

1. The DrizzlePac Handbook	2
1.1 Change Log	5
1.2 Chapter 1: Introduction to AstroDrizzle and DrizzlePac	6
1.2.1 1.1 Introduction	7
1.2.2 1.2 What is DrizzlePac	8
1.2.3 1.3 DrizzlePac Code	10
1.2.4 1.4 Data from the MAST Archive	13
1.3 Chapter 2: Observational Dithering Options for Drizzling Data	16
1.3.1 2.1 Dithering Strategies	17
1.3.2 2.2 Selecting the Right Dither Strategy	19
1.4 Chapter 3: Description of the Drizzle Algorithm	31
1.4.1 3.1 Image Reconstruction and Restoration Technique	32
1.4.2 3.2 Drizzle Concept	37
1.4.3 3.3 Weight Maps and Correlated Noise	39
1.4.4 3.4 Characteristics of Drizzled Data	44
1.5 Chapter 4: Astrometric Information in the Header	50
1.5.1 4.1 Introduction	51
1.5.2 4.2 How Distortions are Represented in AstroDrizzle	52
1.5.3 4.3 Distortion Information in Pipeline Calibrated Images	58
1.5.4 4.4 HST Pointing Accuracy and Stability	64
1.5.5 4.5 Absolute Astrometry	69
1.5.6 4.6 Using Headerlets	76
1.6 Chapter 5: DrizzlePac Software Package	94
1.6.1 5.1 DrizzlePac: An Overview	95
1.6.2 5.2 AstroDrizzle The New Drizzle Workhorse	96
1.6.3 5.3 AstroDrizzle in the Pipeline	105
1.6.4 5.4 The DrizzlePac Package	107
1.6.5 5.5 Configuration Files (cfg)	117
1.7 Chapter 6: Reprocessing with the DrizzlePac Package	120
1.7.1 6.1 Beyond the Standard Calibration Pipeline	121
1.7.2 6.2 Image Alignment	122
1.7.3 6.3 Running AstroDrizzle	127
1.8 Chapter 7: Data Quality Checks and Trouble Shooting Problems	133
1.8.1 7.1 Inspecting the Drizzled Products from MAST	134
1.8.2 7.2 Verifying TweakReg Solutions After User Reprocessing	140
1.8.3 7.3 Inspecting Drizzled Products after User Reprocessing	145
1.9 Chapter 8: DrizzlePac Examples	148
1.9.1 8.1 Jupyter Notebook Introduction	149
1.9.2 8.2 Practical Tutorials	150

The DrizzlePac Handbook

Version 3.0 - Jan 2025

[PDF version](#)

DrizzlePac Handbook

User Support

- Please contact the HST Help Desk for assistance. We encourage users to access the new web portal where you can submit your questions directly to the appropriate team of experts.
 - **Website:** <http://hsthhelp.stsci.edu>

Additional Resources

Information and other resources are available from the [STScI DrizzlePac website](#).

For installation help, coding examples, and additional documentation, please visit the [Drizzlepac Jupyter Notebooks on GitHub](#).

Revision History

Document	Version	Date	Editor
The DrizzlePac Handbook	3.0	January 2025	Anand, G.S., Mack, J., Avila, R.J., Bajaj, V., Kuhn, B., Revalski, M., and Som, D. Susan Rose-Technical Editor
The DrizzlePac Handbook	2.0	February 2021	Hoffmann, S. L., Mack, J., Avila, R. J., Martlin, C., Bajaj, V., and Cohen, Y. Susan Rose-Technical Editor
The DrizzlePac Handbook	1.0	June 2012	Gonzaga, S., Hack W., Fruchter, A., and Mack, J. <i>et al.</i> 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., <i>et al.</i>

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 Gagandeep Anand, Amber Armstrong, Roberto Avila, Varun Bajaj, Mihai Cara, Yotam Cohen, Nadia Dencheva, Tyler Desjardins, Michael Dulude, Andy Fruchter, Shireen Gonzaga, Warren Hack, Samantha Hoffmann, Benjamin Kuhn, Kevin Lindsay, Ray Lucas, Jennifer Mack, John Mackenty, Catherine Martlin, Larry Petro, Vera Kozhurina-Platais, Abhijit Rajan, Mitchell Revalski, Linda Smith, Chris Sontag, and Leonardo Ubeda.

Citation

In publications, refer to this document as:

- Anand, G. S., Mack, J., et al., 2025, "The DrizzlePac Handbook", Version 3.0, (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.

Acknowledgements

Information in this manual 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, Mihai Cara, Nadia Dencheva, Michael Droettboom, Richard Hook (ESO), Chris Sontag, and Megan Sosey.

Under the leadership of ACS and WFC3 team leads at the time, 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 provided the technical expertise for publication of this document.

Change Log

Changes made to this document after version 3.0 was published are logged here.

Chapter 1: Introduction to AstroDrizzle and DrizzlePac

Chapter Contents

- [1.1 Introduction](#)
- [1.2 What is DrizzlePac](#)
- [1.3 DrizzlePac Code](#)
- [1.4 Data from the MAST Archive](#)


A color composite of drizzled NGC 6503 images.




1.1 Introduction

This *DrizzlePac Handbook* is written for both novice users and seasoned "drizzlers." Users new to *HST* data processing and analysis are recommended to familiarize themselves with *HST* data formats and install the newest version of DrizzlePac with Python 3 from PyPI (`pip install drizzlepac`) or via [stenv](#). The latest installation instructions for [stenv](#) can be found [here](#).

The [HST Data Handbook Introduction](#) is a good place to start for information about *HST* data, and instructions for getting ready to use DrizzlePac are provided [here](#). For work on a specific instrument, a review of the [instrument's data handbook](#) is recommended.

 IRAF/PyRAF has been deprecated and is not supported. Please use the latest version of DrizzlePac with Python 3.

 All the code in this document is in Python 3. For those who are switching from PyRAF/IRAF, [this newsletter from 2018](#) explains the motivation behind this transition and contains resources on how best to make the change, including information about [the STAK tutorial documentation](#) which provides Python alternatives to many commonly used PyRAF/IRAF tools.

1.2 What is DrizzlePac

DrizzlePac is a suite of tasks for aligning, distortion-correcting, cosmic-ray cleaning, and combining *HST* images. It also includes several other tasks, such as a task for applying distortions to images, and tasks for transforming sky coordinates to image coordinates and vice-versa.

The Drizzle algorithm ([Fruchter and Hook, 2002](#)) was developed as a powerful method for combining [dithered *HST* images](#), i.e., images (exposures) observed with different pointings (offsets). 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 to 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* images. 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 *CR* (cosmic rays) and bad pixels. In some situations, such as the case of appropriately chosen dither patterns, users were able to regain some of the resolution lost in the original undersampled images. Improved DrizzlePac software replaced the old MultiDrizzle software in the *HST* pipeline in June 2012.

The initial concept of the DrizzlePac package which replaced the original MultiDrizzle package 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. DrizzlePac continues to perform the same functions as MultiDrizzle. It significantly improves the processing of the data, and particularly the astrometric information from the image header. These features will be covered in greater detail in the subsequent chapters.

The DrizzlePac software package includes the following tasks:

- [AstroDrizzle](#) to sky-subtract, CR clean, and co-register multiple distorted images onto a single distortion-corrected frame;
- [TweakReg](#) to perform image alignment by finding the offsets and rotation between *HST* images, and computing a corrected WCS (World Coordinate System) from these offsets and rotations;
- [SkyMatch](#) to subtract (or equalize) sky background in *HST* images;
- [Runastrodriz](#) to control the operation of **AstroDrizzle**.
- [Tweakback](#) to resample and apply distortions to undistorted images (it is a reverse of **AstroDrizzle**);
- [Pixtopix](#) to transform image coordinates of a source in one image to image coordinates in another image using full WCS information;
- [Skytopix](#) transforms sky coordinates to X,Y pixel positions. A reverse transformation can be done using the task [pixtosky](#);
- [MapReg](#) to read and convert DS9 region files based on the WCS information in the image header;
- [PhotEq](#) to equalize the data values of chips within an image so they can all be analyzed in the same way, which is particularly useful for WFPC2 and WFC3;
- [PixReplace](#) to replace certain pixel values with another value;
- [ResetBits](#) to change specific values in the data quality arrays to zero;
- [UpdateNpol](#) to update ACS image headers with new NPOL and D2IM reference filenames;
- [ImageFindPars](#) to set the parameters used by the internal DrizzlePac source finding algorithm, which is based on the DAOFIND algorithms ([Stetson 1987](#));

- [ReflmageFindPars](#) to set the parameters in a similar manner to `imagefindpars`, but only applied to source detection in the reference images.

1.3 DrizzlePac Code

[Code Improvements](#)
[Geometric Distortion Corrections](#)
[A Fundamentally Different Approach to Handling Astrometry](#)

Code Improvements

DrizzlePac maintained much of the same Drizzle algorithm since the beginning, but over time, this 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 with the command line and/or using the TEAL (Task Editor and Launcher) Graphical User Interface (GUI), although the latter is no longer supported due to compatibility issues with some operating systems. While TEAL is referenced at times in this text, a command line alternative is always provided.

A full list of changes can be found in the [DrizzlePac release notes](#).

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

Geometric Distortion Corrections

One of the main functions of **AstroDrizzle** is correcting geometric distortion due to the optical distortion and various manufacturing processes. DrizzlePac incorporates the distortion corrections directly into the WCS in `flt.fits` headers, using the *Simple Image Polynomial (SIP)* convention ([Shupe et. al, 2005](#)) as well as non-polynomial distortion look-up tables. This convention has been used for describing the geometry of *Spitzer Space Telescope* images. Representing image distortion corrections using the SIP convention improves the handling of image combination and WCS astrometric information.

These corrections are unique for different *HST* instruments and stored in these reference files:

- Geometric distortion due to the optical distortion is expressed as a set of high-order polynomial coefficients stored in a reference file called the *IDCTAB (Instrument Distortion Coefficients TABLE)*;
- ACS/WFC requires correction for the pixel-grid irregularities due to the manufacturing process. It is a 2-D look up table which corrects X,Y raw positions before the geometric distortion correction. This look-up table is a reference file which is called D2IMFILE;

- WFC3/UVIS requires correction for the lithographic-mask pattern correction due to the manufacturing process. It is a 2-D look-up table which corrects X, Y raw positions before the geometric distortion. This look-up table is also a reference file called D2IMFILE;
- In addition to the pixel-grid irregularities, the filter-dependent component of the total distortion model is also described using 2-D look-up tables. These tables are unique for each set of ACS/WFC and/or WFC3/UVIS filters and are provided by the NPOLFILE reference files. These filter-dependent distortions correct X, Y positions after correction for D2IMFILE and simultaneously with IDCTAB corrections.

The D2IMFILE and NPOLFILE reference files are images with 4 extensions and 32×64 images, in each X and Y direction for each ACS/WFC or WFC3/UVIS chip, interpolating a set of 32×64 -entry tables representing residual distortion corrections into a full-size image. The non-polynomial distortion corrections, in tabular form, are inserted directly in the header of `flt.fits` or `flc.fits` files as FITS extensions. As a result, ACS/WFC and WFC3/UVIS data have several FITS extensions containing information on the geometry of the detector not described by the image SIP coefficients, the *D2IMARR* and *WCSDVARR* FITS extensions.



`flt.fits` or `flc.fits` images with this format cannot be run with old software as Multidrizzle still requires the IDCTAB and DGEOFILE reference files. Likewise, older `flt.fits` or `flc.fits` images will not be compatible with the current DrizzlePac until the WCS in the headers are updated with this format.

A Fundamentally Different Approach to Handling Astrometry

As of December 3, 2019, `flt.fits` or `flc.fits` images have new absolute astrometric solutions. These improvements mainly reduce the pointing errors (generally a few tenths of an arcsecond), but do NOT affect the distortion solution. The solutions are derived in a number of ways, but fall into two different categories. The first is an "a Priori" solution, which is derived using Gaia Data Release 1 (DR1) coordinates for the guide stars used in the observation, as the Gaia positions for these stars are much more accurate than the previous Guide Star Catalog positions. The other type of solution is an "a Posteriori" solution derived via matching sources detected in an image to an external catalog and correcting the image for the offsets between matches. If one of these solutions exists, the external catalog will either be the Hubble Source Catalog Version 3, Gaia DR1, or Gaia DR2 (the final choice depends on the quality of solution each catalog was able to generate for any given dataset). Other catalogs may be added in the future.

If a new solution is available for a dataset, products retrieved from the MAST archive will have the best solution applied by default. In this case, the application of the solution changes the World Coordinate System (WCS) in the headers of the `flt.fits` or `flc.fits` files, but does NOT affect the pixel values for these data (though the values in `drz.fits` or `drc.fits` images will likely be affected). This is because the WCS defines transformations from pixel to sky coordinates for an image. When a new WCS is applied, this is reflected in the WCSNAME header keyword in the science extensions of the `flt.fits` or `flc.fits` files. Several solutions are available for a given dataset, and are contained in extra "headerlet" extensions appended to the end of `flt.fits` or `flc.fits` files. A detailed explanation of this approach can be found in [Chapter 4: Astrometric Information in the Header](#), or at [the Hubble Advanced Products astrometry webpage](#).

In general, the absolute astrometry of the data products should be significantly improved when new solutions are present (especially with the a Posteriori solutions), i.e. the WCS of the updated images should be very close, if not matching the Gaia frame. However, as with the drizzle products from the archive, a broad set of parameters were used in the derivation of the a Posteriori solutions. Rarely, this may cause a degradation in both the absolute and relative astrometry, especially in sparse/small fields with very few Gaia sources. If the relative astrometry between input images of a drizzled image is compromised, then the resulting drizzled image may have spurious rejection of sources as cosmic rays. Furthermore, since the derivation of the a Posteriori solutions is done on an individual dataset basis, it is possible that images taken in the same visit, but part of different associations (such as those taken in different filters) may not be aligned to each other. Thus, it is strongly recommended to assess datasets visually to verify alignment, e.g. by opening up images in SAOImage DS9, matching the WCS's of the images and blinking them. As always, drizzled images directly from the archive should be treated as quick look/preview images only and should not be assumed to be science-ready.

1.4 Data from the MAST Archive

All `flt.fits` or `flc.fits` images downloaded from the MAST archive contain many extensions with useful data. The formatting of a single WFC3/UVIS image is shown here as an example.

```
--> from astropy.io import fits
--> fits.info('icb701osq_flc.fits')
```

Filename: icb701osq_flc.fits						
No.	Name	Ver	Type	Cards	Dimensions	Format
0	Primary	1	PrimaryHDU	305	()	
1	SCI	1	ImageHDU	224	(4096, 2051)	float32
2	ERR	1	ImageHDU	51	(4096, 2051)	float32
3	DQ	1	ImageHDU	43	(4096, 2051)	int16
4	SCI	2	ImageHDU	220	(4096, 2051)	float32
5	ERR	2	ImageHDU	51	(4096, 2051)	float32
6	DQ	2	ImageHDU	43	(4096, 2051)	int16
7	HDRLET	1	NonstandardExtHDU	18	(8640,)	
8	HDRLET	2	NonstandardExtHDU	26	(112320,)	
9	HDRLET	3	NonstandardExtHDU	26	(112320,)	
10	HDRLET	4	NonstandardExtHDU	26	(112320,)	
11	HDRLET	5	NonstandardExtHDU	26	(112320,)	
12	WCSCORR	1	BinTableHDU	59	14R x 24C	[40A, I, A, 24A, 24A, 24A, 24A, D, D, D, D, D, D, D, 24A, 24A, D, D, D, D, J, 40A, 128A]
13	WCSDVARR	1	ImageHDU	15	(64, 32)	float32
14	WCSDVARR	2	ImageHDU	15	(64, 32)	float32
15	D2IMARR	1	ImageHDU	15	(64, 32)	float32
16	D2IMARR	2	ImageHDU	15	(64, 32)	float32
17	WCSDVARR	3	ImageHDU	15	(64, 32)	float32
18	WCSDVARR	4	ImageHDU	15	(64, 32)	float32
19	D2IMARR	3	ImageHDU	15	(64, 32)	float32
20	D2IMARR	4	ImageHDU	15	(64, 32)	float32
21	HDRLET	6	NonstandardExtHDU	26	(112320,)	
22	HDRLET	7	NonstandardExtHDU	26	(112320,)	
23	HDRLET	8	NonstandardExtHDU	26	(112320,)	

For ACS/WFC and WFC3/UVIS, the first seven extensions ([0] thru [6]) will always remain in this order:

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

Extensions [7] through [23] contain the distortion correction information beyond that captured in the world coordinate system (WCS) and the headerlets noted in [Section 1.3.3](#). The number and order of these extensions are subject to change depending on the amount and placement of the headerlets. While the extensions of other images might look slightly different, they will be made up of these four types:

- The D2IMARR extension in [15], [16], [19], and [20] are a two-dimensional vector containing corrections for physical distortions in the detectors from pixel grid irregularities due to the manufacturing process.
- The WCSDVARR extension in [13], [14], [17], and [18] describe distortion corrections not modeled by the polynomial fit which are mostly due to hard-to-model local distortions introduced by filters. For ACS/WFC and WFC3/UVIS these are four 64×32 images.
- The WCSCORR extension is a binary table that records the history of changes to the WCS in the header by **Tweakreg**, where each line records a single update to a single chip.
- Each HDRLET extension contains independent WCS solutions available for the data.

Other instruments, such as WFPC2 and WFC3/IR, may have different numbers of extensions for the data, error, and DQ arrays depending on the number of detectors for that instrument.

Please see [Chapter 4](#) for more details. Most importantly, the image data in the `flt.fits` or `flc.fits` from MAST is not modified irregardless of any changes to the formatting of the extensions, although the improvements to the WCS can cause differences in the drizzle products.

Chapter 2: Observational Dithering Options for Drizzling Data

Chapter Contents

- [2.1 Dithering Strategies](#)
- [2.2 Selecting the Right Dither Strategy](#)

2.1 Dithering Strategies

[What is Dithering?](#)

[Benefits of Dithering](#)

[Costs and Drawbacks of Dithering](#)

What is Dithering?

A popular technique in UV, optical, and IR imaging observations involves the use of dithering, that is, spatially offsetting the telescope by shifts that are 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 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 CANDELS or PHAT fields. 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*.

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, charge traps, and IR blobs 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 most commonly used 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.

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.


1. 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 \(APT\)](#).
2. A long exposure broken into shorter exposures at each dither point will increase the amount of read noise in the final combined image.
3. If the primary science goal is to measure differential changes over time, as in time-series photometry, dithering may complicate data analysis due to flat-field uncertainties.

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 II Proposal Instructions](#) and use the [Astronomer's Proposal Tool](#). If you need additional assistance, please get in touch with your Contact Scientist or submit a ticket via the [HST Help Desk portal](#).

2.2 Selecting the Right Dither Strategy

[Dealing with Cosmic Rays, Hot Pixels, Undersampling, and Photometric Accuracy](#)
[A Top-level View of Dithering Strategies](#)
[Selecting the Number of Dither Pointings and Step Sizes](#)
[Data with Inaccurate Offsets in Position or Roll Angle](#)
[How Many Images to Obtain at Each Dither Location](#)
[Dithering Considerations for HST Instruments](#)

 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 II Proposal Instructions](#), instrument web pages, and the [Astronomer's Proposal Tool \(APT\)](#), please get in touch with your Contact Scientist or submit a ticket via the [HST Help Desk portal](#).

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 depends 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 WFC2 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 2.1: WFPC2 Exposure Time Calculator Shows Changes in SNR and Cosmic Rays Based on Number of Split Ex

Number 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/flc.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 are 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 sub-pixel amounts can improve PSF sampling, (in other words, increase spatial resolution). Generally, sub-sampling 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.

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 (ACS/WFC, WFC3/UVIS, WFCP2) or artifacts like blobs and the death star (WFC3/IR).

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:

1. Some types of observations may be unsuitable for dithering
 - a. 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.
 - b. 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.
2. 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. For programs allocated about one orbit per target per filter, 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.
3. 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.
4. 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. Indeed, a recent ISR ([ACS 2023-04/WFC3 2023-05](#)) investigates the effectiveness of the existing dither patterns in APT for use on prime+parallel observations with ACS and WFC3 and finds that patterns that were optimized for one detector often result in poor sub-pixel phase sampling in another. The report provides new dither patterns for use with prime+parallel observations which have good sub-pixel phase sampling in both instruments. For now, these dither patterns must be input into the APT manually in the form of POS TARGs.
5. Dithering in WFPC2: the Planetary Camera (PC) and Wide Field Cameras (WFC) had different scales; therefore, a dither pattern was developed to implement subpixel dithering in both camera types.

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.

Sub-pixel Dithering

Strategies and issues for sub-pixel dithering are covered in the remainder of this section. The number of sub-pixel dithers for an observation depends on the amount of available observing time and project goals.

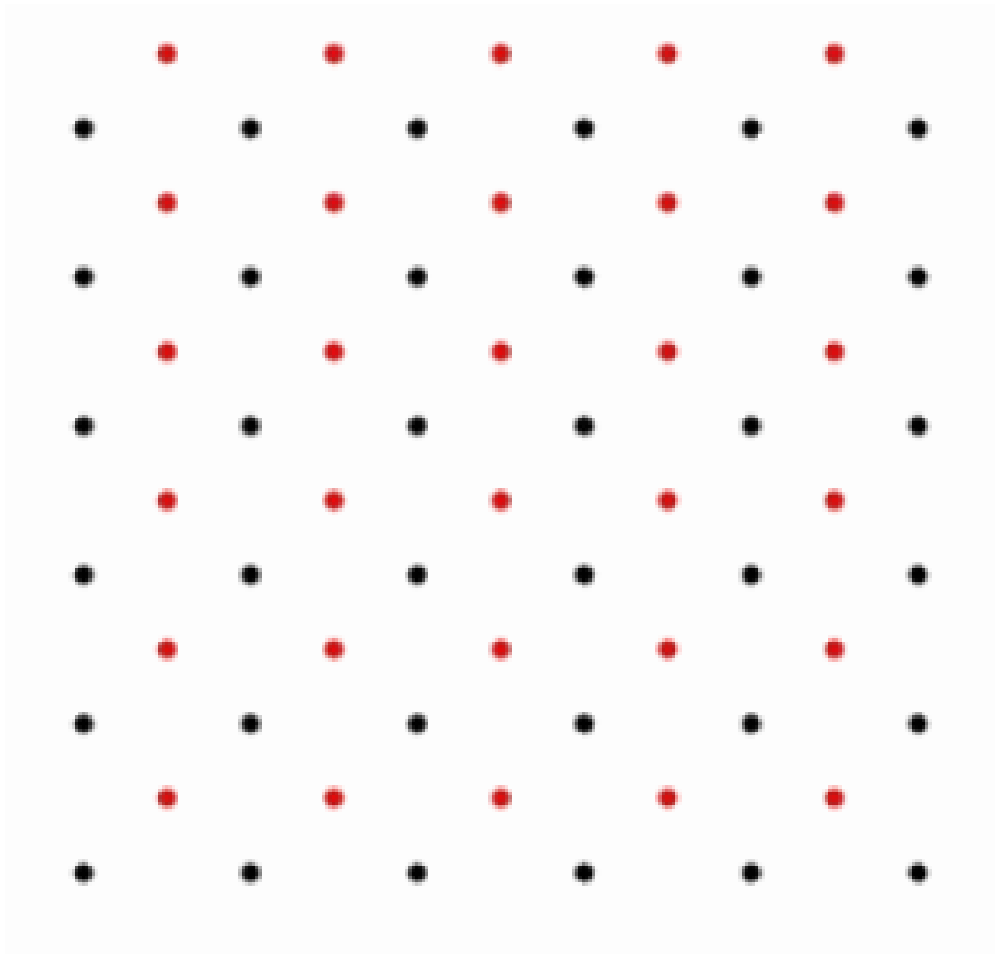
- The simplest type of sub-pixel dither is a two-point dither offset along only one axis; this is used in STIS long-slit spectroscopy for subsampling along the (spatial) slit direction. For example, one exposure would be obtained at the original pixel position of $(0,0)$ and a second obtained at $(0, n+1/2)$ pixels where n is an arbitrary integer value.
- For imaging, a two-point sub-pixel 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 2.1](#) shows the sampling by the WFC3 IR detector on the sky (note the slightly rectangular pixels), and [Figure 2.2](#) shows the sampling produced by introducing a two-point dither. The original placement is shown in black, the second dither is in red.

 WFC3 and ACS pixels are not square pixels.

Figure 2.1: The Sampling of the WFC3 IR Detector on the Sky



Figure 2.2: 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 2.3](#)).

Figure 2.3: A Four-point Dither

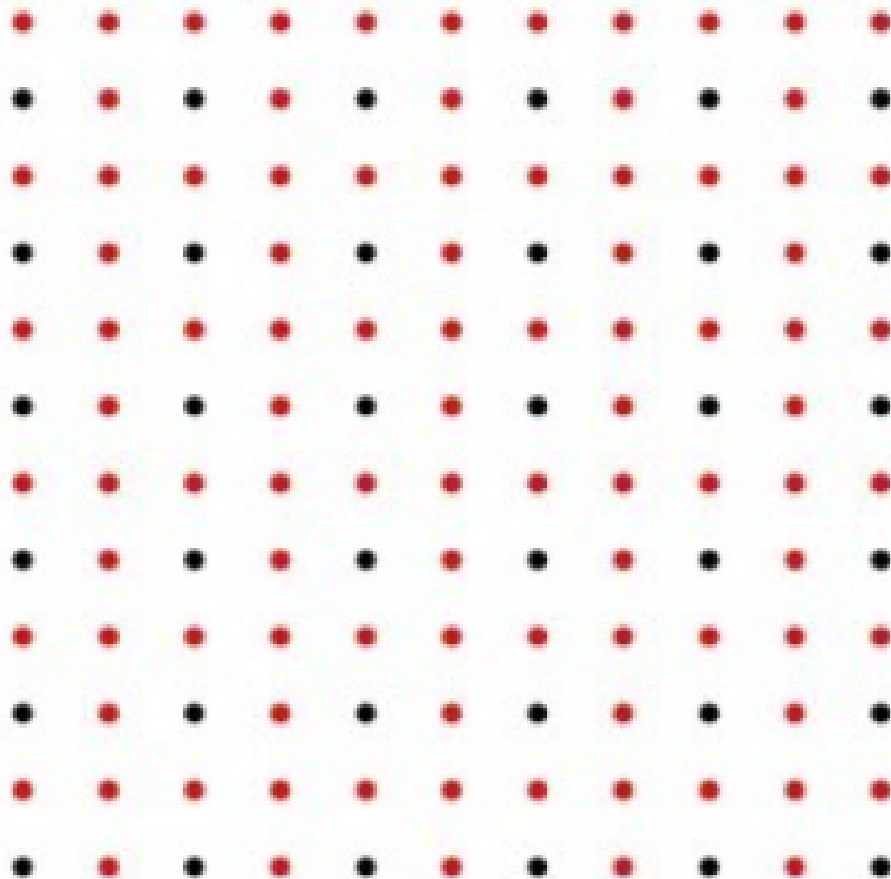
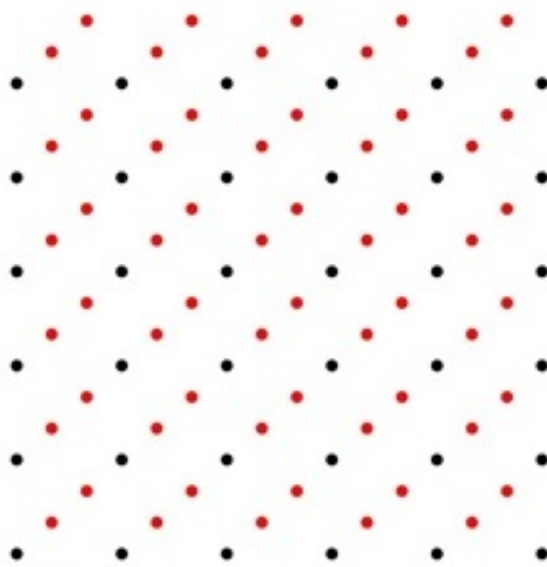


Figure 2.4: A Three-point Dither Applied to the WFC3 NIR Detector



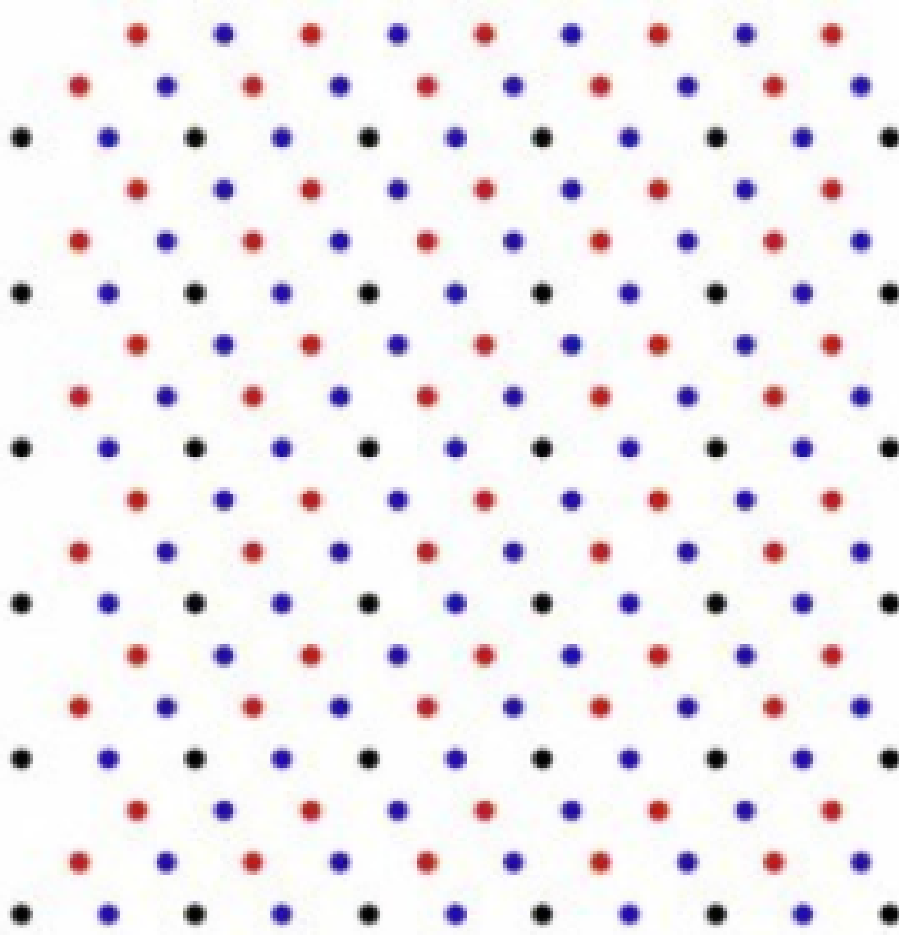
- Use of a three-point dither may arise for cases where the available observing time breaks down more naturally into blocks of three exposures, instead of two or four. However, the best placement for a three-point dither is not obvious because there is no natural way to tile the plane using three placements of a rectangular detector 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. For a three-point dither this can be accomplished 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 2.4](#) shows a three-point dither applied to the WFC3/IR 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 $(m1/4, n1/4)$ 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 sub-sampling 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 is to create a nine-point dither by dividing the original pixel with a 3×3 grid.

i 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,n) is a three-line dither with integer steps, and 'a' is a small integer added to the fractional shift.

Forming a six-point dither, 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 [Figure 2.5](#), 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 2.5: A Six-point Dither



i 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.

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. **DrizzlePac** 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**.

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 is the best approach for small programs (one orbit per target) that require sub-pixel 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 sub-pixel 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.

Dithering Considerations for *HST* Instruments

To ensure this information is kept up to date, links to the dithering strategies for various instruments are provided in this section.

Additional information is available in the [Phase II Proposal Instructions, Chapter 7: Pointings and Patterns](#) for ACS, WFC3, and STIS.



A recent ISR ([ACS 2023-04/WFC3 2023-05](#)) investigates the effectiveness of the existing dither patterns in APT for use on prime+parallel observations with ACS and WFC3 and finds that patterns that were optimized for one detector often result in poor sub-pixel phase sampling in another. The report provides new dither patterns for use with prime+parallel observations which have good sub-pixel phase sampling in both instruments. For now, these dither patterns must be input into the APT manually in the form of POS TARGs.

WFC3

Appendix C in the [WFC3 Instrument Handbook](#)

WFC3 ISRs about dithering: [WFC3 ISR 2010-09](#), [WFC3 ISR 2016-14](#), [WFC3 ISR 2020-07](#), [WFC3 ISR 2023-05](#)

ACS

[ACS ISR 2001-07: "ACS dither and mosaic pointing patterns"](#)

[ACS ISR 2019-07: "Advice for Planning ACS Observations"](#)

[ACS ISR 2023-04: "Dithering for ACS and WFC3 Primes and Parallels"](#)

[Section 7.4 of the ACS Instrument Handbook](#)

STIS

STIS dithering strategies are available in [Section 11.3 of the STIS Instrument Handbook](#)

NICMOS

Appendix D of the [NICMOS Instrument Handbook](#)

WFPC2

[Section 7.6 of the WFPC2 Instrument Handbook](#)

Chapter 3: Description of the Drizzle Algorithm

Chapter Contents

- [3.1 Image Reconstruction and Restoration Technique](#)
- [3.2 Drizzle Concept](#)
- [3.3 Weight Maps and Correlated Noise](#)
- [3.4 Characteristics of Drizzled Data](#)

3.1

Image Reconstruction and Restoration

Image Reconstruction and Restoration Technique
Interlacing
Shift and Add
Drizzle

Image Reconstruction and Restoration Technique

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.

Implementations of the Richardson-Lucy method are available online (e.g. in [skimage](#)). 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.

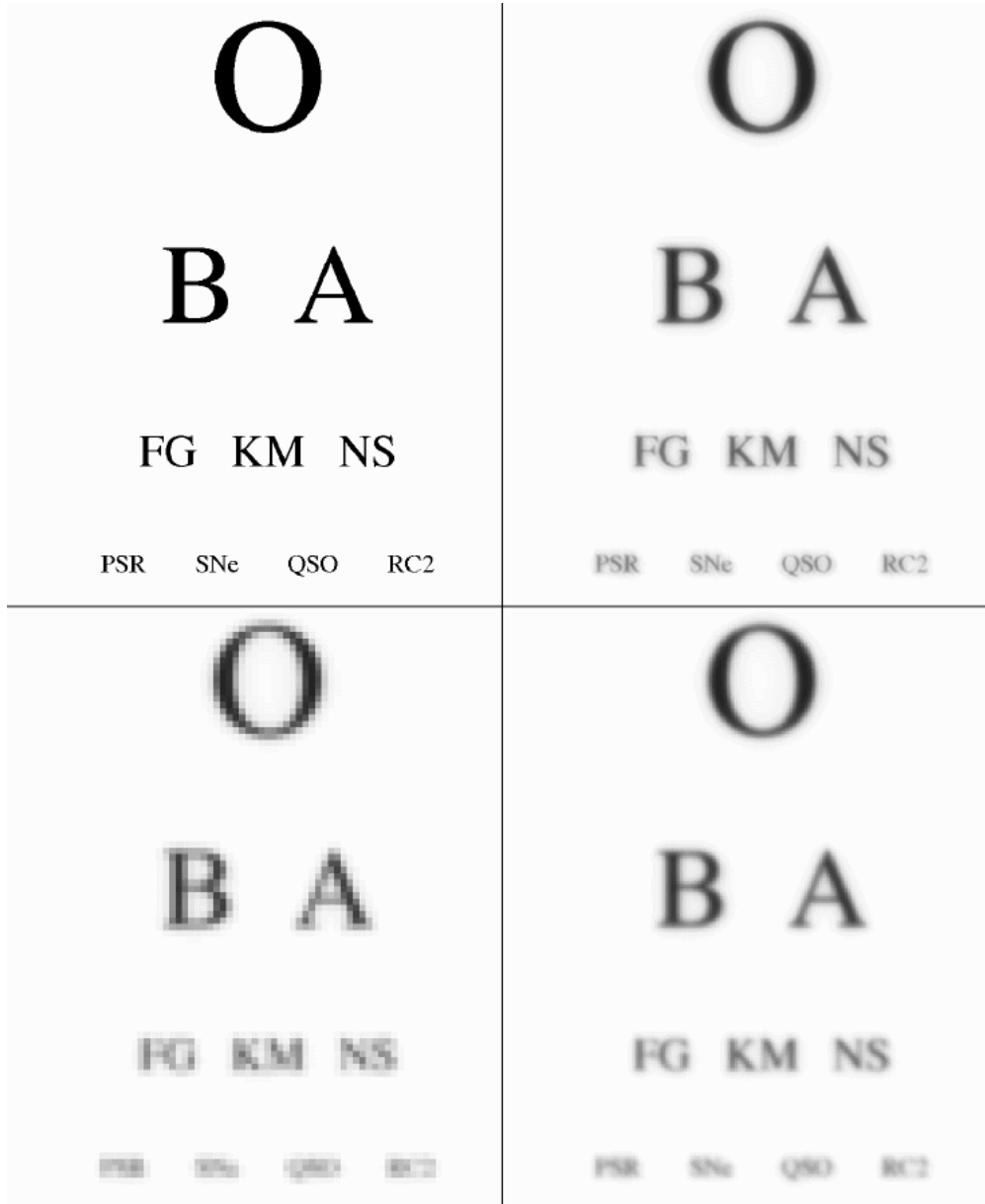
The rest of this section focuses on a family of linear reconstruction techniques that, at two opposite extremes, are represented by the *interlacing* and *shift-and-add* techniques, with the Drizzle algorithm representing a continuum between these two extremes.

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 3.1](#) was restored by interlacing a 3×3 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.

Figure 3.1: 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 the 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.

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.

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 a value 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.

3.2 Drizzle Concept

Drizzle Concept

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 3.1](#).

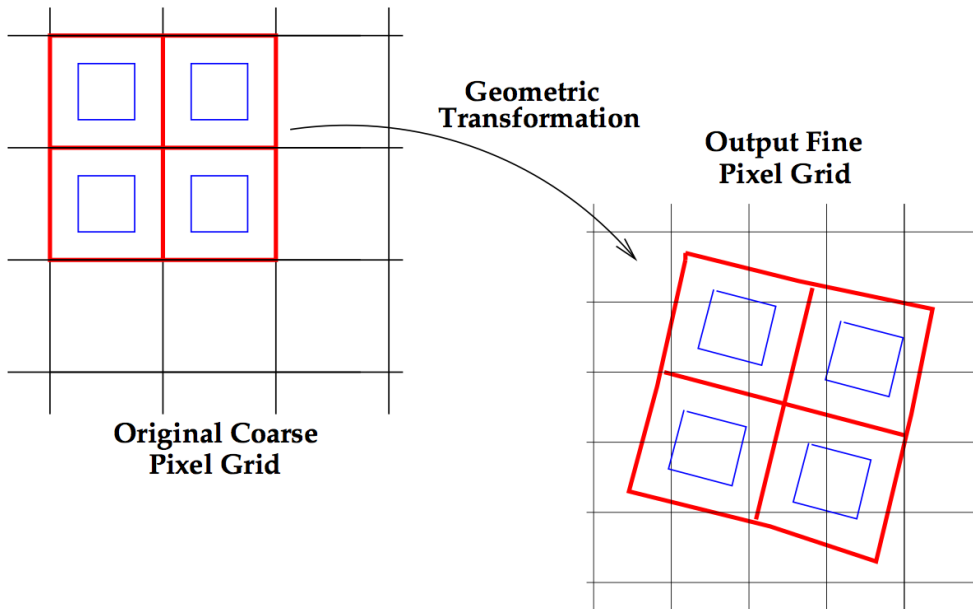
The upper left image represents a "true" image, as seen by a telescope of infinite aperture. The upper right image has been convolved with the *HST*/WFPC2 PSF. In the lower left of the set, the previously-mentioned image has been sampled by the WF2 CCD. The loss of spatial information is immediately obvious.

Much of the information lost to undersampling can be recovered. This is shown in the lower right of [Figure 3.1](#), 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 3.2](#). 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 3.2: 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 3.2. 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.

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.

i 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 arc-seconds). 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.

3.3 Weight Maps and Correlated Noise

Noise

Weight Maps
Correlated Noise

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 (those are discussed in the DrizzlePac User's Guide). 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} are:

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

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,

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

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,

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

Therefore, the weight is

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

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 the DrizzlePac User's Guide)

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 $N \rightarrow \infty$ is referred to as the noise correlation ratio, R .

One can easily see how this situation arises by examining [Figure 3.3](#).

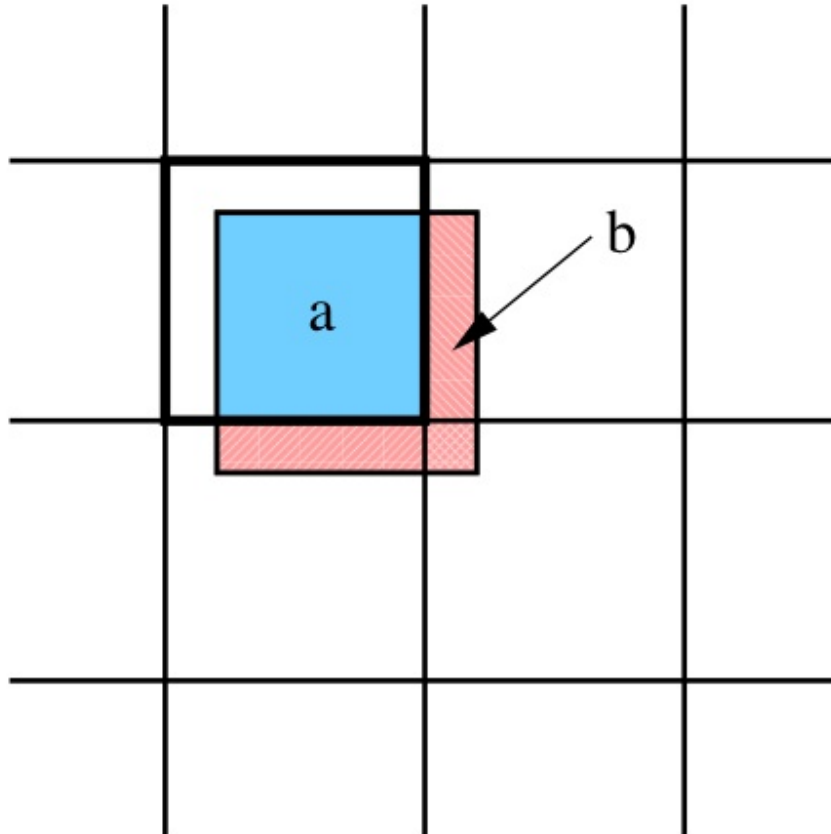
In [Figure 3.3](#) an input pixel (broken up into two regions, a and b) is being drizzled onto an output pixel plane. Let the noise in this pixel be ϵ and let the area of overlap of the drizzled pixel with the primary output pixel (shown with the heavier border) be a , and the areas of overlap with the 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:

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

This inequality exists because all cross terms (a_{b_1} , a_{b_2} , $b_1 b_2$, ...) are missed by summing the squares of the individual pixels. These terms, which represent the correlated noise in a drizzled image, can be significant.

Figure 3.3: 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.)

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.

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 the parameter 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:

$$\sigma_c^2 = \frac{(\sum d_{xy} c w_{xy}^2 s^4 \sigma_{xy}^2)}{(\sum d_{xy} c w_{xy})^2}$$

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
- s 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 $\text{pixfrac}_p > 0$. 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:

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

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 $\text{pixfrac} = p$.

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 $p=0$, P is, by definition, equal to 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,

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

Therefore,

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

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,

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

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

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 N_s . The correlated noise in the block-summed image can be estimated by replacing s with N_s in the above expressions.

3.4 Characteristics of Drizzled Data

Sampling
Photometry
Astrometry

Sampling

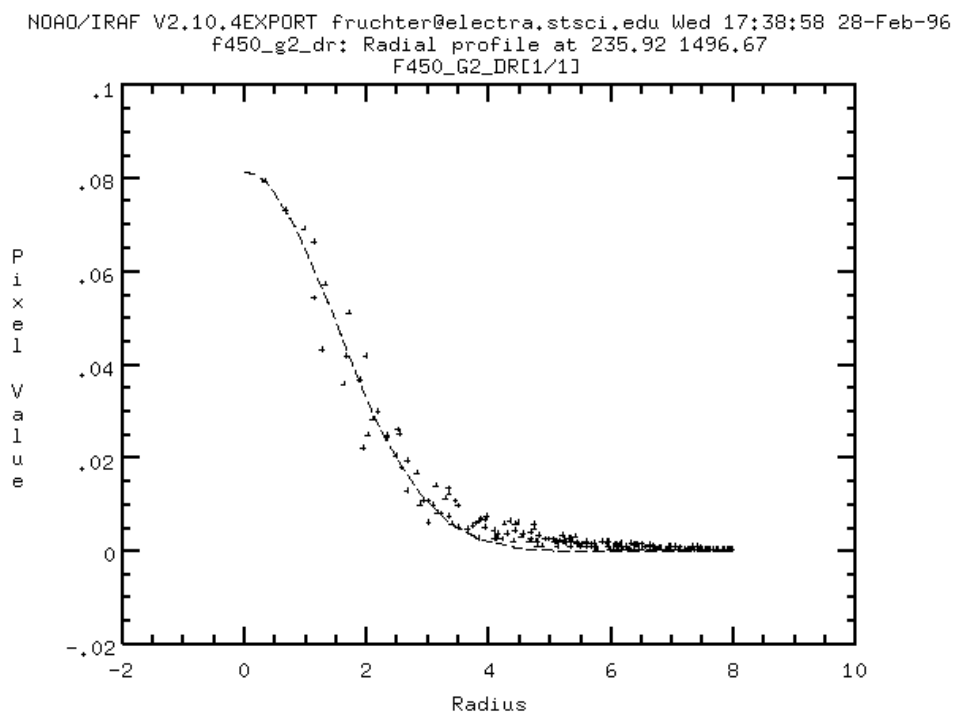
The theory of the Drizzle algorithm posits that the weight of an input pixel in the final output image is independent of its position on the chip. Therefore, if the dithered images do not uniformly sample the field, the center of light in an output pixel may be offset from the center of the pixel, and 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.

This effect is seen in the HDF-N images, where some pointings were not at the requested position or orientation. [Figure 3.4](#) and [Figure 3.5](#) 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).

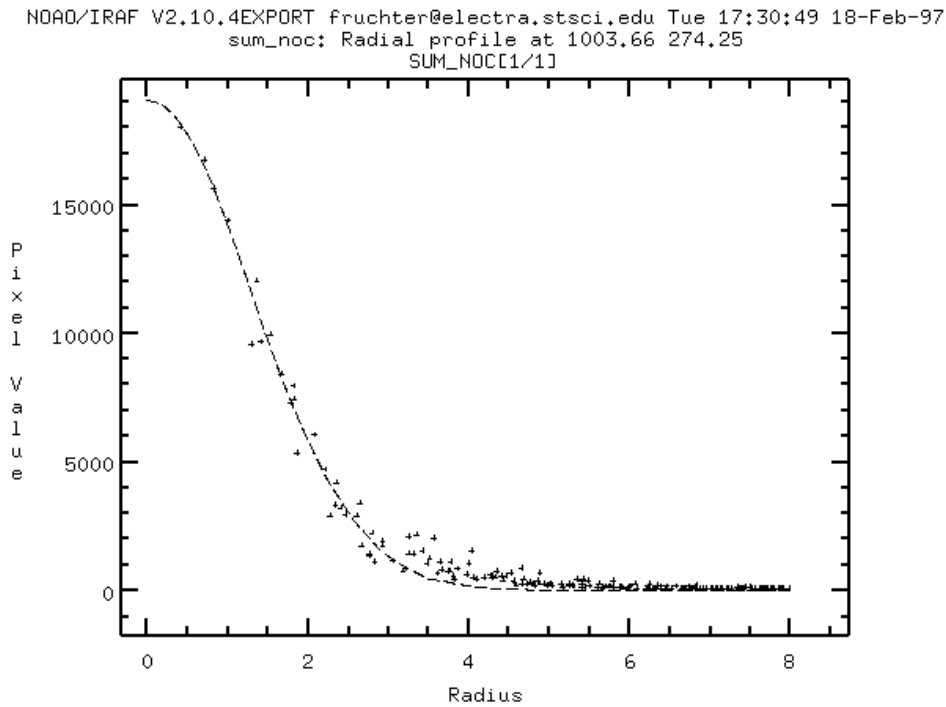
Changes in PSF can also result from other problems, such as charge transfer errors in the CCD ([Whitmore & Heyer 1997](#); [Heyer 2001](#)). Generally, however, these variations are likely to be less noticeable than effects due to non-uniform subsampling of the PSF.

Figure 3.4: 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 3.5: 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.

Photometry

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](#).

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
- PSF centering of the under-sampled *HST* images

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 using the DrizzlePac/Tweakreg task - which find accurate offsets and rotation between two images. These updated offsets and rotation 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 3.6 and 3.7 show plots of the linear solution between two *HST* WFC3/UVIS images as output from the DrizzlePac/Tweakreg task for two images with the F606W filter, observed in the same orbit. The RMS of the solution is accurate to 0.03 pixels in X and Y, and shows the random noise in the residual vectors plot across the field.

More in-depth information about astrometry and positional uncertainties are available in [Chapter 4](#).

Figure 3.6: X & Y residuals as a function of X and Y positions between two images. The RMS of the linear fit solution is 0.03 pixels in each coordinates for two WFC3/UVIS images taken in the same orbit.

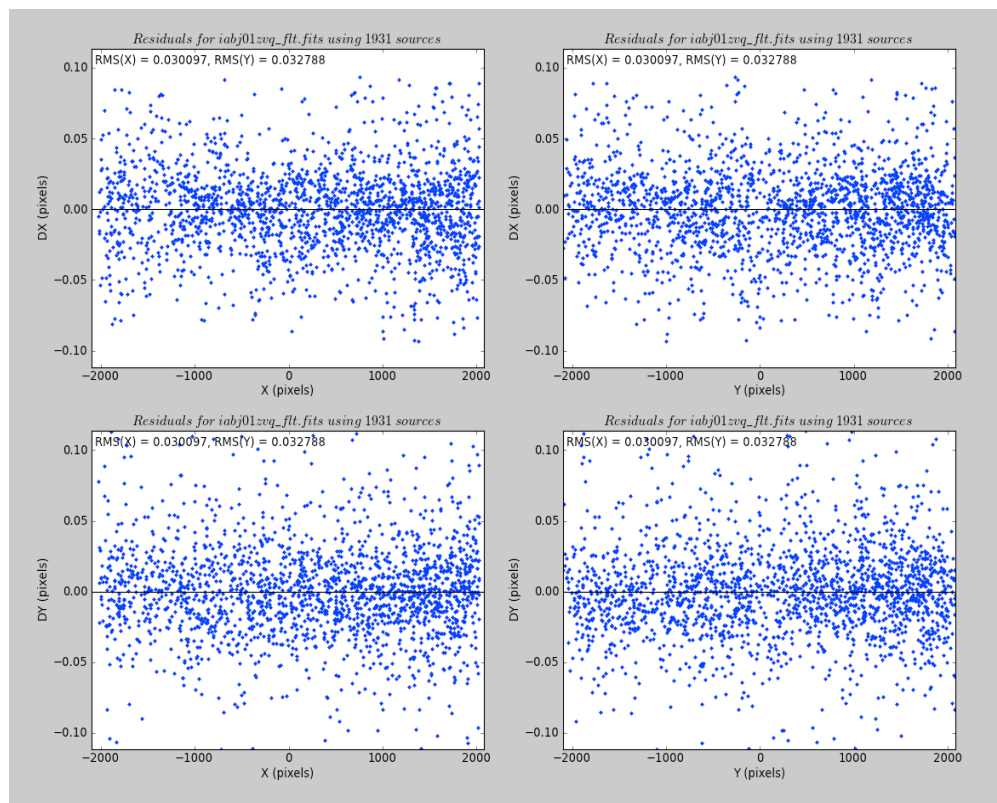
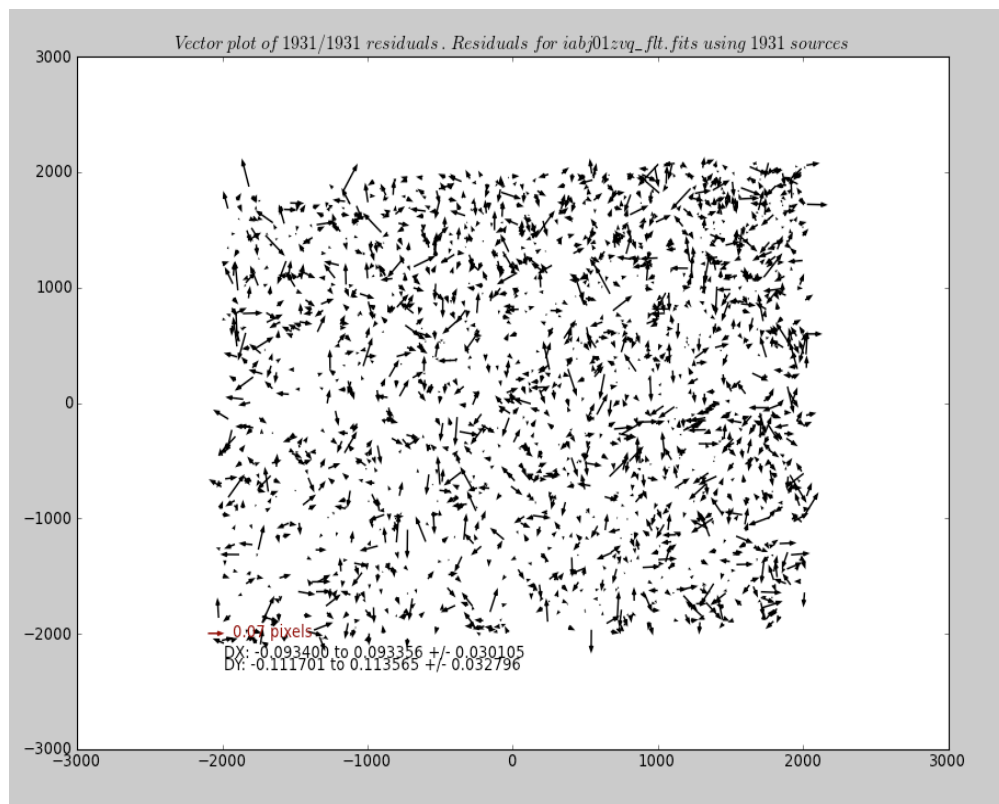


Figure 3.7: The X & Y residuals vector plot from two images taken in the same orbit.



Chapter 4: Astrometric Information in the Header

Chapter Contents

- [4.1 Introduction](#)
- [4.2 How Distortions are Represented in AstroDrizzle](#)
- [4.3 Distortion Information in Pipeline Calibrated Images](#)
- [4.4 HST Pointing Accuracy and Stability](#)
- [4.5 Absolute Astrometry](#)
- [4.6 Using Headerlets](#)

4.1 Introduction

A major feature of the AstroDrizzle software is in its handling of astrometric information. Calibrated *HST* images contain header keywords which describe several types of geometric distortion correction. Linear corrections (scale, rotation, and time-dependent skew) are incorporated in the keywords of the CD matrix. Non-linear corrections are populated as high-order polynomial coefficients using the Simple Image Polynomial (SIP) convention (Shupe, D. L., et al., 2005). These are populated in the image headers by the IDCTAB reference file. WFC3/IR data uses only this type of distortion reference file.

A third class of distortion, known as residual or non-polynomial distortion, cannot be expressed as functions. These corrections are applied in the form of look-up tables which are stored as additional FITS extensions in calibrated images. The first type corrects for pixel-grid irregularities due to the detector manufacturing process and is applied to both for WFC3/UVIS and ACS/WFC images using the D2IMFILE reference file. The second type corrects for filter-dependence in the non-polynomial distortion and is applied using the NPOLFILE reference file. Filter-dependent corrections are applied for the WFC3/UVIS detector and for all three ACS channels (WFC, HRC, SBC).

This chapter describes how AstroDrizzle handles geometric distortion information and how the reference files are used in the pipeline to produce calibrated images. [Section 4.4](#) describes the predicted accuracy of *HST* pointing, its stability, and the precision of any commanded offsets. [Section 4.5](#) summarizes the accuracy of absolute astrometry over the *HST* mission, which until recently was limited by uncertainties in the coordinates of the Guide Star Catalogs. As the accuracy of these catalogs improved over time, the pointing accuracy of *HST* has also improved.

Starting in late-2019, *HST* images were realigned to an absolute reference frame based on Gaia catalogs, making it easier to compare images from different *HST* instruments or from different telescopes. The key to incorporating this improved astrometry is in the use of *headerlets*, self-contained FITS extensions containing a World Coordinate System (WCS) transformation which can be attached to an image. [Section 4.6](#) describes how new these FITS extensions can be used to store alternate WCS solutions after aligning to a particular image or astrometric catalog.

4.2 How Distortions are Represented in AstroDrizzle

Design
World Coordinate System
The Simple Image Polynomial (SIP) Convention
Non-Polynomial Distortions

Design

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 Coefficients Table (named in the FITS header keyword `IDCTAB`).

Astrometric calculations by MultiDrizzle used 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/WFC. 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 files are 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 filter dependent part distortion correction) and `D2IMFILE` (Pixel-grid irregularities distortion - Detector to Image distortion correction). The `NPOLFILE` reference file is a 2-D table unique for each *HST* filter and is presented as residuals from high-order polynomial distortion corrections. The `D2IMFILE` reference file is also a 2-D look-up table which is corrected for pixel-grid irregularities due to the manufacturing process in ACS/WFC images, and for the lithographic-mask pattern in the WFC3/UVIS chips.

During the *HST* pipeline processing of images that require 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 (in the case of ACS/WFC, see [ACS ISR 2015-06](#)) are incorporated into the *CD matrix*.
2. Higher order polynomial corrections are stored as SIP coefficients in the image header.
3. Additional non-polynomial distortion corrections:

- Non-optical detector variations - pixel-grid irregularities;
- Residual distortion corrections for filter-dependent part of distortion

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

1. Apply the detector pixel-grid irregularities (*D2IMFILE*) to image raw pixel values;
2. Apply the SIP coefficients to *DET2IM*-corrected pixel values.
3. Combine the filter-dependent part of distortion as look-up table correction (*NPOLFILE*) with *D2IMFILE+SIP* corrected pixel values.
4. Apply the WCS transformation in the *CD matrix* to the summed results to get intermediate world coordinates.
5. 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.
6. Apply the inverse projection from the tangent plane to the celestial sphere to get the true world coordinates.

All these transformations can be conceptualized as:

$$(u,v) = D2IM(x,y)$$

where 2-D corrections applied to raw x,y pixel using the *D2IM* convention produces the coordinates (u,v) .

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

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

where u',v' are the *D2IM*-corrected coordinates relative to *CRPIX1,CRPIX2* - the reference pixel in X and Y coordinates.

$$\begin{bmatrix} \alpha \\ \delta \end{bmatrix} = \left(\begin{bmatrix} CRVAL1 \\ CRVAL2 \end{bmatrix} + P^{-1} \begin{bmatrix} CD1_1 & CD1_2 \\ CD2_1 & CD2_2 \end{bmatrix} \begin{bmatrix} u' + f(u',v') + LT_x(x',y') \\ v' + g(u',v') + LT_y(x',y') \end{bmatrix} \right)$$

where,

- α 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 in X and Y and angle of rotation
- *LT_x, LT_y* are the filter-dependent distortion corrections in 2-D 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.

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`: 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 of first axis coordinate w.r.t. x
- `CD1_2` is the partial of first axis coordinate w.r.t. y
- `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

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

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

Computation

These keyword values get computed using the operations:

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

The value of the θ angle is calculated 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 and /or WFC3 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 orientation, 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 4.3](#).

The right ascension and declination 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.

Velocity Aberration

The velocity aberration for each exposure is reported as the VAFactor keyword value, computed as a fractional scale change induced both by the velocity of *HST* as it orbits the Earth and the velocity of the Earth+*HST* as they orbit the sun. The VAFactor keyword has values near 1 (one) and represent the change in plate scale for the image. Over a one orbit period the scale can vary by as much as 5 parts in 100,000. Across a long diagonal of the ACS field of view, this amounts to about 0.3 pixels which is sufficient to degrade the registration needed for cosmic ray rejection in CR-SPLIT exposures. Images taken six months apart could have scale differences as large as 12 parts in 100,000 leading to misregistrations up to 1.4 pixels. More details on this calibration can be found in the 2002 *HST* Calibration Workshop proceedings paper: [“The Effect of Velocity Aberration on ACS Image Processing”](#).

The CD matrix is multiplied by the VAFactor to represent the actual pixel scale on the sky for each exposure. The full distortion solution including velocity aberration correction is applied by the pipeline software via the [updatewcs](#) task, which is run prior to AstroDrizzle processing in order to update all of the distortion terms with their most up-to-date values. 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, described in [Section 4.2.3](#).

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:

$$\begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} CD1_1 & CD1_2 \\ CD2_1 & CD2_2 \end{bmatrix} \begin{bmatrix} i + f(i,j) \\ j + g(i,j) \end{bmatrix}$$

where,

- β and γ 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:

$$f(i,j) = \sum_{p,q} A_{p,q} i^p j^q \quad \text{where } (p+q) \leq A_ORDER$$
$$g(i,j) = \sum_{p,q} B_{p,q} i^p j^q \quad \text{where } (p+q) \leq B_ORDER$$

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 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*.

Non-Polynomial Distortions

Image extensions containing non-polynomial distortions corrections are applied 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. 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). The full description of the non-polynomial distortions correction for ACS/WFC images is given in ACS ISR 2015-06 and for WFC3/UVIS images in WFC3 ISR 2014-12.

For the prior generation of software (MultiDrizzle), non-polynomial corrections (residual distortion corrections and detector column or row offset corrections) were represented by the [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 data for the WFC, HRC, and SBC detectors.

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 which is only applied to ACS/WFC and WFC3/UVIS images in the pipeline. Those corrections are represented here as `DET2IM`

$$(x', y') = \text{DET2IM}(x, y)$$

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

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

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

$$\begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} CD1_1 & CD1_2 \\ CD2_1 & CD2_2 \end{bmatrix} \begin{bmatrix} u' + f(u', v') + LT_x(x', y') \\ v' + g(u', v') + LT_y(x', y') \end{bmatrix}$$

i 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.

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.

i 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 and WFC3/UVIS, this offset is in the order of ~ 0.1 pixel. Finally, it is 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.

i The non-polynomial filter-dependent part of distortion corrections were obtained from the `NPOLFILE` (Non-POLynomial distortion correction) reference file during pipeline processing, and stored in the image as FITS extensions.

4.3 Distortion Information in Pipeline Calibrated Images

[Image Structure](#)
[Keywords](#)
[Drizzled Products](#)

Image Structure

ACS and WFC3 images retrieved from the *HST* Archive are calibrated using the best available calibration file, including latest available distortion correction models.

Calibrated science images from the *HST* Archive, with the suffix (*flt.fits*, *flc.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, calibrated 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 for each instrument stored in a reference file called the 'MDRIZTAB' (**mdz.fits*).

As described in the previous section, WCS information in calibrated images is updated by AstroDrizzle in the pipeline to include the full distortion correction model:

- 2-D corrections for ACS/WFC and WFC3/UVIS pixel-grid irregularities with D2IMFILE reference files, appended to the science image as FITS extensions of D2IMARR type
- Scale, rotation, and time-dependent skew (in the case of ACS/WFC) 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 the NPOLFILE reference file is also appended to the science image as new FITS extensions of WCSDVARR type

The code block below shows the new structure of a typical ACS/WFC image.

```
>>>from astropy.io import fits
>>>fits.info('j6d508ojq_flc.fits')

Filename: j6d508ojq_flc.fits
No.      Name      Ver      Type      Cards      Dimensions      Format
  0  PRIMARY      1  PrimaryHDU      283      ( )
  1  SCI          1  ImageHDU       242      (4096, 2048)   float32
  2  ERR          1  ImageHDU       53       (4096, 2048)   float32
  3  DQ           1  ImageHDU       45       (4096, 2048)   int16
  4  SCI          2  ImageHDU      240      (4096, 2048)   float32
  5  ERR          2  ImageHDU       53       (4096, 2048)   float32
  6  DQ           2  ImageHDU       45       (4096, 2048)   int16
  7  D2IMARR      1  ImageHDU       16       (64, 32)        float32
  8  D2IMARR      2  ImageHDU       16       (64, 32)        float32
  9  D2IMARR      3  ImageHDU       16       (64, 32)        float32
 10  D2IMARR      4  ImageHDU       16       (64, 32)        float32
 11  WCSDVARR     1  ImageHDU       16       (64, 32)        float32
 12  WCSDVARR     2  ImageHDU       16       (64, 32)        float32
 13  WCSDVARR     3  ImageHDU       16       (64, 32)        float32
 14  WCSDVARR     4  ImageHDU       16       (64, 32)        float32
 15  HDRLET       1  NonstandardExtHDU 18      (8640, )
 16  HDRLET       2  NonstandardExtHDU 26      (112320, )
 17  WCSCORR      1  BinTableHDU    59      14R x 24C [40A, I, A, 24A, 24A, 24A, 24A, D, D, D, D, D, D, D, D,
24A, 24A, D, D, D, D, J, 40A, 128A]
 18  HDRLET       18 NonstandardExtHDU 26      (112320, )
 19  HDRLET       4  NonstandardExtHDU 26      (112320, )
```

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 pipeline, only ACS/WFC and WFC3/UVIS images require residual optical distortion corrections, stored in the image files as FITS extensions of D2IMARR and WCSDVARR type.

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, python is used to query the header keywords `SIPNAME`, `IDCTAB`, `NPOLFILE`, and `D2IMFILE` for an ACS/WFC image.

The `DISTNAME` keyword is also shown to illustrate naming nomenclature for the keyword value (eg. `ROOTNAME_IDCTAB_NPOL_D2IM`):

```
>>>from astropy.io.fits import getheader
>>>hdr=getheader('j6d508ojq_flg.fits',0)
>>>print([hdr[i] for i in ['SIPNAME', 'IDCTAB', 'NPOLFILE', 'D2IMFILE', 'DISTNAME']])

['j6d508ojq_4bb1536oj', 'jref$4bb1536oj_idc.fits', 'jref$4bb1536hj_npl.fits', 'jref$4bb15371j_d2i.fits',
'j6d508ojq_4bb1536oj-4bb1536hj-4bb15371j']

>>>from astropy.io.fits import getheader
>>>hdr=getheader('j6d508ojq_flg.fits',0)
keynames=['sipname','idctab']
[print(f'{name}:{hdr[name]}') for name in keynames]

sipname:j6d508ojq_4bb1536oj
idctab:jref$4bb1536oj_idc.fits
```

Similar for any ACS/WFC image but with different name of the reference files.

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 4.1: 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	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, $(p + q) \leq B_ORDER$

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

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 in ACS/WFC and its keywords

The linear terms of the ACS/WFC geometric distortion is changing with time as it described in [ACS ISR-2015-06](#), which is hard coded now in the new DrizzlePac as follows:

$$\begin{aligned} \text{OCX11}' &= \text{OCX11} - \text{TDD_CXB1} \times (\text{TIME} - \text{RDATE}) \\ \text{OCY11}' &= \text{OCY11} - \text{TDD_CTB1} \times (\text{TIME} - \text{RDATE}) \end{aligned}$$

where OCX11 and OCY11 are averaged linear coefficients in IDCTAB formalism and correspond to scale in X and rotation in Y respectively for each CCD chip, WFC1 and WFC2.

The linear time-dependency TDD_CXB1, TDD_CTB1 and RDATE (2004.5 and 2012.0 before and after SM4) are kept in the primary header of the IDCTAB which are:

TDD_DATE= 2012.0


TDD_CT1= 1.6597930806E-06 / Y-rotation cy_11

TDD_CT2= 1.5787128139E-06 / Y-rotation cy_11

TDD_CX1= -1.0217767093E-06 / X-scale cx_11

TDD_CX2= -1.0658206323E-06 / X-scale cx_11

Since the plate-scale and the parameters of the linear time-dependent distortion are slightly different before and after SM4, two IDCTAB (before and after SM4) are provided to the user and to the *HST* data-base reference system to be used in DrizzlePac and the *HST* pipe-line. The SIP convention presented by the CD matrix of the linear terms of distortion is convenient to apply the time-dependent of the linear terms for each chip.

 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 $q_3' = q_3 + \delta_{q_3}(q_1', q_2')$

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`, `flc.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 is 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 4.2: 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.
D2IMERR1, 2	Maximum value of the correction for axis 1,2

Drizzled Products

AstroDrizzle's primary product is a multi-extension FITS file with the suffix either `drz.fits`, `drc.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 astrometrically 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).

The default set of **astrodrizzle** parameters defined for each instrument may be changed by the user during post-pipeline reprocessing to best suit the actual data and user's needs. 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 is written to the drizzled 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.

4.4 HST Pointing Accuracy and Stability

Pointing
Tracking Stability
Precision of Commanded Offsets

Pointing

An understanding of *HST's* pointing stability and the accuracy of commanded offsets is essential for planning dither observing strategies, regardless of whether integer or subpixel shifts are desired. 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 4.3.

Table 4.3: Typical *HST* pointing and stability characteristics

Observing Scenario (with fine lock on two guide stars)	Type of Program	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
<i>Guiding with a single guide star</i>	<i>Unavoidable in many crowded fields such as those along the galactic plane</i>	<i>1.5 mas/sec roll For ACS: 0.052 arcsec /orbit in WFC</i>

Tracking Stability

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.

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.

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.

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.

4.5 Absolute Astrometry

[HST Pointing Accuracy](#)
[Absolute Astrometry Improvements in MAST](#)
[New Data Products for WFC3, ACS, and WFPC2](#)
[Caveats](#)

HST Pointing Accuracy

Historically, the accuracy of *HST* absolute astrometry has been limited primarily by uncertainties in the celestial coordinates of the guide stars as specified in the [Guide Star Catalog](#). GSC 1.1 had nominal rms errors of ~ 0.5 arcsec per coordinate, with errors as large as $\sim 1\text{--}3$ arcsec reported near the plate edges. This accuracy improved substantially in October 2005 (during HST Cycle 15) with the introduction of GSC 2.3.2, where rms errors per coordinate were reduced to ~ 0.3 arcsec over the whole sky. An updated version of the catalog (GSC 2.4.0) was released in October 2017, improving the celestial coordinates with the positions from Gaia DR1 and reducing errors to $< 30\text{mas}$ over the entire sky. Thus, after including uncertainties in the positions of the science Instruments (SIs) in the alignment of the focal plane to the Fine Guidance Sensors (FGS), the total error in *HST* absolute astrometry is ~ 1 arcsec for observations made with GSC1.1, ~ 0.3 arcsec for those made with GSC 2.3.2, and ~ 0.1 arcsec when using the newer GSC 2.4.0. A summary of the GSC catalogs and associated errors over the *HST* lifetime is provided in [Table 4.4](#).

Information on the guide stars used for a particular observation can be found in the FITS header of the associated jitter file, which can be retrieved from MAST and end with the suffix *jif.fits*. Additional information such as the positional uncertainty and parallax of these guide stars can be found by using the [web form here](#).

Table 4.4: Key Guide Star Catalog releases and associated errors

Catalog	Release Date	Mean Epoch of catalog positions	Typical errors	Worst errors	Total Error (including SI to FGS alignment)	Comment
GSC 2.4.0	Oct 2017	2015.0	0.03"		0.1"	GSC2.3.4 aligned to Gaia DR1 Complete GSC Summary
GSC 2.3.3	Oct 2009					WFC3 installed May 2009
GSC 2.3.2	Oct 2005	1992.5	0.3"	0.75"	0.3"	Public Release GSC 1.1 and GSC 2.3.2 Comparison
GSC 2.2.0	Jun 2001					Public Release ACS installed Mar 2002
GSC 2.0	Jan 2000					Science target fields only; GSC2 summary
GSC 1.1	Aug 1992	1981.8	0.5"	~1"	~1"	First version published for the user community Used by <i>HST</i> operations prior to Cycle 15 WFPC2 installed Dec 1993
GSC 1.0	Jun 1989			1-2"		GSC1 summary

Absolute Astrometry Improvements in MAST

The coordinates populated in the FITS headers of *HST* observations retrieved from DADS (the *HST* Data Archiving and Distribution Service) were derived based on the guide star coordinates in use at the time of the observation. As the accuracy in these catalogs were refined over time, the pointing accuracy of *HST* has also improved. [Table 4.4](#) lists the catalog in use at the time of installation of the three main imaging cameras (WFPC2, ACS, and WFC3) and the typical errors at each epoch.

The goal of the *HST* Astrometry Project is to correct these inconsistencies in the archival data products as much as possible. As observations are processed or reprocessed in the *HST* pipeline, their World Coordinate System (WCS) will be updated to use the most accurate solution available. There are two types of corrections that can be performed:

- *a priori*: correct the coordinates of the guide stars in use at the time of observation to the coordinates of those guide stars as determined by Gaia by applying a global offset to the WCS
- *a posteriori*: identify sources in the *HST* image and cross-match with positions from an external reference catalog (such as Gaia) to derive an improved WCS based on fitting x/y to RA/Dec


Note that *a priori* corrections are only relevant for observations which executed prior October 2017 (e.g. prior to the release of GSC 2.4.0), and these will still include small errors in the alignment of the science instruments to the *HST* focal plane. The *a posteriori* corrections are limited to imaging instruments for which there are an adequate number sources to define a reference catalog for matching. These solutions remove uncertainties in the focal plane and are expected to have the smallest absolute astrometric error.

New Data Products for WFC3, ACS, and WFPC2

Enhanced data products were delivered to MAST in several stages and are summarized below.

- **December 2019** - Improved astrometry for WFC3 and ACS imaging data includes two corrections to the header world coordinate system (WCS). The first includes a new version of HST's Guide Star Catalog (GSC, version 2.4.0) which updates the coordinates of the guide stars with the positions from Gaia DR1. This reduces the typical uncertainties in the positions of the guide stars to $< \sim 100$ mas over the entire sky. Combining this new information with the knowledge of the instrument distortions, an *a priori* correction has been made for all WFC3 and ACS observations in order to lock all *HST* observations onto a common absolute reference frame. When possible, an additional *a posteriori* correction has been applied by aligning sources in each *HST* image to an external reference catalog (e.g. Gaia). While some observing modes cannot be aligned to Gaia (e.g. grism and moving target observations) or the alignment may fail due to a lack of sources in either the *HST* image or the Gaia catalog, approximately 70% of the WFC3 and ACS frames have been aligned successfully. For these data products, the typical pointing uncertainty is reduced to ~ 10 mas, although the uncertainties increase for observations further in time from the Gaia reference epoch (2015.0 for DR1, 2015.5 for DR2, and 2016.0 for eDR3). The software used to create the new data products is described on the [Pipeline Astrometric Calibration](#) page.
- **December 2020** - Production of new advanced data products for ACS and WFC3 begins in the MAST pipeline. These [Hubble Legacy Archive](#) (HLA)-style mosaics comprise the data from a single HST visit which are aligned to a common astrometric reference frame. These new 'Hubble Advanced Products' (HAP) are referred to as '**Single Visit Mosaics**' (SVMs) and are described in a [MAST Newsletter article](#). The data products are all drizzled onto the same north-up pixel grid and may include improved relative alignment across filters for datasets acquired within the same visit, enabling easy comparison of the images through multiple filters. When possible, sources in the images have been aligned to Gaia to improve the WCS. SVM data products with both relative alignment (by filter) and absolute alignment to Gaia will contain the string 'FIT_SVM_GAIA' in the 'WCSNAME' keyword in the science extension of the image header. The software used to compute these new data products is described in the DrizzlePac documentation for [Single Visit Mosaic Processing](#).
- **November 2021** - Source catalogs are now included as part of the SVM data products. Because the SVM drizzled images include an additional relative alignment across filters in a visit, these are used to generate point source and segment catalogs during pipeline processing. These catalogs supersede those produced by the Hubble Legacy Archive and will be the basis of the next version of the [Hubble Source Catalog](#).

- **April 2022** - A new type of Hubble Advanced Product is distributed through MAST. These are cross-visit, cross-proposal mosaics called '**Multi-Visit Mosaics**' (MVMs) which combine all ACS/WFC, WFC3/UVIS, or WFC3/IR images falling within a pre-defined $0.2^\circ \times 0.2^\circ$ 'sky cell' for each detector+filter and drizzled to a common all-sky pixel grid. MVM products are described in the following [MAST Newsletter article](#) and complement the Single Visit Mosaics (SVMs). Because they may include observations spanning a large date range, MVMs may have photometric errors of several percent or systematic alignment errors when combining visits which are aligned to different reference catalogs. These products are therefore recommended as 'discovery images' for comparing observations in different detectors and passbands and not for precise photometry.
- **December 2023** - MAST begins producing new WFPC2 data products with improved absolute astrometry. The calibrated images combine the previously separate science array 'c0m.fits' and data quality array 'c1m.fits' files into a new 'flt.fits' data product, similar to ACS and WFC3. These are used to produce new 'drw.fits' drizzled data products, where the 'drw' suffix reflects the improved WCS. These are drizzled to the scale of the PC chip ($0.0455''/\text{pix}$) and replace the old 'drz.fits' products, which were drizzled to the scale of the WF chips ($0.0996''/\text{pix}$). For more details, see the notebook [Drizzling new WFPC2 fit data products](#).

 A detailed instrument science report '*Improved Absolute Astrometry for ACS and WFC3 Data Products*' describes the new WCS solutions which are present in both MAST 'standard' data products ('ippssoot_drz.fits') and in the advanced SVM and MVM data products. Statistics are provided on the alignment fraction for each detector and the estimated uncertainties when aligning to different reference catalogs. (See [ACS ISR 2022-03](#) or [WFC3 ISR 2022-06](#) for more information).

WCS Naming Conventions

Successfully aligning an observation to Gaia using the *a posteriori* processing will result in an update of the 'active' WCS of the image with the new solution and a new '[headerlet](#)' extension. This headerlet includes the WCS keywords which define the transformation from pixels to Gaia-aligned positions on the sky, as well as information about how this solution was derived and the uncertainty of the fit.

The various WCS solutions are identified by the `WCSNAME` keyword found in each FITS headerlet and use the following naming convention:

wcsName = OriginalSolution - CorrectionType

where **OriginalSolution** may be either

- OPUS : initial ground system wcs, no distortion correction
- IDC_XXXXXXXX : initial distortion corrected wcs (where XXXXXXXXX = geometric distortion model used, eg. the rootname of the IDCTAB reference file)

and **CorrectionType** may have several forms

- GSC240 : '*a priori*' WCS where guide star coordinates are corrected from the original reference frame (e.g. GSC1.1 or GSC2.3) to the Gaia DR1-based GSC2.4.0
- HSC30 : '*a priori*' WCS corrected from the original reference frame to the Hubble Source Catalog (HSC v3.0) frame, which is based on Gaia DR1
- FIT-IMG-RefCat : '*a posteriori*' WCS matched to a reference catalog, where 'IMG' implies **each FLT is separately aligned** to the reference catalog
- FIT-REL-RefCat : '*a posteriori*' WCS matched to a reference catalog, where 'REL' implies that **FLTs within the same filter within the same visit are aligned** before a global catalog alignment
- FIT-SVM-RefCat : '*a posteriori*' WCS matched to a reference catalog, where 'SVM' implies that **FLTs in multiple filters within the same visit are aligned** before a global catalog alignment

and **REFcat** may be one of the following when an adequate number of matches are found in the HST frame to compute the linear transformations (shift, rotation, scale) to sky coordinates:

- Gaia eDR3
- GSC 2.4.2
- 2MASS

A list of possible WCSNAME values in the FITS image headers is provided in [Table 4.5](#). More details on interpreting the WCS names may be found on the [Astrometry in Drizzled Products](#) page.

Table 4.5: Sample WCSNAME keyword values and the corresponding WCSTYPE description.

WCSNAME	WCSTYPE	Comment
OPUS	'distorted not aligned'	No distortion correction has been applied; analysis of these FLT /FLC files may only be performed if corrected by the instrument-specific pixel area map
IDC_0461802ej	'undistorted not aligned'	Distortion-correction applied using the IDCTAB reference file '0461802ej_idc.fits', with no additional corrections
IDC_0461802ej-GSC240	'undistorted <i>a priori</i> solution based on GSC240'	WCS is based on the Guide Star Catalog v2.4.0 (GSC240). Absolute errors ~0.1"
IDC_0461802ej-HSC30	'undistorted <i>a priori</i> solution based on HSC30'	WCS is based on Hubble Source Catalog v3.0. HSC30 errors are typically smaller than GSC240. If both corrections are available, HSC takes precedence.
IDC_0461802ej-FIT_REL_catalog	'undistorted <i>a posteriori</i> solution relatively aligned to catalog'	Exposures are aligned to one another, and then aligned "as a set" to the reference catalog.
IDC_0461802ej-FIT_IMG_catalog	'undistorted <i>a posteriori</i> solution aligned image-by-image to catalog'	Exposures are individually aligned to the reference catalog (not as a set)

IDC_0461802ej-
FIT_SVM
_catalog

'undistorted *a posteriori*
solution relatively aligned
filter-by-filter to catalog*'

HAP-SVM solution; Exposures are aligned "as a set" to the reference catalog and include improved relative alignment across filters in a visit. (Typically the BEST WCS solution)

Implementation

The key to implementing improved astrometry is the use of *headerlets*, self-contained FITS extensions containing a WCS transformation which can be attached to a FITS file and applied to the primary WCS. MAST data processed with the [Enhanced Pipeline Products code](#) will have headerlets added as extra extensions to the FITS file. An observation can have multiple headerlets, each of which may have astrometry derived by differing methods. As *HST* data is processed/reprocessed, all available headerlets will be present as FITS extensions in the archived image with the *best* solution applied to the primary WCS. More details on how the WCS information is stored in headerlets may be found on the page [Astrometry in Drizzled Products](#).

Once the observations are downloaded from MAST, the python notebook '[Improving Astrometry Using Alternate WCS Solutions](#)' will familiarize users with how to check the active WCS or switch to an alternate WCS, if desired. Alternatively, the WCS solutions may be downloaded directly from MAST as separate 'hlet.fits' headerlet files and applied to existing FITS data. Python functions for creating, updating, and applying headerlets to FITS images are described via the [Headerlet User Interface](#).

Caveats


While the majority of calibrated HST data products are now aligned to a common absolute reference frame, further improvements may be possible via manual realignment using the [drizzlepac](#) tools. This is particularly true for exposures acquired in the same visit in which the `WCSNAME` keyword values do not match. Additionally, the quality of the WCS may be compromised when the number of sources used for the alignment is small, e.g. the `NMATCHES` keyword has a value less than ~ 10 .

For standard drizzled data products 'ippssoot_drz.fits', alignment errors may be present in the following situations:

- Short and long exposures obtained in the same visit may no longer be aligned due to potentially different number of Gaia matches.
- Exposures in different filters (eg. narrowband vs broadband) acquired in the same visit may no longer be aligned to one another, e.g. if each filter had a different number of Gaia matches.
- Grism images may now be offset from their direct image counterparts, where only the later may be aligned to an external reference catalog. In order to preserve relative alignment between grism and direct images, users may wish to back out the updated WCS solutions entirely, as described in Section 4 of the notebook, '[Improving Astrometry Using Alternate WCS Solutions](#)'.

The Single Visit Mosaic drizzled data products 'hst_ippssoo_*_drz.fits' are recommended for the best alignment across all exposures acquired in a single visit, and the `WCSNAME` keywords will contain the string 'FIT_SVM', as shown in [Table 4.5](#). If desired, these WCS solutions may be downloaded as separate headerlet files,

applied to standard calibrated data products ('`ippssootflt.fits`') and redrizzled.

 An updated set of [Drizzlepac tutorials](#) have been developed for working with the new MAST data products and are compatible with the latest STScI distributed software environment [stenv](#). For details, see Chapter 8.

4.6 Using Headerlets

[Introduction](#)
[WCS Information in Archival Images](#)
[Storing Multiple WCSs in an image](#)
[Headerlet Structure](#)
[Working with Headerlets](#)
[Python Syntax of the Headerlet Tasks](#)

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 `iabf01ckqflt.fits` for the WCS solution named `TWEAK` would be named `iabf01ckqtweakhlet.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.

WCS Information in Archival Images

For active instruments¹ (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 DGEOTAB reference image, are now stored directly in the images in the form of keyword values and FITS extensions. [This report](#) contains a 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`², `A_ORDER`, `B_ORDER`, `OCX10`, `OCX11`, `OCY10`, and `OCY11`
- ³Residual 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](#))
- `7WCSDVARR` 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)
- `7Column` 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`.

- [7D2IMVARR](#) 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](#).

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.

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, making them a space efficient means of managing all of the distortion and WCS information.

WCSCORR Image Extension for Storing Multiple WCSs

The code block in [Section 4.3](#) 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.

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 headerlet 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 extensions⁴ 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](#)".

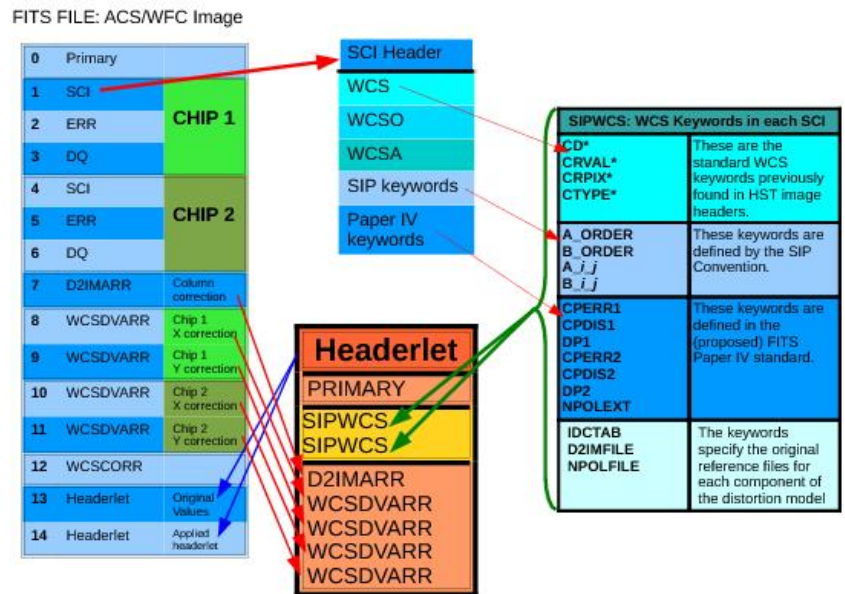
On the other hand, the STWCS package used by DrizzlePac tasks can use a headerlet to define a WCS for an image to perform coordinate transformations (complete with distortion correction) without even requiring the original science image and all its data.

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 in a headerlet, created from an ACS/WFC image, is shown in "Accessing the Headerlet Extension using PyFITS". 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 4.1 below.

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 4.1: 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 headers also provide a record of how much of an offset there is between this solution and the default OPUS-generated solution.

Working with Headerlets

A set of tasks in `stwcs.wcsutil.headerlet` have been developed to support interactive management of headerlets, either creating, applying, or even deleting headerlets from a given image.

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

Creating a Headerlet

The tweakreg task 'create_headerlet' 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
<code>filename</code>		Name(s) of science file(s) from which headerlets will be created and written. <ul style="list-style-type: none"> • 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 <code>datadir*\$flt.fits</code> as input, assuming “datadir” has been defined in the user’s environment.
<code>hdrname</code>		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_ft.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. <ul style="list-style-type: none"> • If not found, for <i>HST</i> 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.
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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 filename 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 overwrite 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.

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`.

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 j94f05bgqflt.fits and label the Headerlet with the name “VERSION1” would be:

```
>>> from stwcs.wcsutil import headerlet
>>> hdrlet = headerlet.createHeaderlet('j94f05bgqflt.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('94f05bgqflt_with_hlet.fits')
>>> hdul.info()
Filename: j94f05bgqflt_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.

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('j94f05bgqflt_with_hlet.fits')
>>> hdul.info()
Filename: j94f05bgqflt_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 WCS CORR BinTableHDU 56 10R x 23C [40A,I,1A D,D,D,D,D,D,D,24A,
24A,D,D,D,D,D,D,D,D,D,J,40A]
8 WCS DVARR ImageHDU 15 (65, 33) float32
9 WCS DVARR ImageHDU 15 (65, 33) float32
10 WCS DVARR ImageHDU 15 (65, 33) float32
11 WCS DVARR 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_qbul641sj' / SIP distortion model name
NPOLFILE= '/grp/hst/acs/lucas/new-npl/qbul6424j_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 WCS DVARR ImageHDU 15 (65, 33) float32
4 WCS DVARR ImageHDU 15 (65, 33) float32
5 WCS DVARR ImageHDU 15 (65, 33) float32
6 WCS DVARR 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.

```

1 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.

2 For $A_{p,q}$, $(p + q) \leq A_ORDER$; for $B_{p,q}$, $(p + q) \leq B_ORDER$

3 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.

4 FITS-compliant files can be a multi-extension FITS file or an given extension to a multi-extension FITS file.

Chapter 5: DrizzlePac Software Package

Chapter Contents

- [5.1 DrizzlePac: An Overview](#)
- [5.2 AstroDrizzle The New Drizzle Workhorse](#)
- [5.3 AstroDrizzle in the Pipeline](#)
- [5.4 The DrizzlePac Package](#)
- [5.5 Configuration Files \(cfg\)](#)

5.1 DrizzlePac: An Overview

DrizzlePac is a package of tasks used primarily for registration and resampling of images after the calibration pipeline. **AstroDrizzle** and **TweakReg** are its flagship tasks, and the rest of DrizzlePac's tasks support other drizzling operations.

AstroDrizzle uses the same historic algorithmic base implemented in previous drizzling software with a modified set of core routines that have been recoded in C and Python. One notable change in comparison to previous versions of drizzling software is that AstroDrizzle moves pixels according to the astrometry encoded in the WCS, as opposed to doing so according to shifts.

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 (`flc.flat/flt.fits`) as follows:

- Using astrometric information and distortion information found 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 the first approximation of the "truth" image.
- The median image is then blotted, or reverse-drizzled, back to the frame of each input `flc.flat/flt.fits` image.
- By comparing each `flc.flat/flt.fits` image with its blotted counterpart median image, the software locates bad pixels in each of the original `flc.flat/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, 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.

5.2 AstroDrizzle The New Drizzle Workhorse

[Set-up and Initialize astrodrizzle Parameters](#)
[Create a Static Mask Containing Permanent Bad Pixels](#)
[Perform Sky Subtraction](#)
[Create Separately Drizzled and Registered Images](#)
[Create a Median Image](#)
[Blot Median Image](#)
[Create a Cosmic Ray Mask for Each Image](#)
[Create a Final Distortion-Free Combined Image](#)
[Overriding Instrument-Specific Parameters](#)
[A Note about Photometry and Weights in AstroDrizzle](#)
[AstroDrizzle Memory Usage](#)

AstroDrizzle uses flat-field calibrated images (`flc.flt/flt.fits`, `c0m.fits` for WFPC2) 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 a cosmic ray masks, one for each input `flc.flt/flt.fits` image. Creating cosmic ray masks is a major part of AstroDrizzle, so it is briefly described below (with more detailed information available later in this section):

- Each input `flc.flt/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 `flc.flt/flt.fits` image.
- By comparing each `flc.flt/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.

Regarding WFC3/IR Image

For IR images, the steps "Drizzle separate images," "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 drizzling the final image, or the science array will be compromised by increased noise.

Detailed Description of the AstroDrizzle algorithm

Each subsection below describes each step of the **AstroDrizzle** task in detail.

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 ensure 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.

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. These low values could be caused by 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. For each detector in the list of input images¹, the individual masks are combined to create a master-mask file for use in the final drizzle step. The clipping level for identifying bad pixels is provided by the **static_sig** parameter and is an integer multiple of the RMS below the image's mode.

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 a mask of bad pixels described in the input image's data quality arrays, minus any bits marked as good in the **driz_sep_bits** parameter.

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.

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 valid pixels as bad.

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 `flc.flt/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 offsets in the background levels between exposures prior to combination. The variations in background levels may be due to any number of external sources (e.g., 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 (`flc.flt/flt.fits`, `c0m.fits` for WFPC2) delivered to users from the Archive are not sky-subtracted. However, the sky values calculated by AstroDrizzle in the pipeline are held in the science extension 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 than 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, if 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*, *skysigma*, 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 *flc.fits/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.

Create Separately Drizzled and Registered Images

Each input image is drizzled to create an undistorted copy in the output frame². 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

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.

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.

Create a Cosmic Ray Mask for Each Image

In this step, the software compares each blotted image with its counterpart original `flc.fits/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 `flc.fits/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:

$$|data_image - blotted_image| > scale \times deriv + SNR \times noise$$

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 *flc.fits/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.

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.

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.

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 and the budget of the noise is dominated by read noise and Poisson noise of the background. 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 `flc.fits/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.

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.

The **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 × 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 5.1](#) 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.

Figure 5.1: Processing Time for Three WFC3/UVIS Images using One and Six Cores

Timing: Adriz (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: Adriz (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

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

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

5.3 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 (`flc.fits/flt.fits`). 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/drz.fits`. When an association file is provided as input to AstroDrizzle, all `flc.fits/flt.fits` images belonging to that association are drizzle-combined to create a product with the same rootname (with suffix `drz.fits/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.
 - The weight extension contains an image that is a relative weight map of the output pixels, and is essentially an effective exposure time map.
 - The context extension contains an image that is a map of the output images, and a record of which images contributed to each pixel.
- `drz.fits`: These drizzled multi-extension FITS products are generated from ACS/WFC or WFC3/UVIS 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 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`. Drizzle-combined data from the pipeline should generally be regarded as quick-look products, and users are strongly encouraged to reprocess their images using AstroDrizzle to see if the quality of data can be tuned to the specific scientific needs by 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, 4, 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.

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](#)).

5.4 The DrizzlePac Package

DrizzlePac Tasks
Aligning Images with TweakReg and ImageFindPars
TweakBack
Updating Manually Reprocessed Images
Handling WCS Information With stwcs
blendheaders

drizzlepac is a Python package containing tasks that allow users to align *HST* images, combine them, and perform coordinate transformations on source positions.

astrodrizzle uses image WCS information to combine images. For the most part, this works well for images taken in the same orbit. Images in one visit over several orbits usually adequately aligned 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 Python help files and the [Read the Docs page](#).

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/ refimagefindpars	“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 (<code>flc.fits/flt.fits</code>) 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 <code>flc.fits/flt.fits</code> data quality (DQ) values to 0
mapreg	Provides functions for mapping DS9 region files given in sky coordinates to DS9 region files specified in image coordinates of multiple images using the WCS information from the images.
photeq	A tool to adjust data values of images by equalizing each chip's PHOTFLAM value to a single common value so that all chips can be treated equally by astrodrizzle .
pixreplace	Replace pixels which have one value with another value.
updatenpol	Add the names of the new ACS distortion reference files <code>NPOLFILE</code> and <code>D2IMFILE</code> , 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 that has been manually reprocessed through the calibration pipelines (e.g. CALACS) has 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.

tweakreg (which calls the **imagefindpars** sub-task) is used for aligning images. Please see [Section 7.2.3](#) 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 `flc.fits/flt.fits` images. **astrodrizzle** can then be used to process those updated `flc.fits/flt.fits`. Note that **tweakback** only aligns the WCS in the image headers.

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.

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* can be found in the [notebook tutorials](#).

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.

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 reacquisition 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 [Section 4.4](#).

⚠ *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 (`flc.fits/flt.fits` or `c0m.fits`) 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 make drizzled products before obtaining source positions, as was the case for MultiDrizzle.)
3. A reference frame is determined that can contain all the input images, and to which all images will be aligned to. 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. The offsets between each image and the reference frame are determined using a catalog-matching algorithm similar to **xyxymatch**. 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.

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.

A Note About “Difficult” Images

Discussions in this handbook assume images with enough stellar point sources to allow alignment using the **daofind**-like source-finding algorithm 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).

TweakBack

Drizzled images are 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 input images which were combined to create those drizzled images still contain distortion, so a fit computed using drizzled images cannot be applied to those 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 two 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 (`flc.fits/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.

Updating Manually Reprocessed Images

Data that has been manually reprocessed through the calibration pipelines (e.g. CALACS) has to be updated for compatibility with **drizzlepac** tasks. 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. 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.

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 [WCSSLIB](#) 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.

The tasks available in the STWCS package 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

More detailed information and API on these tasks can be found at the [STWCS Read The Docs page](#).

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 solution implemented for AstroDrizzle is the **blendheaders** task. This task creates 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 science extension (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.

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.

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.

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.*

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.

Configuration files can 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.

A sample configuration file for **tweakreg** is shown below.

```
_task_name_ = tweakreg
input = *flt.fits
refimage =
expand_refcat = False
enforce_user_order = True
exclusions =
writecat = True
clean = False
interactive = True
verbose = False
runfile = "tweakreg.log"

[UPDATE HEADER]
updatehdr = False
wcsname = TWEAK
reusename = False

[HEADERLET CREATION]
headerlet = False
attach = True
hdrfile = ""
clobber = False
hdrname = ""
author = ""
descrip = ""
catalog = ""
history = ""

[OPTIONAL SHIFTFILE OUTPUT]
```

```
shiftfile = False
outshifts = shifts.txt
outwcs = shifts_wcs.fits

[COORDINATE FILE DESCRIPTION]
catfile =
xcol = 1
ycol = 2
fluxcol = ""
maxflux = None
minflux = None
fluxunits = counts
xyunits = pixels
nbright = None

[REFERENCE CATALOG DESCRIPTION]
refcat =
refxcol = 1
refycol = 2
refxyunits = degrees
rfluxcol = ""
rmaxflux = None
rminflux = None
rfluxunits = mag
refnbright = None

[OBJECT MATCHING PARAMETERS]
minobj = 15
searchrad = 1.0
searchunits = arcseconds
use2dhist = True
see2dplot = True
separation = 0.5
tolerance = 1.0
xoffset = 0.0
yoffset = 0.0

[CATALOG FITTING PARAMETERS]
fitgeometry = rscale
residplot = both
ylimit = None
labelsize = 8
nclip = 3
```

```
sigma = 3.0
```

```
[_RULES_]
```

Chapter 6: Reprocessing with the DrizzlePac Package

Chapter Contents

- [6.1 Beyond the Standard Calibration Pipeline](#)
- [6.2 Image Alignment](#)
- [6.3 Running AstroDrizzle](#)

6.1 Beyond the Standard Calibration Pipeline

DrizzlePac is written in **Python** (with core drizzle algorithms written in **C**). Its interface is a departure from previously historically conventional **IRAF** usage. To learn more, please refer to [the DrizzlePac Jupyter Notebook Tutorials](#) for an introduction to using **Python** 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 using the best available software to calibrate data with the best available reference files. If it has been a long time since their data was retrieved from the archive, users are encouraged to re-download it to ensure that the data contains the most up-to-date header information and calibrations.

There are presently two major steps in the processing: (1) Calibration of individual datasets using instrument-specific calibration software, such as **calacs** for ACS and **calwf3** 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 automatic reprocessing is not ideal may include when a user has a preference for self-made calibration reference files, or the use of non-default calibration switches, or when using non-default software parameter values. Reason for these actions could be to improve hot pixels and cosmic ray removal or to deal with image persistence/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 beneficial. Drizzled pipeline data is created with conservative values which are stored in the MDRIZTAB reference file. More can be read about **AstroDrizzle** in the pipeline [here](#).

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. Further information can be found in this example notebook on [optimizing image sampling](#).

6.2 Image Alignment

[Alignment Error Sources](#)
[Processing Large Images](#)
[Using TweakReg for Image Alignment](#)
[Aligning Under-Sampled Images](#)

Flat-field calibrated images from the calibration pipeline have been updated to incorporate the full distortion model that is stored in SIP header keywords and the updated CD matrix. This is done in the pipeline using the **updatewcs** task in the **STWCS** package. For ACS/WFC3, 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.

Alignment Error Sources

Sources of Alignment Errors:

- 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.
 - 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 mis-aligned at that level.
 - On rare occasions observations do not properly lock onto the guide stars which causes drifting 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 these bad observations and realign those that are still useful.
- An inability to get accurate centroids on objects like extended sources, targets obscured by dust, or 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](#).

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 the size of their data and the amount of memory available in the processing computer. More information on **AstroDrizzle** Memory usage can be read about in [Section 5.2.11](#).

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`, `*flc.fits`, etc.) 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. The residual shifts between the visits are due to uncertainties in the guide star positions as discussed in [Section 4.4](#). Smaller-scale residual offsets could also occur during guide star re-acquisitions for observations taken in a multi-orbit visit.

TweakReg is a WCS-based task, not pixel shift-based like previous software versions like **MultiDrizzle**. For images with residual offsets with respect to a reference image or catalog, WCS information in their headers are modified by **TweakReg** to "tweak" their WCS information 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

A matched sources list is a list of common sources found in an input image and the reference image or catalog.

TweakReg performs the following processing steps to determine a fit between each an input image and a specified reference WCS:

1. It builds a catalog of source positions for each input image using one of these modes:
 - a. Using a DAOFIND-like algorithm called **ImageFind** to detect stellar sources (the default mode).
 - b. Using a user-supplied source catalog.
2. The WCS from a reference image is selected from one of these options:
 - a. The first input image (the default mode)
 - b. An image specified by the user. Possibilities include:
 - i. One of the input `FITS` files - perhaps an image that has the most overlap with other images.
 - ii. A different type of image - perhaps from another *HST* instrument or a different telescope.
3. A reference catalog of sources is identified using either:
 - a. A source catalog from the previously chosen reference image.
 - b. A catalog of source positions on the sky (R.A., Dec.) provided by the user.

4. All source positions for the input `FITS` images and reference source positions are converted to X, Y positions in the reference WCS tangent plane using all available distortion corrections provided by various distortion reference files.
5. For each input image and reference image pair, the difference in the source positions are represented in a two-dimensional histogram, allowing the determination of an initial offset based on the histogram peak in X and Y .
6. Algorithm based on the **xyxymatch IRAF** task is used to match input source positions and reference image source positions using the initial offset from 5.
7. For each input image, a fit to determine the most accurate offsets is performed on the matched sources lists; 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.
8. When the user is satisfied with the fit, **TweakReg** can be run a final time with `updatehdr` set to True. This will update the headers of the input images with their new WCSs that put all images in the same coordinate frame.
9. A headerlet can also be (optionally) created from the updated input image WCS. More details on the current headerlet's available for pipeline-drizzled *HST* data can be found [here](#).

Catalog Matching between Input-Reference Pairs

A widely utilized method for computing offsets between images begins with identifying sources in each image. For each input 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 contain 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:

1. The filename of a single image.
2. The filename of an association (ASN) table.
3. Wild card specification for files in directory (i.e., `*flt.fits`).
4. Several filenames separated by a comma.
5. 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`).
6. A Python list.

When comparing the input images, the **TweakReg** software defines the reference frame either by:

1. The first image from the list of input images. (default)
2. A catalog derived from a reference image specified by the user. The user can pick a specific reference image by setting the `ref_image` parameter in **TweakReg**.
3. A source catalog provided by the user.

Aligning Under-Sampled 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 under-sampled detectors on *HST*: WFC2, 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. They exhibit very systematic residuals up to ± 0.1 pixels. These residuals arise because typical centroid-ing and PSF-fitting applications tend to move the position of a star in an under-sampled detector towards the center of the pixel in which the star is brightest. In order to avoid this bias, one must explicitly take the under-sampling of the detector into account. One method for doing this is the **ePSF (effective PSF) method** of Anderson and King (2006), an example of residuals found using their method on the same set of images can be seen below in [Figure 6.2](#). A newer citation is also available in [Bellini et al, 2018](#).

Figure 6.1: Residuals from Aligning Two WFC3/IR Images Using TweakReg Source-Finding Algorithm

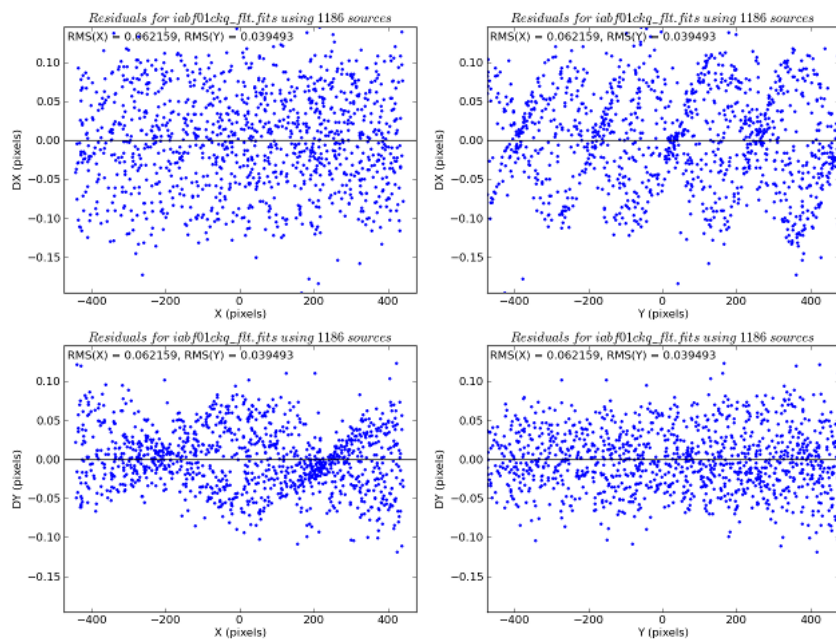
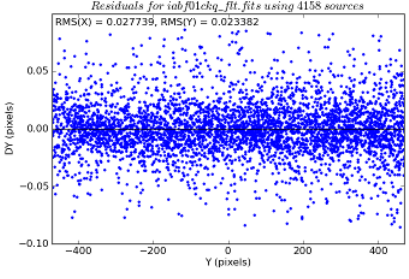
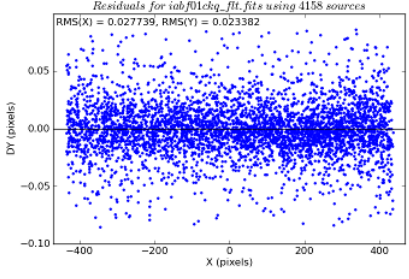
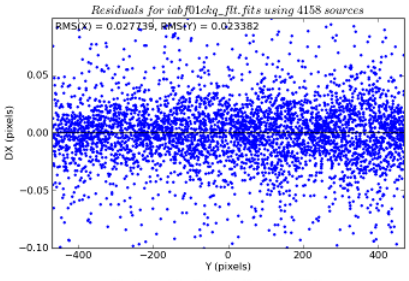
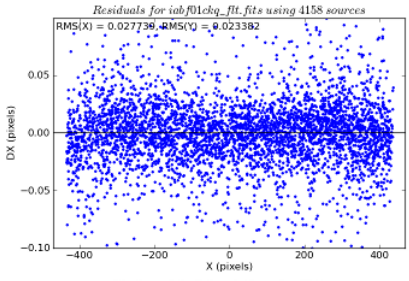


Figure 6.2: Residuals from Aligning two WFC3/IR Images with TweakReg Using Positions Determined with the ePSF Method



6.3 Running AstroDrizzle

[Sky Subtraction Considerations](#)
[Cosmic Ray Rejection](#)
[Selecting the Optimal Scale and Pixfrac](#)
[Controlling the Bit Mask](#)

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. This section provides some considerations on key aspects of the **AstroDrizzle** processing that can help guide the user in deciding on the best way to reprocess their images.

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 needed. Incorrect sky subtraction by **AstroDrizzle** can slightly bias the cosmic ray identification, but more noticeably 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 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 values 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 they want a record of the custom sky subtraction values applied to each image, in which case, a list with a custom sky value for each chip could be written to a file. This file, tagged 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.

Further information on sky matching can be found in the [Jupyter Notebook on Sky Matching](#).

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 themselves, then they must be sure the images are left with sky values that match.

Cosmic Ray Rejection

AstroDrizzle processes 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 give 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 (DQ) and image (SCI) extensions from the same `flat.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.

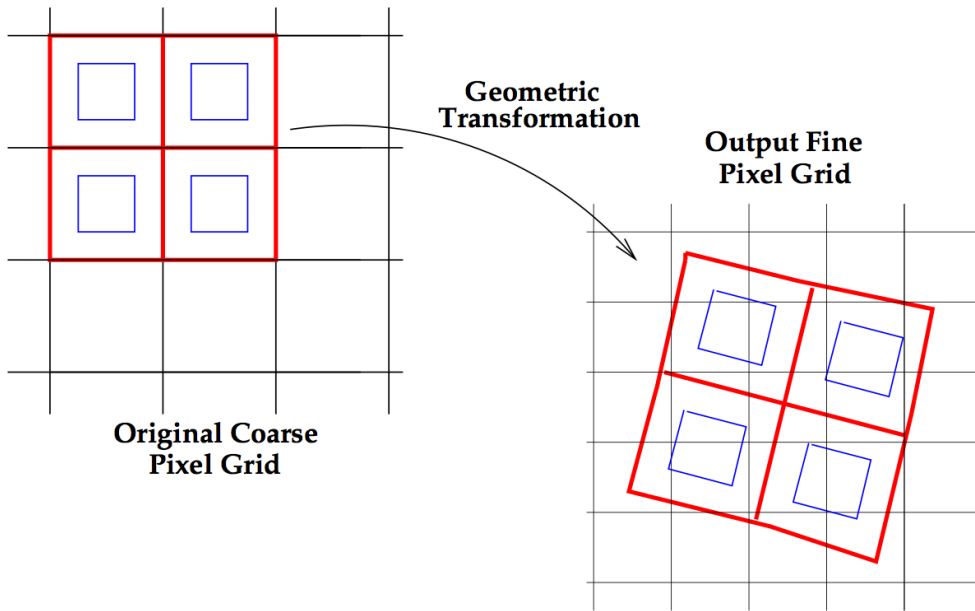
Selecting the Optimal Scale and Pixfrac

In combining images, the Drizzle algorithm maps pixels from input images into pixels in a sub-sampled 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, **Astrodrizzle** 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. Sub-sampling 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. With only a few images, a large *pixfrac* will 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. A common exception is the single image case, where the goal is to primarily remove 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.

Figure 6.3: A schematic representation of Drizzle. This is the same as Figure 3.2, repeated here for convenience.



In [Figure 6.3](#) above, 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. When a user has few images (i.e. 2 or 3), but has used one of the standard dither patterns which does an optimal sub-pixel dither, they 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 for 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 in a typical area of the image does not go significantly below 0.2. 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 with large numbers of images 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 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. 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* $> \sim 0.7$, but are free to experiment. To summarize, when experimenting with *pixfrac* and scale users should keep these points in mind:

1. For sub-pixel 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.

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 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 and in the "final drizzle" step. 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.

In early 2017, the ACS instrument team changed the definition of data quality (DQ) flags populated in the calibrated FLT/FLC files. New calibration techniques now make it possible discern between unstable and stable hot pixels, the later of which are corrected by 'calacs' when subtracting the dark. Thus, pixels identified as hot and stable (DQ flag=16) may now be treated as 'good' data when drizzling, and those identified as unstable (DQ flag=32) should be treated as 'bad'. A new MDRIZTAB reference table (16r12191j_mdz.fits) was delivered in June 2017 and contains a set of default parameters for combining exposures with AstroDrizzle. With changes to the DQ flag definitions, the parameters *driz_sep_bits* and *final_bits*, which define DQ flags for drizzle to ignore (e.g. to treat as good), are now set to a value of 336 (the sum of 16+64+256) so that stable hot pixels, warm pixels, and full-well saturated pixels will not be rejected when combining exposures. For details, see [ACS ISR 2017-05](#).

The WFC3 instrument team implemented a similar change to the DQ flag definitions in December 2018, and an updated MDRIZTAB reference file (2ck18260i_mdz.fits) reflects the new recommended drizzle parameter settings such that DQ flag values 16, 64, and 256 are treated as good pixels. These new flags are valid for UVIS observations obtained after Nov 08 2012, when the dark calibration program began using post-flash to mitigate hot pixel trailing due to poor charge transfer efficiency at low background levels. A description of the new UVIS bad pixel tables is described in [WFC3 ISR 2018-15](#).

AstroDrizzle was designed for use with multiple instruments, so the default value for the *driz_sep_bits* and *final_bits* parameters are 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 **AstroDrizzle** these parameters may be given as either the sum of the DQ flags or as a comma-separated list.

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 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 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 and the cosmic ray masks. 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 when the parameter *driz_cr_corr* is **True**, 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 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: Data Quality Checks and Trouble Shooting Problems

Chapter Contents

- [7.1 Inspecting the Drizzled Products from MAST](#)
- [7.2 Verifying TweakReg Solutions After User Reprocessing](#)
- [7.3 Inspecting Drizzled Products after User Reprocessing](#)

7.1 Inspecting the Drizzled Products from MAST

[Inspecting the Drizzled Products from MAST](#)
[Examine the Drizzled Science Image](#)
[Verify the Image Header Sky Keyword](#)
[Examine the Data Quality Array](#)
[Examine the Drizzled Weight Image](#)
[Examine the Headerlet](#)

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, currently MAST only creates drizzled products for images obtained in a single visit, so additional visits may only be combined together by reprocessing. In the future, however, the HAP will produce multi-visit mosaics.

While single visit data with small POSTARG dithers (like the 4-point dither box) are usually aligned to better than 0.1 pixels, with an accuracy of about 2 to 5 milliarcseconds. The drizzled products for most, though not all instruments such as SBC, 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 POSTARG dithers (for example, to create mosaics), 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.

Poor alignment can lead to poor cosmic ray rejection, and flagging of true sources which can lead to compromised photometric accuracy. 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 7.3](#).

Examine the Drizzled Science Image

Do the drizzled products look "clean?"

The science extension `"image_drz.fits[sci,1]"` or `"image_drc.fits[sci,1]"` from the drizzled MAST data products should be inspected for any obvious anomalies and compared with the calibrated data products, `"image_fl*.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](#).

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 7.3.1](#).

Are the PSFs "round" and "narrow," as expected?

Elongated PSFs in drizzled images may indicate that the alignment needs to be improved. A task called **TweakReg**, part of the **DrizzlePac** package, can be used to improve the image alignment for data taken within a single visit, across multiple visits, across most filters, and across most detectors. 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](#). It may also be related to bad guiding, so checking the success of the guide star lock can be useful.

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*. Information about the ACS PSF may be found in [Section 5.6](#) of the *Advanced Camera for Surveys Instrument Handbook*.

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 6.3.3](#).

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.

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 "[How Distortions are Represented in AstroDrizzle](#)" in [Section 4.2.2](#).

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 re-acquisitions from orbit to orbit. In other, less common instances, losing the lock on the 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 3.3](#). 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.

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 MDRIZSKY header keyword in the science extensions of `flt.fits/flc.fits` images.

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. However, for observations of targets that fill the field of view the sky background may be overestimated. An inaccurate sky subtraction could compromise the accuracy of the cosmic ray rejection which in turn may impact the accuracy of the 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 (much larger 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** then 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 `FITS` images prior to processing.

Examine the Data Quality Array

Did the pipeline flag an excessive number of pixels as 4096 (the cosmic ray DQ flag)?

Users are encouraged to blink the science array of the calibrated `FITS` images with its corresponding data quality (DQ) array. Cosmic rays that are flagged during pipeline processing are assigned a flag value of 4096, and the accuracy of these flags should be inspected. 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.

Reprocessing the drizzled products 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 instead be computed by the user during reprocessing.

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. `drz.fits[wt,1]`) are, by default, 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 4.2.2](#).

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 which reflect detector artifacts and cosmic rays that were excluded. If the sources themselves are flagged as cosmic rays then 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.

Examine the Headerlet

The headerlets included in the `flt.fits/flc.fits` file dictate what the astrometric solution of the pipeline drizzled image will be. Thus, it is necessary to check which headerlet/astrometric solution is the active solution (that is, which one was used when the images were drizzled in the pipeline). This can be checked by looking at the value of the WCSNAME keyword in the header of the SCI extension of either the drizzled or exposure data. If the WCSNAME keyword contains 'FIT-GAIA', then sources in the images were detected and matched with a Gaia catalog, and should therefore have high quality absolute astrometry. However, it is possible that the quality of the matching/fitting is insufficient for science goals, or the fit solution may be erroneous. In cases when working with multiple datasets, only some of the images may have been fit to Gaia, and the full set of images may not be relatively aligned to each other. In these situations, it can be beneficial to switch the solution to one of the other options provided in the headerlets. The jupyter notebook "[Using updated astrometry solutions](#)" discusses these headerlets further, and shows how to switch which solution is active. If the Gaia solutions prove insufficient, then switching to the GSC-240 solutions and/or running **TweakReg** can rectify the issues.

7.2 Verifying TweakReg Solutions After User Reprocessing

[Examine the Fit Residuals](#)
[Examine the Astrometric Residuals](#)
[Examine the Vector Plot](#)
[Examine the 2-Dimensional Histogram](#)
[Verify the Number of Matches Used to Compute the Solution](#)

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.

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 POSTARGs are specified to offset the telescope from the commanded R.A. & Dec. In these cases, small residual rotations on the order of a few thousandths 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 [Section 4.4](#).

To estimate the residual offsets before running **TweakReg**, users may display the images in **ds9**, align by WCS, and then blink the different frames. For more details on inspecting the WCS alignment, refer to section 3 in this [notebook](#).

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 then 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.

If *imagefindpars* (used by **TweakReg**) is unable to produce source catalogs suitable for image alignment, it may be necessary for the user to import their 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.

Is the fit RMS small?

For a good solution with a large number of sources (a few hundred at least), 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.

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 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. 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. 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.

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 FITS images and over plot the matched source lists - an example of how to do this can be seen in step 2f of the [Align Multiple Visits](#) notebook. If sources are only matched in one portion of the detector, this may indicate that the sigma clipping parameters *nclip* and *sigma* were assigned too aggressively. 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 5.4.2](#).

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. 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 `FITS` images and over plot the matched source lists to ensure that the fit is based on "real" objects and not artifacts.

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.

Verify the Number of Matches Used to Compute the Solution

Does TweakReg crash with the message "not enough matches found?"

When TweakReg finds fewer 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 that are 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.

7.3 Inspecting Drizzled Products after User Reprocessing

[Examine the Drizzled Science Image](#)
[Examine the Reprocessed Drizzled Weight Image](#)

Are the reprocessing task parameters optimal?

The same set of checks discussed for Archival data should be performed for reprocessed data. 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 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 under-sampled.

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 (DQ) 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.

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 **False** 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* is set to **True**, 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., `*drz.fits[sci,1]` or `*drc.fits[sci,1]`) with the science array derived during reprocessing (i.e., `*drz_sci.fits` or `*drc_sci.fits`). Look over the below set of questions which are addressed above:

- Do the new 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?

A further question about the new reprocessed drizzled data: 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.

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. 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. 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.

Examine the Reprocessed Drizzled Weight Image

Similar to the questions above for the pipeline processed archival **drz.fits* or **drc.fits* images:

- 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?

Further, for the reprocessed images, 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 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 for situations when only a few input images are being combined. The two **bits** parameters indicate 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 a *final_pixfrac* value that is too aggressive. For routine observations containing several dithered images, the *pixfrac* or "drop" size, should be between 0.5 and 1.0 all 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 sub-pixel 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 and a more routine and conservative use of *pixfrac* and *final_scale* is usually in order.

Chapter 8: DrizzlePac Examples

Chapter Contents

- [8.1 Jupyter Notebook Introduction](#)
- [8.2 Practical Tutorials](#)

8.1 Jupyter Notebook Introduction

This version of the **DrizzlePac** Handbook uses Jupyter Notebooks for the practical examples. The notebooks contain live code and visualizations, along with the traditional narrative text, making them an ideal training exercise for users. Those unfamiliar with Jupyter Notebooks can find information about installation and usage at [the Project Jupyter website](#).

Each tutorial includes blocks of code demonstrating how to download calibrated data from the [MAST archive](#), how to align frames and update the image world coordinate system (WCS), and how to enhance the scientific value of the drizzled data products using advanced reprocessing techniques.

All the [DrizzlePac Jupyter Notebooks](#) are hosted at [the Space Telescope Science Institute Notebook repository](#). For additional assistance with the **DrizzlePac** notebooks, users may submit a ticket at the [STScI Help Desk portal](#) (preferred) or send an email to help@stsci.edu.

8.2 Practical Tutorials

[Improving Astrometry Using Alternate WCS Solutions](#)
[Aligning HST Images to an Absolute Reference Catalog](#)
[Aligning Multiple HST Visits](#)
[Aligning Deep Exposures of Sparse Fields](#)
[Creating HST Mosaics Observed with Multiple Detectors](#)
[Using Sky Matching Features for HST Mosaics](#)
[Optimizing the Image Sampling for Sub-pixel Dithers](#)
[Drizzling New WFPC2 FLT Data Products](#)
[Masking Satellite Trails Prior to Drizzling](#)
[Using DS9 Region Files in TweakReg](#)

This section provides a summary of each notebook to help identify which example will be useful for specific issues and science goals.

Improving Astrometry Using Alternate WCS Solutions

As mentioned in [Chapter 1](#) and discussed in detail in [Chapter 4](#), a new type of astrometry for *HST* images has been adopted as of December 2019 as part of an effort to automatically align to Gaia. This notebook guides users through the new features available in images from MAST including headerlets and provides code to change between the various world coordinate solutions (WCS), to evaluate the image alignment of a given WCS, and to change the WCS of a drizzled image. For those who need to learn about and manipulate the new MAST astrometry, this notebook is an ideal place to start and should answer most common questions.

Aligning HST Images to an Absolute Reference Catalog

This notebook details a workflow that will align *HST* images to an external catalog, which can be useful for many science applications, especially when absolute astrometry is necessary. The example uses WFC3/UVIS images of NGC 6791 and walks through downloading the *HST* images and catalogs (e.g. SDSS or Gaia) with **astroquery**, using **TweakReg** to align to the catalog positions, and then creating a final drizzled image with **AstroDrizzle**.

Aligning Multiple HST Visits

This notebook contains an example of aligning images from multiple visits. In this case, three ACS/WFC images of the globular cluster NGC 104 taken over a three month period with different telescope orientations are downloaded programmatically from the MAST archive, aligned with **TweakReg**, and then processed by **AstroDrizzle** into a combined image. The notebook also defines different **TweakReg** parameters and shows how to optimize them, and explains the outputs of **AstroDrizzle** and how to use them to evaluate the final image.

Aligning Deep Exposures of Sparse Fields

This notebook discusses one way to align deep exposures of sparse fields that is helpful when there are more cosmic rays or other artifacts than point sources in a given field. This example uses four ACS/WFC images from the COSMOS 2-Degree ACS Survey and describes several key parameter changes to the photometry code in **TweakReg** that detects the sources necessary to align images. These modifications to the source detection can help exclude the spurious detections from cosmic rays and include small compact sources like background galaxies, which enables the alignment of the images. The final combined image from **AstroDrizzle** is then evaluated.

Creating HST Mosaics Observed with Multiple Detectors

The Aligning Mosaics notebook walks through a reduction of the Eagle Nebula (M16) with both UVIS and IR images from WFC3, although the theory also works for ACS/WFC images. It explains the observation strategy and the dither patterns used, and will programmatically download the data from the MAST archive. The notebook explains manipulating **TweakReg** parameters to better align images and how to evaluate and troubleshoot the resulting solution. Then it discusses features in **AstroDrizzle** to use when combining images into a mosaic and the procedure for combining multiple filters.

If the full notebook is run, it will produce a complete mosaic image of the Eagle Nebula with *HST* WFC3 images.

Using Sky Matching Features for HST Mosaics

This notebook explains the four options available in **AstroDrizzle** to match the sky background between images when creating a larger mosaic. It downloads WFC3/IR images of the Horsehead Nebula and describes how the images are combined, applies each of the four sky background methods, and then compares the results, providing recommendations about how to get the best image.

At the end of the notebook, a mosaic image of the Horsehead Nebula will be produced.

Optimizing the Image Sampling for Sub-pixel Dithers

This example walks through the process of recovering some of the information lost in undersampled images with **AstroDrizzle**. WFC3/IR images of NGC 3370 are downloaded and then drizzled with various scale and pixel fraction parameters. The results are evaluated to determine the optimized parameters, balancing between the extremes of failing to recover information by remaining undersampled and degrading the combined image by introducing noise. This type of analysis is especially beneficial for the lower resolution of WFC3/IR images, but can be used any time proper dithering is performed.

Drizzling New WFPC2 FLT Data Products

This notebook shows how to work with a new type of WFPC2 calibrated data product in MAST. The new products combine the previously separate files containing the science array `c0m.fits` and data quality array `c1m.fits` into a new file with suffix `flt.fits`, similar to calibrated images for ACS and WFC3. These `flt` files have now been corrected for differences in the inverse sensitivity the science arrays of each chip using the software '[photeq](#)' so that a single PHOTFLAM (or PHOTFNU) value may be used for photometry. This means that the SCI arrays from different chips may now be drizzled together in a mosaic. Additionally, the header WCS now includes improved absolute astrometry by aligning to Gaia, when possible.

This example uses observations of Omega Centauri (NGC 5139), where two exposures have been dithered to place the same stars on the WF2 and WF4 chips to measure relative photometry across the two chips. After using **TweakReg** to realign the images, we show how to combine the images with **AstroDrizzle**.

Masking Satellite Trails Prior to Drizzling

This notebook explains how to mask satellite trails, which are a common artifact in *HST* images. Images of galaxy cluster MACSJ0717.5+3745 from the *Hubble* Frontier Fields, chosen because they contain satellite trails, are downloaded and used to demonstrate two techniques: 1) An automated tool developed by the ACS team to find and mask satellite trails in the DQ array, and 2) manually identifying image artifacts with DS9 region files and using the regions to mask the DQ array. The first technique is often quicker due to the automation, but the second allows for more control and for masking other types of image artifacts, such as dragon's breath and blooming. When the DQ arrays are appropriately masked, **AstroDrizzle** knows to avoid those pixels when creating the combined image, leading to a final result that is not impacted by the satellite trails.

Using DS9 Region Files in TweakReg

This example explains how to use DS9 region files to include and exclude sources in **TweakReg**'s internal source detection photometry. The notebook first demonstrates how to download two specific ACS/WFC images of MACSJ1149.5+2223-HFFPAR without having to download the entire association, creates source lists with **TweakReg**, and explains the format of DS9 region files and how to read them using Python. The notebook walks through writing exclusion and inclusion text files with their unique and very particular formatting so that **TweakReg** can utilize the DS9 region files. The regions can exclude areas of the image from **TweakReg**, which can be useful to remove galaxy centers or objects that are not point sources from the source list. They can also be used to include, so that only the sources inside the DS9 regions are used by **TweakReg**. Lastly, the exclusion and inclusion functions can be combined to give complete control over which areas of the images are in the source list, although care should be taken with this as the formatting is not straightforward. Examples of the DS9 region files and the exclusion files used by **TweakReg** are provided in the repository.

The introduction of this notebook includes a brief explanation of DS9 and its region files, including some resources for those unfamiliar with this tool.