the *Y*-axis (not quite along rows).

The plate scale is 0.04 arcseconds/pixel on each axis, and the FWHM of the point spread function is between 1.6 and 2.3 pixels, depending on wavelength. WFC3/UVIS images will thus benefit from half-pixel dithering, but not as much as WFC3/IR images. Non-linear distortion causes the projected area of the pixels to vary by +/-3% relative to that at the center of the detector, so `POS TARG`s and patterns will produce shifts in pixels that vary with location on the detector.

Seven patterns are available for Phase 2 dithering and mosaicing of WFC3/UVIS images.

- `WFC3-UVIS-DITHER-LINE` dithers the UVIS aperture by (2.5, 2.5) pixels to sample the point spread function with fractional pixel steps.
- `WFC3-UVIS-DITHER-LINE-3PT` dithers the UVIS aperture by (2.33, 2.33) pixels to sample the point spread function with fractional pixel steps.
- `WFC3-UVIS-GAP-LINE` dithers over the gap between the two chips of the UVIS detector with relative steps of (-2.25,-30.25) and (2.25,30.25) pixels.
- `WFC3-UVIS-DITHER-BOX` samples the point spread function with fractional pixel steps and produces spacings of more than one column to remove hot columns. The relative steps in pixels are (0, 0), (4.0, 1.5), (2.5, 4.0), and (-1.5, 2.5).
- `WFC3-UVIS-MOS-DITH-LINE` has a primary pattern that dithers over the gap between the two chips of the detector with relative steps of (-4.5,-60.25), (0, 0), and (4.5, 60.25) pixels. A secondary pattern adds a dither of (2.5, 1.5) pixels to the primary pattern.
- `WFC3-UVIS-MOS-BOX-LRG` produces a UVIS mosaic that can be executed with a single set of guide stars. It dithers the gap between the chips so that no region lies in the gap more than once. The relative steps in pixels are approximately (-1000, -997), (1000, -1001), (1000, 997), and (-1000,1001).
- `WFC3-UVIS-MOSAIC-LINE` is designed for observations using the full WFC3/UVIS detector for primary exposures and the full ACS/WFC detector for parallel exposures. It dithers over the inter-chip gap on both detectors. The relative steps on the WFC3/UVIS detector are (0, 0) and (36.5, 71.5) pixels.

For programs requiring high precision small aperture photometry, observers should see (Brown 2008) for a discussion of features called "droplets," caused by contamination on the outer window of the UVIS detector. Dithers ~100 pixels are recommended to improve photometry.

The WFC3 pipeline produces cosmic ray-rejected (CRJ) images from input `flt.fits` images for `CR-SPLIT` exposures, but these have not been corrected for geometric distortion. When AstroDrizzle is run in the pipeline, it will use the `flt.fits` images as input, tagging cosmic rays in those images with a different data quality flag value (4096) from those cosmic rays found by CALWFC3 (DQ value of 8192). Observers are generally encouraged to use dithers instead of `CR-SPLIT` exposures for a number of reasons: to change the placement of hot pixels on the field, to resample the point spread function, and to reduce the impact of pixel-to-pixel errors in flats.

WFC3/IR

The WFC3/IR pixels are projected as rectangles on the sky, with *X* and *Y* plate scales ~0.14 and 0.12 arcseconds per pixel. A `POS TARG` in *Y* is along the *Y* axis of the aperture (along columns), and a `POS TARG` in *X* is along the *X* axis (along rows). The FWHM of the point spread function is between 1.0 to 1.25 pixels, so subpixel dithering is needed to recover spatial resolution. Non-linear distortion causes the projected area of the pixels to vary by +/-4% relative to that at the center of the

detector, so POS TARGs and patterns will produce shifts in pixels that vary with location on the detector.

Note that there is a ~45 pixel diameter dead spot near the lower edge of the WFC3/IR detector, centered at about (358, 54). A dither larger than this diameter should be used if imaging in that area is required, such as the WFC3-IR-DITHER-BLOB pattern.

Five patterns have been installed in the phase 2 software to dither and mosaic WFC3/IR images:

- WFC3-IR-DITHER-LINE takes steps large enough for photometric accuracy and samples the point spread function with fractional pixel steps. The relative steps in pixels are (0, 0) and (3.5, 3.5).
- WFC3-IR-DITHER-LINE-3PT takes steps large enough for photometric accuracy, and samples the point spread function with fractional pixel steps. The relative steps in pixels are (0, 0) and (3.33, 3.33).
- WFC3-IR-DITHER-BOX-MIN takes steps large enough for photometric accuracy and samples the point spread function with fractional pixel steps. The relative steps in pixels are (0, 0), (4.0, 1.5), (2.5, 4.0), and (-1.5, 2.5).
- WFC3-IR-DITHER-BOX-UVIS is a four-point box pattern that produces an IR mosaic covering the same area as the UVIS detector. The IR imaging is intended to be accompanied by a UVIS exposure (or small dither pattern) using the aperture UVIS-CENTER.
- WFC3-IR-DITHER-BLOB dithers over the IR "blobs," described in WFC3 ISR 2010-09, using relative steps of (-14.25,-14.25) and (14.25,14.25) pixels.

WFC3/IR exposures are made with predefined timing sequences of non-destructive reads. As in NICMOS, up-the-ramp fitting of the fluxes in the sequence is used to identify and remove cosmic ray flux from each pixel. The accuracy of the procedure depends on the timing sequence and the number of frames specified in the proposal, just as the accuracy of traditional cosmic ray rejection in CR-SPLIT exposures on a CCD detector depends on the number of exposures and the exposure time. This cosmic ray rejected IR flt images for WFC3/IR and NICMOS data is used as input to AstroDrizzle, so for most cases the final drizzle step would be necessary while sky subtraction may be performed depending on the data. As for WFC3/UVIS images, DQI values of 4096 and 8192 are used to tag pixels with cosmic rays identified by AstroDrizzle and by **calwfc3**, respectively. These DQ values may be reset when running AstroDrizzle to insure that only a successful run of AstroDrizzle based on proper alignment and parameter usage will be used to update the DQ arrays with properly identified cosmic rays.

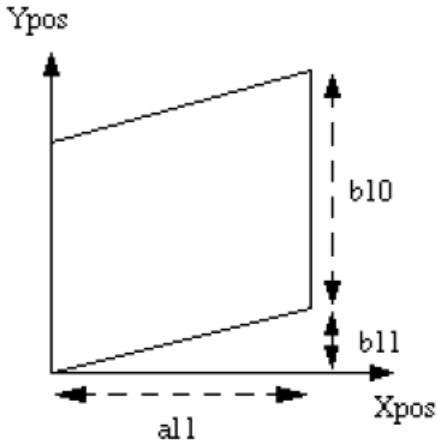
ACS

When taking several exposures of a field in a single filter, observers are generally encouraged to use dithers instead of `CR-SPLIT` exposures for a number of reasons: to change the placement of hot pixels on the field, to resample the point spread function, and to reduce the impact of errors in the pixel-to-pixel flats.

The relatively large distortion, up to 8% across the WFC camera, is an important consideration in designing a dither strategy. Each detector pixel, when projected on the sky, corresponds to a parallelogram. Interior angles of these parallelograms differ from 90° by as much as 5°, depending on pixel location in the detector. As a result, a shift of several pixels will produce noticeably different subpixel offsets in the distorted (i.e., `raw.fits` or `flt.fits`) image. The two WFC chips are separated by about 2.5 arcseconds (about 50 WFC pixels). Therefore, many ACS/WFC dither strategies use offsets large enough to overlap this gap, which results in different subpixel shifts effects in the distortion-corrected image.

Since dither offsets are executed as `POS TARG` shifts along the detector *x* and *y* axes, this means that each `POS TARG` shift on the sky follows the edges of a parallelogram. These shifts have been defined such that displacements in rectilinear sky coordinates are aligned along the *y*-axis of the detector (Mutchler and Cox 2001). For example, a displacement of one WFC pixel along the *x*- and *y*-axes of the detector is represented in sky coordinates as follows: one pixel displacement along the detector *y*-axis is a 0.0497 arcsecond offset along the *Y-POS* direction; however, a one pixel displacement along the detector *x*-axis requires a 0.0496 arcsec offset along the *X-POS* direction and 0.0038 arcseconds along the *Y-POS* direction, due to the non-orthogonality of the pixels on the sky. (*X-POS* and *Y-POS* are the undistorted `POS TARG` reference frame.)

Figure C.6: ACS POS TARG Shifts in Sky and Detector Coordinates



$$XPOS = x.\ a11 + y.\ a10$$

$$ypos = x.\ b11 + y.\ b10$$

$$Pointspacing = SQRT(XPOS^2 + YPOS^2)$$

$$pattern\ orient = atan((YPOS)/(XPOS))$$

Distortion coefficients (just the first two terms of the full 4th-order polynomial)

Instrument	a10	a11	b10	b11
WFC	0.0000	0.0494	0.0494	0.0040
HRC	0.0000	0.0283	0.0248	0.0029
SBC	0.0000	0.0336	0.0301	0.0033

A number of pre-defined dither patterns that cover a wide range of observing requirements have been created for ACS detectors (Mutchler and Cox 2001), and are incorporated in the Phase II Astronomer's Proposal Tool (APT) software. These patterns are summarized below, more details are available in the Phase II Proposal Instructions.

- `DITHER-LINE`: observations are taken along a "line" with integer pixel shifts in the detector *x*- and *y*-axis to remove most detector artifacts like hot pixels. For WFC, the shifts are 5 pixels in *x* and 60 pixels in *y* to cover the inter-chip gap. The SBC shifts are 10 pixels in *x* and *y*. (HRC, now inoperable, had 5 pixel shifts in *x* and *y*.) This pattern could be augmented by a secondary dither pattern at each `DITHER-LINE` pointing for PSF subsampling using half-pixel offsets (i.e., [2.5, 1.5]) or even one-third pixel offsets (i.e., [0, 0], [2.3, 1.3], [4.6, 2.6]).
- `MOSAIC-LINE`: large offsets, comparable in size to the detector, taken along a line to increase the field of view. By default, this is a 2-point dither. For WFC, the default offset size is 1948 pixels along the *y* axis (about 47% of the *y*-dimension for the combined WFC detectors as projected on the sky) which gives a 200 x 300 arcsec field of view. This configuration, in most instances, allows the same guide star pair for the two pointings and will cover the inter-chip gap. For the HRC and SBC, this is a shift of about 95% of the *y* dimension of the detector (973 pixels) as projected on the sky, which nearly doubles the field-of-view. Up to nine pointings are allowed for this dither pattern. Sub-pixel dithers, as described above, may also be used at each pointing for better sampling.
- `DITHER-BOX`: a set of four pointings using integer-pixel and half-pixel offsets at each pointing to provide better PSF subsampling. WFC, SBC, and HRC have the same relative pixel offsets: (0, 0), (5.0, 1.5), (2.5, 4.5), (-2.5, 3.0).
- `MOSAIC-BOX`: a four-point dither pattern with large offsets, comparable to the size of the detectors. This pattern cannot be used for WFC because it requires the use of two guide star pairs, and therefore has to be manually implemented as `POS TARG` special requirements. For SBC and HRC, the default pattern uses shifts that are 95% the detector dimensions to create a mosaic roughly four times the field of view.

These predefined patterns should cover the needs of the vast majority of scientific program. Other patterns can also be created using a combination of `POS TARG` offsets.

STIS

STIS is used to obtain images and spectra; therefore, the best dithering method depends on the science goals for the observing program. Those goals may require increased spatial resolution, removal of hot pixels, and minimizing uncertainties in pixel-to-pixel sensitivity with respect to the reference flat fields.

Additional information about STIS dithering strategies are available at Section 11.3 of the *STIS Instrument Handbook*. A summary is provided below.

Image Mode Dithering

Observers can reduce the effect of flat-field uncertainties (particularly for the MAMA detectors) by using a small step pattern with integral pixel shifts. This effectively smoothens the detector response over the number of steps, achieving a reduction in pixel-to-pixel non-uniformity by the square root of the number of steps, assuming the pixel-to-pixel deviations are uncorrelated on the scale of the steps. This approach requires sufficient signal-to-noise to allow image registration.

Alternatively, the spatial resolution may be somewhat improved with a dither pattern that includes subpixel shifts. Images obtained with the STIS/CCD (0.0508 arcsec/pixel)[4], have nearly the same spatial scale as those obtained by the WFPC2/PC camera (0.045 arcsec/pixel); spatial resolution improvements from dithering would be similar for both cameras. The spatial scale of MAMA images is half that of the CCD images, so the gain in spatial resolution from dithering MAMA images would be more modest, and probably insignificant for the majority of programs. Although the PSF on the MAMA detectors should be narrower than on the CCD because of the shorter wavelengths at which the MAMAs operate, in practice this advantage is offset by additional complications introduced through the instrument optics. It is important to realize that the focus varies across the field of view for STIS imaging modes, with the optical performance degrading by about 30% at the edges of the field of view. Thus, the achievable spatial resolution is significantly compromised in those regions.

Whether or not the dither pattern includes subpixel shifts, the effects of bad columns, hot pixels, and other detector artifacts on the CCD can be reduced or eliminated if the dither pattern is greater than a few pixels. Predefined dither patterns are available for observers to use, which include:

- `STIS-CCD-BOX`: A four point parallelogram scan designed for dithering across the CCD pixels. The default setting produces relative pixel shifts of (0, 0), (10, 5), (15, 15), and (5, 10) to compensate for hot pixels and small-scale detector non-uniformities.

4. http://www.stsci.edu/hst/stis/documents/handbooks/currentIHB/c13_specref03.html#922126

- `STIS-MAMA-BOX`: Like its CCD counterpart, this pattern produces a four point parallelogram with the same relative pixel positions.
- `STIS-SPIRAL-DITH`: A spiral dither pattern, starting at the center and moving outward counterclockwise. When this pattern uses four points, the result is a square pattern. For a 4 point dither, this is not the optimum strategy. Users will get better results, such as enhanced resolution, by using the `STIS-CCD-BOX` or `STIS-MAMA-BOX` patterns.

Spectroscopic Mode Dithering

Dither patterns can be used with STIS spectroscopic modes to do the following:

- Average over pixel-to-pixel flat-field uncertainties.
- Facilitate removal of hot and cold pixels (e.g., by using integer pixel steps).
- Subsample the spatial PSF along the slit (by subpixel steps along the slit).
- Step along the dispersion direction, perpendicular to the spatial axis of the slit.
- Subsample the spectral Line Spread Function (LSF) by stepping a fraction of a pixel along the dispersion direction.
- Map out a two-dimensional region of the sky by using larger step sizes equal to or greater than the slit width.

In first-order spectroscopic modes, improved S/N ratios can be achieved by stepping the target along the slit, taking separate exposures at each location. These separate exposures will subsequently be shifted and added in post-observation data processing. This dithering smooths the detector response over the number of steps, in a manner analogous to that for imaging. For echelle modes, stepping is only possible using the long echelle slit (6 x 0.2 arcseconds). Note that in the high dispersion echelle modes, Doppler shifting due to spacecraft motion will cause the counts from any output pixel to have been sampled at many independent detector pixels in the dispersion direction (for exposures comparable to an orbit visibility period and targets well away from the orbital pole of HST).

In slitless or wide-slit mode, stepping along the dispersion would allow independent solutions for spectrum and flat field, bearing in mind, however, the increased complexity due to the convolution of the spectrum with the spatial structure in the source. This technique is likely to be useful only if the constituent spectra have a good S/N ratio (perhaps 10 or better), so that the shifts between spectra can be accurately determined.

A variation on this technique involves using one of the contingent of fixed pattern, or FP-SPLIT slits. These slits are designed to allow the wavelength projection of the spectrum on the detector to be shifted such that the fixed pattern noise in the flat field and the spectral flux distribution of the target can be computed simultaneously using techniques that have been successfully applied to data taken with GHRS. Note that this approach is likely to work best if the spectra have a good S/N ratio. More detailed information on the use of FP-SPLIT slits is provided in the *STIS Instrument Handbook*.

In many configurations the spectral line FWHM is less than two detector pixels. Possible solutions include stepping the target along the dispersion direction in a wide slit or slit-less aperture to subsample the LSF by displacing the spectrum. This technique can also be used to increase the S/N ratio. To employ this strategy, the observer will have to trade off the benefits of improved sampling with the negative impact of increased wings in the LSF when using a wide slit, particularly for MAMA observations. The use of high-res mode (which is the default) for MAMA observations may provide 15 to 30 percent better sampling, but flat-field variability may make it difficult to realize the benefits, particularly if high S/N ratio spectra are needed.

There are several pre-defined dither patterns that are available for observers to use, these include:

- `STIS-PERP-TO-SLIT:` This is normally used with a spectroscopic slit. It produces a scan along the `POS TARG` X-axis of the aperture; this is used to map a two-dimensional region of the sky. The target is moved perpendicular to the slit along the AXIS1 (dispersion) direction.
- `STIS-ALONG-SLIT:` This is also normally used with a spectroscopic slit. It produces a scan along the `POS TARG` Y-axis of the aperture; this is used to step a target along the long slit to dither bad pixels or improve spatial resolution). The target is moved along the slit in the *AXIS2* (cross-dispersion or spatial) direction.

Additional information about these spectroscopic dither patterns are available in the *STIS Instrument Handbook*.

NICMOS

NICMOS has been decommissioned. The information below is provided as a reference for archival data users.

A wide variety of pre-defined patterns has been created for NICMOS, to allow an easy implementation of both integer-pixel and subpixel dithering. These are generalized extensions to the simple line and box dithers by including spiral and chopping dithers, which are necessary to allow successful removal of a number of NICMOS artifacts. The advantages offered by dithering with NICMOS are the following:

- Post-SAA cosmic ray Persistence: The NICMOS detectors suffer from persistent after-images when exposed to a strong signal. This can arise from astronomical objects, but it also occurs due to cosmic ray bombardment during every passage of HST through the South Atlantic Anomaly (SAA). After SAA passages, a very large fraction of NICMOS pixels glow with a persistent signal that can take up to a few orbits to decay completely. Dithering can help average over the additional noise (really non-Gaussian, spatially correlated signal) that results from SAA-induced persistence. The worst effects of CR persistence can sometimes be removed by the Drizzle and Blot techniques. The NICMOS team has also implemented Post-SAA cosmic ray persistence

removal software and dark observations which ameliorate a substantial amount of the noise induced by traversing the contours. More information on the details of CR persistence removal can be found in the *NICMOS Data Handbook* (McLaughlin & Wiklind 2007).

- Photometric accuracy: the effects of large-scale flat-field variations and of bad-pixels can be controlled via integer-pixel dithering. In addition, for relatively bright objects, dithering can eliminate potential problems of image persistence. Geometric distortion in NICMOS is relatively small, except for the NIC3 camera in its out-of-focus position. We recommend dither steps of ~10 pixels for compact sources. The SPIRAL-DITH pattern can be used to generate dither patterns with 2 positions or more.
- Improved sampling: NIC3, NIC2 (shortward of 1.75 microns) and NIC1 (shortward of 1.0 microns) undersample the image. As in the case of WFPC2, the quality of the image can be improved by subpixel dithering. Most of the information can be recovered via a two-point dither, and virtually all the information can be recovered with four-point dithers. Since NICMOS geometric distortion is relatively small (except for NIC3 when out-of-focus), large dither steps of order ~10 pixels can be used. Telescope pointing errors, which can be of the order of 0.02 arcseconds, may prevent one from obtaining an optimal dither pattern in NIC1 and NIC2, where the uncertainty corresponds to 0.43 NIC1 pixels and 0.27 NIC2 pixels; in this case more than four dither positions are advisable. For NIC3, the telescope pointing uncertainty corresponds to 0.1 pixels shift only, and four dither positions should still be viable for recovering the information. The pre-defined `SPIRAL-DITH` pattern can be effectively used for this purpose. Please see the section on HST pointing accuracy and stability for more information.
- Background removal in uncrowded fields of compact objects: Observations with the NICMOS long wavelength filters (central wavelength longward of 1.7 microns) are affected by variable thermal emission from the telescope (Sosey 2003). To remove this contribution from the images, suitable background observations must be obtained. For compact targets and uncrowded fields, observations of the background can be obtained by dithering the targets across the detector's FOV. The use of the `SPIRAL-DITH` pattern with two or four positions, and a dither step of 10 pixels or more (depending on the size of the targets), may be appropriate for many cases, although the parameters may change according to the nature of the observations. The advantage of dithering in such a case (rather than chopping, for example) is that the target will remain on the chip for all observations, increasing the efficiency of the observation.

Dithering NICMOS observations may also have disadvantages that an observer should consider:

- Cosmic ray removal is not straightforward in pairs of subpixel dithered images. If you plan to use subpixel dithering to improve the image sampling, then `MULTIACCUM` mode or two `ACCUM` mode exposures per position should

be obtained to help cosmic ray removal BEFORE image reconstruction. Some cosmic ray detection and removal is also performed during the calibration of Multiaccum datasets as multiple reads during the exposure allows for statistical elimination of abnormal flux values. In general, the use of `ACCUM` mode is discouraged because there is little on-orbit calibration done for this mode (e.g., dark frames, etc.).

- NICMOS attached parallels: the three NICMOS cameras, NIC1, NIC2, and NIC3, have different plate scales; care should be taken in ensuring that if integer-pixel steps are desired in attached parallel (NIC1+NIC2) observations, the steps are carefully chosen to satisfy this requirement.
- Overheads: The implementation of patterns requires at least 10 - 12 seconds overhead per dither step. Large numbers of dithers can easily add up to minutes taken out of a visibility period for an entire pattern. The trade-off between the advantages offered by dithering, and the diminished amount of observing time should be considered in deciding whether or not to dither.
- Rapid dithering can impose an additional load on the full system in terms of command volume needed to execute the observations, overheads for science data buffer management, and in the volume of data that must be processed through the pipeline. In extreme cases, such as when the overheads required to execute the observations far surpass the actual exposure times, these extra loads can result in lowered overall efficiency of HST observations.

In general, the benefits of dithering greatly outweigh the disadvantages for NICMOS observations. Whenever possible without incurring excessive overhead, we recommend dithering as much as possible when taking NICMOS data. Note, however, that many NICMOS observations are significantly affected by read-out noise, especially for Cameras 1 and 2 and observations shorter than 1.8 micron. Therefore, the effects of read-out noise on multiply-dithered short exposures should always be carefully balanced against the benefits provided by extensive dithering.

WFPC2

WFPC2 has been decommissioned. The information below is provided as a reference for archival data users.

In addition to increasing information on the smallest spatial scales, dithering can be used to reduce the effect of flat-field errors in very deep images. Large dithers (tens of pixels) were used in the HDF for this purpose. Furthermore, dithers greater than one or two pixels can be used effectively to eliminate chip defects such as hot-pixels and bad columns.

The Effect of WFPC2 Geometric Distortion on Dither Offsets

Pixels near the edge of the CCD differ in size on the sky from those near the center. For instance, a shift of (10,10) pixels at detector coordinates (400, 400) corresponds to a shift of about (10.2, 10.2) pixels at location (700, 700).

Default shift values for the DITHER-LINE pattern, in *x* and *y*, are 2.5 pixels for the WF detectors, and 5.5 pixels for the PC detector. For this scenario, over nearly the entire field of view, the difference in offset - even on the PC - is less than 0.1 pixels; the shift is essentially optimal across the whole field.

However, for the standard DITHER-BOX pattern, spacing offsets are as much as 0.75 arcseconds (15.5 PC pixels). This results in a shift difference of about 0.3 pixels in *x* and *y* between detector location (700,700) and the detector center (400,400). While the drizzle software is able to remove geometric distortion, it cannot change the fact that the sampling will not be optimal across the entire field of view. DITHER-BOX pattern default values were chosen to avoid the overlap of detector defects such as bad columns and hot pixels in the final combined image. However, for those willing to risk the possibility that a target area of interest could repeatedly fall on one of several bad chip columns, they could specify smaller offsets to produce a box that would have smaller shift differences across the entire chip, like, for instance, a square 2 x 2 box with sides of 2.5 WF pixels or 5.5 PC pixels.

The Exact Relationship Between POS TARGs and WFPC2 CCD Rows and Columns

For WFPC2 an additional complication is introduced by the fact that the four chips are not precisely aligned with one another, but have small rotational offsets (less than 0.5°) from their nominal alignments. Thus, the POS TARG axes run exactly along the CCD rows and columns on whichever aperture is specified for the observations. For example, if aperture WF3 is specified, the POS TARG axes will run exactly along the rows and columns on WF3, and will run only approximately along the rows and columns of the other CCDs. Note that if WFALL is specified, then the rotation for WF3 is used since WF3 is the reference chip for the WFALL aperture.

The CCD rotation misalignments lead to errors when attempting to dither by certain pixel amounts. For small dithers (less than 0.3 arcseconds) the rotational offsets between the CCDs are unimportant, as they imply pixel registration errors less than 3 milliarcseconds, which is roughly the nominal pointing and guiding stability for HST. But such small dithers do not allow integral pixel stepping simultaneously on the PC as well as the WF chips. A dither of 0.5 arcseconds (5 WFC pixels or 11 PC pixels) gives near-integral stepping on the PC and the WF chips, though the CCD rotations will then introduce registration errors up to 5 mas. An offset of (1.993, 0.000) arcseconds in *x* on WF3 would cause spurious motion in *y* of 0.17 pixel on WF4, due to the rotation.

Two basic types of dither patterns are defined for WFPC2, and are implemented in the Phase II software that is used to process observing programs. These patterns can also be used with non-default spacings when necessitated by very specific types of observations, although in general we recommend that observers use the default spacings which are optimized for a wide variety of scientific programs.

- WFPC2-LINE: the default setting is a two-point dither along a 45° degree diagonal with respect to the pixels of the primary camera, with points spaced by 0.3535 arcsec. This is equivalent to offsets of 2.5 pixels in *x* and *y* for the WF, and approximately 5.5 pixels in *x* and *y* for the PC, providing an offset

with half-pixel increments along both the *x* and *y* axes on all the chips. This pattern produces half-pixel subsampling of the PSF on all the chips, while at the same time including integer-pixel offsets to ameliorate the effects of hot pixels and other chip artifacts.

- `WFPC2-BOX`: a four-point parallelogram dither with points spaced 0.559 arcsec apart that's equivalent to these POS TARG offsets, in arcseconds: (0.0, 0.0) (0.5, 0.25) (0.75, 0.75) (0.25, 0.5). This default setting was designed to improve resolution in the final combined image. This combination of integer-pixel and half-pixel shifts produces complete half-pixel subsampling of the PSF by all 4 quadrants of each pixel. Therefore this strategy is an improvement over the simple 2-point dither which only provides subsampling in two quadrants of each pixel.

APPENDIX D:

Previous Drizzle Software Packages

In this appendix . . .

D.1 Introduction

Since the drizzle algorithm was first introduced in 1996 for the processing of the original Hubble Deep Field observations, it's become the core of several software packages for correcting image distortion, identifying and cleaning cosmic rays, aligning and combining dithered images, to create a clean, distortion-free data product.

Original development of the Drizzle software was a collaborative effort by Andy Fruchter and Richard Hook (http://adsabs.harvard.edu/abs/2002PASP..114..144F). The main code, written in Fortran77, was available as standalone executables, and also as an IRAF task in the STSDAS v2.0 Dither package.

Later, Drizzle[1] software became the basis for a more automated task, called PyDrizzle, that was written in Python. It aligned images based on WCS information to produce distortion-corrected images.

1. The Python drizzle tasks could only be run in the PyRAF environment whereas the original IRAF drizzle tasks could be run from either PyRAF or the original IRAF command language, or ***cl*** environment.

In 2009, an automated Python task called MultiDrizzle was introduced. It implemented steps for identifying cosmic rays in images aligned by PyDrizzle, allowing images to be drizzle-combined with cosmic rays masks to create a final product free of cosmic rays.

D.2 First Generation Drizzle IRAF Tasks

The original Dither package consisted of several separate tasks, executed separately in a specific order, to align observations, as well as detect cosmic rays and other detector defects (see WFPC2 ISR 98-04: The Drizzling Cookbook).

At its heart was the Drizzle task, a set of compiled Fortran code, that was used to combine dithered observations and improve sampling of all the data while using cosmic ray and bad pixel masks created using the other tasks in the dither package. Drizzle relied on the user to provide all necessary inputs for re-sampling the images, including application of distortion coefficients, pixel offsets, rotation or plate scale changes, and the size of the output image. Supplemental tasks were also provided to perform coordinate transformations using the full Drizzle algorithm, instead of going through all the effort of re-sampling images themselves. While these older tasks are no longer supported, their functionality has been incorporated into new scripts used by PyDrizzle, MultiDrizzle, and DrizzlePac.

The following functions are the original IRAF interface routines in the first Dither package, and are no longer supported:

```
avshift - Averages the shifts measured on the 4 WFPC2 chips
crossdriz - Builds a set of cross-correlation images (shift + rotation)
    deriv - Takes the absolute derivative of each image
  driz_cr - Creates cosmic ray masks for dithered data
loop_blot - Runs blot for a given list of input and output images
loop_driz - Runs drizzle for a list of input, output, and mask images
  offsets - Builds cross-correlation image offsets (shift only)
mask_head - Attach mask names to image headers; invert mask if requested
   precor - Cosmic ray removal in images that are used for the crossdriz task
  rotfind - Finds rotation angle from a set of cross-correlation images
shiftfind - Finds X,Y shifts in a cross-corelation image
 dunlearn - Resets all task parameters to default values
    gprep - Transform shifts for one WFPC2 chip into parameters for
            mosaicing all four chips
wfpc2_chips @ parameter set for 'avshift' and 'gprep' tasks
      cdriz @ parameter set for the 'offsets' task
  drizzle - Perform linear image reconstruction by "drizzling" an image
     blot - Image sampling using interpolation (reverse "drizzling")
     tran - Transform coordinates taking into account distortion
 tranback - Transform an X,Y position from output to input pixel position
            using the same conventions as drizzle
    traxy - Transform an X,Y position from input to output pixel position
            using the same conventions as drizzle
   wcs2dr - Converts WCS header information into drizzle parameters for
            image registrationSecond Generation Dither tasks
```

D.3 Second Generation Dither tasks

PyDrizzle provides a semi-automated interface for computing the parameters necessary for running Drizzle. Written in Python, PyDrizzle may be run in either the Python or PyRAF environments, although PyRAF is necessary to have access to the **epar** interface. PyDrizzle performs the task of determining the parameters necessary for aligning images based on the WCS information in the input image headers, as well as any supplemental alignment information provided in shiftfiles. PyDrizzle uses this information to combine the images onto a common WCS. PyDrizzle does not identify cosmic rays, however, it has the ability to ignore pixels flagged as bad, such as pixels identified by other programs as affected by cosmic rays. PyDrizzle only aligns and combines images and still includes parameters for controlling the specifics of how this is accomplished, including exactly how to specify the output product's size, pixel scale, center and orientation. The full set of PyDrizzle parameters can be found in the original *MultiDrizzle Handbook*, along with basic examples for running PyDrizzle in pure Python and/or PyRAF session. For more information on the Python environment see the Python webpage.

D.4 WCS-enabled Dither Tasks

Instead of explicitly providing all of the alignment information, the input image's WCS information can be used in conjunction with the distortion coefficients to place the image in the output frame. This can be done using WCS-enabled versions of the basic **drizzle** and **blot** tasks, as well as the coordinate transformation tasks **traxy** and **tranback**. These tasks allow the user to simply specify the output WCS information or use the WCS information of an existing drizzled image to define the output frame. The WCS keywords from the input image will then be fit to the WCS derived from the output frame to determine how to place the image in the output image without requiring the user to explicitly provide offsets, rotations and scale changes as required by the original Drizzle task.

D.5 WCS-enabled Drizzle

The task **wdrizzle** provides the user with the ability to use the WCS information to apply the Drizzle algorithm to an input image and place that corrected image in an output frame specified through the use of WCS information. The full parameter set for **wdrizzle** can be found in the tasks help file under PyRAF or IRAF. If the output image already exists and the ***geomode*** parameter has been set to ***wcs***, then the WCS information from that image will be used in place of the WCS parameter values specified for ***outscl***, ***raref***, ***decref***, ***xrefpix***, ***yrefpix***, and ***orient***.

D.6 Coordinate Transformation Tasks

Several supplemental tasks have also been developed using the core Drizzle Fortran code base, primarily to provide the ability to apply the Drizzle transformations to coordinates rather than to whole images. This capability allows the user to measure an object of interest in one frame and find its exact position in the other frame, making it possible to eliminate the effects of re-sampling the PSF. The tasks **traxy** and **tranback** require the user to provide the full set of transformation parameters as they would be given to **drizzle** or **blot** respectively. These tasks also have WCS-enabled versions; namely, **wtraxy** and **wtranback**. These WCS enabled versions accept the same inputs as **wdrizzle** or **wblot** to set up the parameters for the transformation, and will use the WCS from an existing image to compute the results. The task **tran**, on the other hand, asks for the original input image as well as the output drizzled image as created by any of the **drizzle** tasks and a direction for the transformation. It then reads the **drizzle** parameters written to the header of the output image to set up the transformation, and will automatically call either **tran** or **tranback** depending on the direction of the transformation requested by the user. All of these tasks support operations on either a single *X,Y* position or on a list of *X,Y* positions provided as an ASCII file with a column of *X* positions and a column of *Y* positions.

The second generation Drizzle tasks focused on automating the use of the original separate IRAF tasks for creating a final drizzle-combined product that was free of cosmic rays. Most of the task development was done in Python. These tasks used PyRAF to provide an IRAF **epar** interface to simulate the run environment of native IRAF tasks. These tasks can only be run under PyRAF or directly under Python. They included:

```
multidrizzle - Automated Python task for performing cosmic ray rejection
   pydrizzle - Automated Python-based interface for the "drizzle" task
 tweakshifts - Compute residual shifts between images
     xytosky - Translate a 2-D image pixel coordinate to right ascension
               and declination, optionally applying the distortion coefficients
               for the detector.
      wtraxy - Transform an X,Y position from input to output pixel position
               using the same conventions as drizzle, with WCS and list support
  wtranback - Transform an X,Y position from output to input pixel position using
               the same conventions as drizzle, with WCS and list input support
       wblot - Image sampling using Interpolation; reverse Drizzle version with
               WCS support
```

These Python tasks use the same core Fortran77 codebase used by the original native IRAF tasks for applying the Drizzle algorithm to the images.

D.7 Supplemental tasks

The final set of tasks addressed operations that are generally useful, but not directly involved in image alignment or coordinate transformations. These were supporting IRAF tasks which may still aid the user in preparing and analyzing their datasets:

```
cgsky - Compute sky using ‘gsky’ plus a correction for histogram truncation.
     gsky - Compute sky using the same mode algorithm used in task "crrej".
imextreme - Locates the maximum and minimum pixels in an image.
      sky - Sky processing by "crrej-like" algorithm.
       dq @ parameter set for the ’sky’ task.
```

These tasks, however, have been written as native IRAF tasks and have been known to have problems with some datasets, while improving the results in other datasets. Note that the use of these non-supported functions may result in the creation of inferior data products, as the most current updates to the code and methodology are only implemented in the Python functions.

References

Papers and Instrument Science Reports

Anderson, J., King, I. R., 2000, “Toward High-Precision Astrometry with WFPC2. I. Deriving an Accurate Point-Spread Function”, PASP 112, 1360

Anderson, J.,”Astrometry with the Advanced Camera: PSFs and Distortion in the WFC and HRC”, 2002 HST Calibration Workshop, eds. Arribas, S., Koekemoer, A., Whitmore, B.

Anderson, J., 2007, “Variation of the Distortion Solution of the WFC”, ACS-ISR 07-08

Arendt, R. G., Fixsen, D. J., Moseley, S. H., 2000, “Dithering Strategies for Efficient Self-Calibration of Imaging Arrays”, ApJ 536, 500

Beckwith, S., et al, 2006, “The Hubble Ultra Deep Field”, AJ, V132, Issue 5, pp. 1729-1755

Bertin, E. & Arnouts, S. 1996: “SExtractor: Software for Source Extraction”, Astronomy & Astrophysics Supplement 317, 393

Biretta, J., “Dithering: Relationship Between POS TARGs and CCD Rows/Columns”, (WFPC2 Website)

Biretta, J., Wiggs, M., “General Advice on Dithering HST Observations”, (WFPC2 Website)

Brown, T., Hartig, G., Baggett, S., 2008, “WFC3 TV3 Testing: UVIS Window Contamination”, WFC3-ISR-2008-10

Bowers, C., Baum, S., 1998, “Plate Scales, Anamorphic Magnification and Dispersion: CCD Modes”, STIS-ISR-98-23

Casertano, S., Wiggs, M., 2001, “An Improved Geometric Solution for WFPC2”, WFPC2-ISR-2001-10

Casertano, S., de Mello, D., Dickinson, M., Ferguson, H. C., Fruchter, A. S., Gonzalez-Lopezlira, R., Heyer, Inge; Hook, R. N., Levay, Z.; Lucas, R. A., Mack, J., Makidon, R. B.; Mutchler, M., Smith, T. E., Stiavelli, M., Wiggs, M. S., Williams, R. E., 2000, "WFPC2 Observations of the Hubble Deep Field South", AJ 120, 2747

Cox, C., 1994, "The WFPC2 Scales and Alignments", SOB-ISR-94-10-21

Cox, C., and Gilliland, R. L., "The Effect of Velocity Aberration Correction on ACS Image Processing proceedings", 2002 HST Calibration Workshop.

Cox, C., Ritchie, C., Bergeron, E., Mackenty, J., Noll, K., 1997, "NICMOS Distortion Correction", OSG-ISR-CAL-97-07

Dencheva, N., Hack, W., Fruchter, A., Greenfield, P., "Distortion Correction in HST FITS Files - DRAFT", 2010

Fruchter, A., et al, "BetaDrizzle: A Redesign of the MultiDrizzle Package", 2010 STScI Calibration Workshop, eds. Deustua, S., Oliveira, C.

Fruchter, A. S., Hook, R. N. 1997, "A Novel Image Reconstruction Method Applied to Deep Hubble Space Telescope Images", in Applications of Digital Image Processing XX, Proc. SPIE, Vol. 3164, A. Tescher, editor, p.120.

Fruchter, A. S., Hook, R. N., 2001, "Drizzle: A Method for the Linear Reconstruction of Undersampled Images", 2002, PASP 114:144-152

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, "A Package for the Reduction of Dithered Undersampled Images", 1997 HST Calibration Workshop, ed. S. Casertano et al.

Fruchter, A. S., Mutchler, M., "Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration", 1997 HST Calibration Workshop, ed. Casertano, S, et al.

Gardner, J. P., Baum, S. A., Brown, T. M., Carollo, C. M., Christensen, J., Dashevsky, I., Dickinson, M. E.; Espey, B. R., Ferguson, H. C., Fruchter, A. S., Gonnella, A. M., Gonzalez-Lopezlira, R. A., Hook, R. N., Kaiser, M. E., Martin, C. L., Sahu, K. C., Savaglio, S., Smith, T. E, Teplitz, H. I.; Williams, R. E., Wilson, J., 2000, "The Hubble Deep Field South: STIS Imaging", AJ 119, 486

Gilliland, R. L., "Guiding Errors in 3-Gyro: Experience from WF/PC, WFPC2, STIS, NICMOS, and ACS", ISR-TEL-2005-02

Greisen, E. W., nd Calabretta, M. R., "Representations of World Coordinates in FITS (FITS Paper I)", 2002, Astronomy & Astrophysics, 395, 1061-1075

Gull, S.F. and Daniel, G.J., 1978, “Image reconstruction from incomplete and noisy data”, Nature 272:686–690

Gilliland, R. L. Gilliland, Brown, T. M., Guhathakurta, P., Sarajedini, A., Milone, E. F., Albrow, M. D., Baliber, N. R., Bruntt, H., Burrows, A., Charbonneau, D., Choi, P., Cochran, W. D., Edmonds, P. D., Frandsen, S., Howell, J. H., Lin, D. N. C., Marcy, G. W., Mayor, M., Naef, D., Sigurdsson, S., Stagg, C. R., Vandenberg, D. A., Vogt, S. S., and Williams, M. D., 2000, “A Lack of Planets in 47 Tucanae from a Hubble Space Telescope Search”, ApJ 545, L47

Gilmozzi, R., Ewald, S., Kinney, E., 1995, “The Geometric Distortion Correction for the WFPC Cameras”, WFPC2-ISR-95-02

Gonzaga, S., Biretta, J., Wiggs, M., et al., 1998, “The Drizzling Cookbook”, WFPC2-ISR-98-04

Hack, W., Cox, C., “Geometric Distortion Table: IDCTAB”, ACS ISR 2000-11

Hack, W., Cox, C., 2001, “Revised IDCTAB Definition: Application to HST Data”, ACS-ISR-2001-008

Hartig, G., Kinney, E., Hodge, P., Lallo, M., Downes, R., 1999, “STIS Coordinate System Orientation and Transformations”, TIR STIS 99-02 (Available upon request.)

Heyer, I., 2001, “The WFPC2 Photometric CTE Monitor”, WFPC2-ISR-2001-09

Holtzman, J., et al., 1995, “The Performance and Calibration of WFPC2 on the Hubble Space Telescope”, PASP 107, 156

Kalirai, et al., “A Deep, Wide-field, and Panchromatic View of 47 Tuc and the SMC with HST: Observations and Data Analysis Methods”, 2012, AJ, 143, 11.

Koekemoer, A. M., Fruchter, A. S., Hook, R. N., Hack, W., 2002, “MultiDrizzle: An Integrated PyRAF Script for Registering, Cleaning and Combining Images”, 2002 HST Calibration Workshop (eds. Arribas, S., Koekemoer, A. M., Whitmore, B.; Baltimore: STScI), 337

Koekemoer, A. M., Grogin, N. A., Schreier, E. J., Giacconi, R., Gilli, R., Kewley, L., Norman, C., Zirm, A., Bergeron, J., Rosati, P., Hasinger, G., Tozzi, P., Marconi, A., 2002, “HST Imaging in the Chandra Deep Field South: II. WFPC2 Observations of an X-Ray Flux-Limited Sample from the 1 Million Second Chandra Catalog”, ApJ 567, 657

Kozhurina-Platais, V., et al., “WFC3/UVIS and IR Time Dependency of Linear Geometric Distortion over Cycles 17 & 18”, WFC3 ISR 2012-03

Kozhurina-Platais, V., et al., “Multi-Wavelength Geometric Distortion Solution for WFC3/UVIS and IR”, 2010, “HST 2010 Calibration Workshop”, pg 277

Kozhurina-Platais, V., et al., “WFC3 SMOV Proposal 11444 - UVIS Geometric Distortion Calibration”, WFC3 ISR 2009-33

Kozhurina-Platais, V., et al., “WFC3 SMOV Proposal 11445 - IR Geometric Distortion Calibration”, WFC3 ISR 2009-34

Kozhurina-Platais, V., Anderson, J., Koekemoer, A. M., 2003, “Toward a Multi-Wavelength Geometric Distortion Solution for WFPC2”, WFPC2-ISR-2003-02

Lauer, T. 1999, “Combining Undersampled Dithered Images”, PASP 111, 227

Lauer, T. 1999, “The Photometry of Undersampled Point-Spread Functions”, PASP 111, 1434

Lucy, L. B., & Hook, R. N., “Co-adding Images with different PSF’s”, 1991, Astronomical Data Analysis Software and Systems I, A.S.P. Conf. Ser., Vol. 25, eds. D. M. Worrall, C. Biemesderfer, & J. Barnes (San Francisco: ASP), 277

Lucy, L. B., “An iterative technique for the rectification of observed distributions”, 1974, Astronomical Journal, Vol. 79, p. 745

Malumuth, E. M., Bowers, C. W., 1997, “Determination of Geometric Distortion in STIS Images”, in 1997 HST Calibration Workshop, ed. S. Casertano, et al.

Mutchler, M., Cox, C., 2001, “ACS Dither and Mosaic Pointing Patterns”, ACS-ISR-2001-07

Mutchler, M., Fruchter, A. S., 1997, “Drizzling Dithered WFPC2 Images - A Demonstration”, in 1997 HST Calibration Workshop, ed. S. Casertano, et al.

Sahu, K. C., Casertano, S., Livio, M., Gilliland, R. L., Panagia, N., Albrow, M. D., Potter, M., 2001, “Gravitational Microlensing by Low-Mass Objects in the Globular Cluster M22”, Nature 411, 1022

Schreier, E. J., Koekemoer, A. M., Grogin, N. A., Giacconi, R., Gilli, R., Kewley, L., Norman, C., Hasinger, G., Rosati, P., Marconi, A., Salvati, M., Tozzi, P., 2001, “Hubble Space Telescope Imaging in the Chandra Deep Field-South. I. Multiple Active Galactic Nucleus Populations”, ApJ 560, 127

Richardson, W.H., 1972, “Bayesian-Based Iterative Method of Image Restoration”, JOSA, Vol. 62, Issue 1, pp. 55-59

Rots, A. H., Bunclark, P. S., Calabretta, M. R., Allen, S. L., Manchester, R. N., Thompson, W. T., "Representations of Time Coordinates in FITS" 2012, (Draft, FITS Paper IV)

Shupe, D.L., et al., "The SIP Convention for Representing Distortion in FITS Image Headers", 2005, ADASS XIV, ASP Conference Series, Vol. 347, p. 491

Sosey, M., Wheeler, T., Sivaramakrishnan, A., 2003, "Analysis of HST Thermal Background as Seen by NICMOS+NCS", NICMOS-ISR-2003-007

Sparks, W. B., Hack, W., Clampin, M., 2001, "ACS Software Tool Development", ACS-ISR-2001-10

Sparks, W. B., Hack, W., Hook, R. N., 2001, "Initial Implementation Strategy for Drizzle with ACS", ACS-ISR-2001-04

Stetson, P. B., "DAOPHOT - A computer program for crowded-field stellar photometry", Astronomical Society of the Pacific, Publications, (ISSN 0004-6280), vol. 99, March 1987, p. 191-222.

Trauger, J. T., Vaughan, A. H., Evans, R. W., Moody, D. C., 1995, "Geometry of the WFPC2 Focal Plane" in Calibrating HST: Post Servicing Mission, eds. A. Koratkar & C. Leitherer (Also available upon request)

Walsh, J. R., Goudfrooij, P., Malumuth, E. M., 2001, "STIS Geometric Distortion - SMOV3A Tests for CCD, NUV-MAMA and FUV-MAMA", STIS-ISR-2001-02

Weir N., Djorgovski S., Piotto G., "A Maximum Entropy Deconvolution Approach to Photometry in Globular Clusters", 1990, Bull. Am. Astron. Soc., 22, 1290

Whitmore, B., Heyer, I., 1997, "New Results on CTE and Constraints on Flat-Field Accuracy", WFPC2-ISR-97-08

Documents

Bostroem, K. and Proffitt, C., 2011, "STIS Data Handbook", Version 6.0, (Baltimore: STScI)

Fruchter, A. and Sosey, M. et al. 2009, "The MultiDrizzle Handbook", version 3.0, (Baltimore, STScI)

Gonzaga, S., & Biretta, J., et al. 2010, "WFPC2 Data Handbook", v. 5.0, ed., Baltimore, STScI

Gonzaga, S., et al., 2011, "ACS Data Handbook", Version 6.0, (Baltimore: STScI).

Koekemoer, A. M., et al. 2002, "HST Dither Handbook", Version 2.0 (Baltimore: STScI)

Rajan, A. et al. 2010, "WFC3 Data Handbook", Version 2.1, (Baltimore: STScI)

Smith, E., et al. 2011, "Introduction to the HST Data Handbooks", Version 8.0, (Baltimore: STScI)

Thatte, D. and Dahlen, T. et al. 2009, "NICMOS Data Handbook", version 8.0, (Baltimore, STScI)

Websites

Instrument Pages

WFC3
http://www.stsci.edu/hst/wfc3

ACS
http://www.stsci.edu/hst/acs

STIS
http://www.stsci.edu/hst/stis

NICMOS
http://www.stsci.edu/hst/nicmos

WFPC2
http://www.stsci.edu/hst/wfpc2

Data Analysis

SAOImage ds9
http://hea-www.harvard.edu/RD/ds9/

DrizzlePac
http://drizzlepac.stsci.edu

WCSLIB
http://www.atnf.csiro.au/people/mcalabre/WCS/wcslib/index.html

STSDAS

STSDAS
http://www.stsci.edu/institute/software_hardware/stsdas

PyFITS
http://www.stsci.edu/institute/software_hardware/pyfits

PyRAF
http://www.stsci.edu/institute/software_hardware/pyraf/localProductDescription

STScI_Python
http://www.stsci.edu/institute/software_hardware/pyraf/stsci_python

stwcs
http://stsdas.stsci.edu/stsci_python_epydoc_2.12/docs/stwcs/index.html

Reference Files

ACS Reference Files webpage
http://www.stsci.edu/hst/observatory/cdbs/SIfileInfo/ACS/reftablequeryindex

WFC3 Reference Files Webpage
http://www.stsci.edu/hst/observatory/cdbs/SIfileInfo/WFC3/reftablequeryindex

WFPC2 Reference FIles Webpage
http://www.stsci.edu/hst/wfpc2/analysis/calfiles

STIS Reference Files Webpage
http://www.stsci.edu/hst/observatory/cdbs/SIfileInfo/STIS/reftablequeryindex/

NICMOS Reference Files Webpage
http://www.stsci.edu/hst/observatory/cdbs/SIfileInfo/NICMOS/reftablequeryindex

Observing Campaigns

Hubble Deep Field
http://www.stsci.edu/ftp/science/hdf/hdf.html

Hubble Ultra Deep Field
http://www.stsci.edu/hst/udf

Hubble Deep Field South - Data Products
http://www.stsci.edu/ftp/science/hdfsouth/dataprod.html

Index