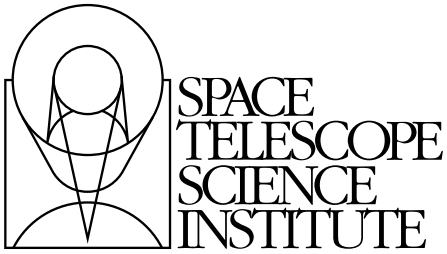
Version 3.0
January 2009

The MultiDrizzle Handbook

A Guide for Combining HST Images



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

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

How to Get Started

More technical instrument specific documentation, such as that provided in the Instrument Handbooks, can be accessed from:

http://www.stsci.edu/hst/HST_overview/documents

User Support

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

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

MultiDrizzle Handbook History

Version	Date	Editors
3.0	January 2009	Fruchter, A., Sosey, M., Hack, W., Dressel, L., Koekemoer, A. M., Mack, J., Mutchler, M. and Pirzkal, N.
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.
1.0	December 2000	Koekemoer, Anton M., et al.

Citation:

In publications, refer to this document as:
Fruchter, A. and Sosey, M. et al. 2009, "The MultiDrizzle Handbook", version 3.0, (Baltimore, STScI)

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

Table of Contents

Acknowledgments

The information contained in this Handbook represents the cumulative experience and contributions of many members of the STScI community, including the WFPC2, WFC3, ACS, NICMOS and STIS instrument groups, the Observatory Support Group and the Science Software Branch. The drizzle code, which underlies all the analysis routines described here, was originally developed by Richard Hook and Andy Fruchter. Initial implementation of drizzle and its supporting CL scripts in IRAF/STSDAS was coordinated with the Software Support Group at STScI, in particular Ivo Busko and J.-C. Hsu.

Since then numerous people have worked on this project to create what is now the MultiDrizzle code, including but not limited to, Anton Koekemoer, Warren Hack, Andy Fruchter, Richard Hook, Nadezhda Dencheva, Perry Greenfield, Michael Droetboom and Chris Hanley.

This version of the MultiDrizzle Handbook was edited by Andy Fruchter, Megan Sosey and Warren Hack, with contributions from many others at STScI, in particular Linda Dressel, Anton Koekemoer, Jennifer Mack, Max Mutchler and Nor Pirzkal and with the technical assistance of Susan Rose, Michael Droetboom.

We at the Institute also thank the many users of MultiDrizzle whose feedback has been invaluable in improving the software.

CHAPTER 1:

Introduction

In This Chapter...

1.1 How to Use The MultiDrizzle Handbook

The MultiDrizzle Handbook is designed to help users learn about combining dithered observations. The information in this handbook has been drawn from the various individual HST instrument handbooks, as well as a considerable number of other separate documents available on the STScI Web site:

http://www.stsci.edu/hst/HST_overview/documents/

In Chapter 8, a bibliography of the relevant material is presented at the end of this handbook. The information in many of these older documents is out-of-date and has been updated as necessary when included in this handbook. They are referenced primarily for the sake of bibliographic completeness, and if they are consulted this caveat should be borne in mind.

Dithers are observations where the telescope is offset between exposures. The most common way of combining dithered HST images is to use the "Drizzle" algorithm (Fruchter & Hook 2002). This is predominantly done through the "MultiDrizzle" program (Koekemoer et at. 2002), which also performs many of the steps necessary between acquiring your data from the HST archive and combining it using the drizzle algorithm.

Users of HST range from the novice to the experienced black belt drizzler, So different users may want to use these pages in different ways. These are some suggestions for how to proceed:

- If you do not have a good grasp of the ideas behind dithering observations or drizzling data, we suggest you start with Chapter 1 "Dithering Basics" and Chapter 3 "How the Drizzle Algorithm Works". You may then wish to skip right to the Chapter 6 "Real-World Examples" and work backwards to the more detailed descriptions of parts of the MultiDrizzle code as needed.

- If you have experience with dithered data, and have used Drizzle or MultiDrizzle before, but are not sure you are up-to-date with the latest software, you may wish to go directly to the example pages to see how to use the latest version of MultiDrizzle. You can then use other chapters as required.
- If you have used MultiDrizzle extensively in the past, or have your own code for combining dithered images and are interested in the details, you may find most of the information you need in the sections detailing the astrometric header information and the MultiDrizzle parameters.

1.2 What is New in Multidrizzle?

MultiDrizzle has undergone a number of internal changes since its last major release. Fortran has been converted to C; python code has been improved and better documented. Additionally a fundamental change has been made in the philosophy used to handle image distortions and astrometry. In the past, MultiDrizzle used a set of coefficients separate from the image world coordinate system (WCS) to define nonlinear distortions in the image, and even some small linear corrections in certain cases. The distortion information came from a separate file containing a table called the IDCTAB (or instrument distortion coefficients table), to supply distortion coefficients to the drizzle program. By contrast, the new version of MultiDrizzle incorporates the distortion information directly into the WCS using a standard called the "simple image polynomial" or SIP standard (Shupe et al., 2005). This standard has previously been used to describe the geometry of the Spitzer Space Telescope images, and we expect it to become a FITS standard in the near future. As MultiDrizzle is run by the HST archive during calibration-on-the-fly, whenever a user receives HST images in the future, they will have the distortion information incorporated into the headers of their calibrated data using FITS standard conventions. This means that users will find it easier to write their own programs to combine HST data which take full advantage of our knowledge of distortions in the image plane. Indeed, a number of programs already in use in the astronomical community, such as SAOimage and DS9, are already able to read the SIP coefficients and use them to convert between pixel and sky coordinates.

In the longer term, this change of philosophy will allow even greater improvements to the way we handle image combination and astrometry. For instance, we plan to distribute code that will compare a catalog of object positions derived from an HST image with an astrometric catalog supplied by the user. The code will then calculate a "headerlet" that will contain the changes to the WCS necessary to bring the HST image coordinate system into agreement with the astrometric catalog. The "astrometric catalog" could equally well be a set of positions derived from other HST images. Thus this method will provide a powerful tool both for combining and creating mosaics of HST images as well as aligning them with external catalogs.

CHAPTER 2:

Introduction to Dithering

In This Chapter...

2.1 What is Dithering?

An increasingly popular technique in UV optical and IR imaging observations involves the use of dithering or spatially offsetting the telescope by shifts that are generally small relative to the detector size, thereby moving the target to a number of different locations on the detector. Two of the main strategies involve offsets by an integer number of pixels to facilitate the removal of bad pixels, and 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, where the PSF is so small that it is significantly undersampled by the majority of the primary science instruments.

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, and involves observational considerations and methods of data analysis that are beyond the scope of this document. However, the techniques we discuss in this document are essential to mosaicing with HST.

2.2 Benefits of Dithering

Dithering of HST observations is hardly new; the primary data acquisition modes of the GHRS and FOS involved both sub- and multi-diode offsets to obtain well-sampled data along the spectral dimension without gaps resulting from the presence of a few dead diodes. However, dithered observations in imaging-mode became routine only after the dramatic improvement in the HST optics that corresponded to the installation of COSTAR/WFPC2. Dithering often provides considerable benefits to the science program, specifically the following:

- Dithering can reduce the effects of pixel-to-pixel errors in the flatfield or spatially varying detector sensitivity.
- Integer shifts of a few pixels allow the removal of small scale detector defects such as hot pixels, bad columns, and charge traps from the image.
- Non-integral (sub-pixel) dithers allow the recovery of some of the information lost to undersampling by pixels that are not small compared to the point spread function.

The third point is particularly important in the case of HST imaging, since nearly all imaging instruments on HST are unable to take full advantage of the resolving power of the optics. This is because the instrument designers had a choice 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 the primary imagers on HST. The width of a WF pixel on WFPC2, at about 0.1 arcseconds, is already comparable to the full-width at half-maximum of the optics in the I-band and substantially exceeds it in the blue. The NICMOS camera 3 detector similarly undersamples the image over much of its spectral range. While the ACS/HRC does adequately sample the PSF at optical wavelengths, this comes at the cost of a drastically reduced field of view (1/50th the area of ACS/WFC). Even the ACS WFC, and the WFC3 UVIS and IR channels, have pixels comparable in width to the full-width at half maximum (FWHM) of the PSF, where ideally one would like a minimum two samples per FWHM for the full recovery of the image resolution. By dithering, one can recover these “missing” samples; however, one cannot completely undo the small blurring produced by a larger pixel. Nonetheless, dithering does substantially improve the final image quality, while simultaneously allowing improved removal of detector defects.

2.3 Costs and Drawbacks of Dithering

While dithering provides substantial benefits, there are a number of trade-offs that must be understood and considered when deciding whether or not to obtain dithered data. These are described more fully later in the document but are summarized here:

- Obtaining the final combined data product will require special reductions and thus more work on the part of the observer.
- Some extra spacecraft overhead time will be incurred, and observers will need to judge if this is significant by actually running tests on various scenarios using the RPS2 scheduling software.
- If longer exposures are broken into shorter exposures to obtain more dithers, the user may suffer an increase in read noise. The increased volume of data will fill the data buffer faster, possibly complicating the observing.
- If the primary science goal is to measure differential changes over time, as in time-series photometry, then dithering can in some cases complicate the resulting analysis due to intra-pixel sensitivity variations in the detectors. This was primarily a concern with NIC3, and should not be an issue for WFC3 which has not shown the same intra-pixel sensitivity as NICMOS during it's ground testing.
- Elimination of cosmic rays may be slightly compromised, especially if one has only a few sub-pixel offsets with just one image at each location.

For most observing programs on HST the potential drawbacks to dithering are outweighed by the scientific benefits. However, in specific instances it might be possible that the above drawbacks are deemed too severe, as in programs with very few available orbits. If you have questions about how your particular program would be affected by dithering, and you do not find the answers in this document, please feel free to get in touch with help@stsci.edu or your Contact Scientist if one has been assigned.

2.4 Observational Dither Strategies

Here we provide, broad, general recommendations aimed at guiding observers toward different sampling strategies, depending upon the type of science that will be extracted from the data. Please note that these are guidelines only and in no way intended as solid rules. There will likely be science programs that do not fit exactly into any one of these categories, or that have different requirements.

2.4.1 No Dithering

Very Short Exposures: If each target is observed for less than a few minutes then the extra overhead for dithering can significantly impact the overall S/N, thus offsetting any advantages gained by dithering

Critical Photometric measurements: For high-precision time-dependent photometric monitoring, dithering may introduce additional complications due to intra-pixel sensitivity variations, thus some observers may prefer to obtain all the images at a single pointing location.

2.4.2 Simple Dithering

Even if sub-pixel dithering is not necessarily required, dithering each exposure by an integer pixel shift reduces the impact of hot pixels. To improve spatial sampling, 2- or 3-point sub-pixel dithering may be used, depending on how much overhead can be afforded. It is possible to do CR rejection with a single image at each dither point, although two or more images at each location will yield more robust rejection. For programs up to about one orbit per target, at least two to three exposures should be obtained to facilitate cosmic ray rejection. If one is interested in targets throughout the field, rather than one single star, 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.

2.4.3 "Full" Dithering

If improved spatial sampling is desired on programs of two or more orbits per target/filter combination, then a "full" 4-point dither is recommended (e.g., providing 1/2 pixel sub-sampling along both detector axes). Most of the sub-pixel information in an image is recovered by a four-point dither. However, for deep programs even larger numbers of dithers can be considered. 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 4 points.

2.4.4 Dithering for Parallel Images

It is not always possible to obtain optimal dithers simultaneously for primary and parallel instruments due to the large separation and generally different pixel scales. Uniformly spaced dithers for the primary instrument generally yield non-uniform dithers for the parallel instrument. A specific exception are the Planetary Camera (PC) and Wide Field Cameras (WFCs) of WFPC2, where a pattern that produces a sub-pixel dither on both instruments has been developed (Section 6.4). However, in most cases, we recommend that users select their dither pattern in order to obtain the best possible data from their primary instrument.

2.5 More Detailed Considerations

2.5.1 Data Quality Issues involved in Dithering

The primary considerations in designing a dithered observational program are cosmic rays, hot pixels, spatial sampling and signal-to-noise. Finding the optimal strategy to deal with these issues is not always straightforward. Careful consideration must be given to the impact of breaking a long observation into multiple exposures, particularly in terms of increasing the overall read-out noise and reducing the amount of science exposure time due to observational overheads. The optimal strategy chosen will ultimately depend upon carefully weighing all these issues against one another, and also against the scientific questions involved such as: is the underlying structure totally unknown, is spatial resolution of paramount importance or is photometric accuracy the most crucial aspect?

- **Cosmic rays**: The best way to deal with cosmic rays is to obtain a minimum of two exposures, preferably 3 or more, thereby reducing the number of common cosmic ray hits according to the binomial theorem. Even two exposures can have a substantial number of pixels with overlapping cosmic ray hits. For example, 2x1500s WFPC2 exposures will have ~1500 pixels on each chip that are affected by cosmic rays on both images, but 3x1000s exposures have only ~20 pixels on each chip that would be hit by cosmic rays in all three exposures. Exact cosmic ray losses for any given observing scenario can be determined by running the appropriate Exposure Time Calculator which is available on the STScI Web site for each instrument.
- **Hot Pixels**: There are three ways to deal with hot pixels: (a) recalibrate using "dark frames" that bracket the date of the observation; (b) obtain a second image (or pair of images which will best reject cosmic rays) shifted by a small amount spatially (e.g. about 5 pixels); (c) use an IRAF/STSDAS task such as "warmpix" to filter out the obvious hot pixels. Since some hot pixels are variable on very short time scales, the most robust strategy is to obtain multiple images.
- **Undersampling**: to improve sampling of the PSF, together with increased spatial resolution, the images need to be shifted by sub-pixel amounts. Generally, subsampling by 1/2-pixel offsets provides the most dramatic improvement over non-dithered images. In some cases, observers wish to further explore the limits of the instrument and spacecraft pointing accuracy by considering sub-pixel shifts of 1/3 of a pixel or less. The extent to which such refinements can be explored depends primarily upon the number of orbits available and the instrument being used.

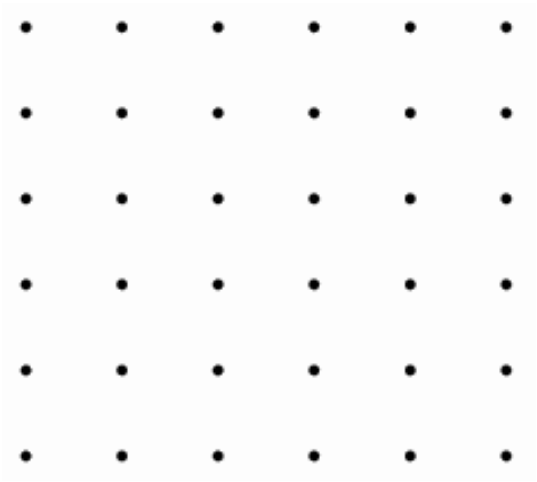
- **Photometric Accuracy**: HST instruments can have variations in the sensitivity across each individual pixel, this is referred to as intra-pixel sensitivity. If the PSF is undersampled, this can complicate the photometric analysis of dithered images. Thus, programs requiring maximal accuracy in photometry may not always benefit from dithering.

2.5.2 How Many Dither Positions - 2, 3, 4 or more?

If integer-pixel dithers are all that is required, specifically to ameliorate the effect of hot pixels, then 2 or 3 different locations should be sufficient to guarantee that sources falling on hot pixels are not completely unrecoverable. The remainder of this discussion focuses on sub-pixel dithering, including the strategies and issues involved. The best choice for the number of sub-pixel dithers depends on the amount of time available and the goals of the project. Dithering requires a noticeable amount of spacecraft overhead, with each dither offset typically adding ~2-3 minutes of overhead to the total observing plan.

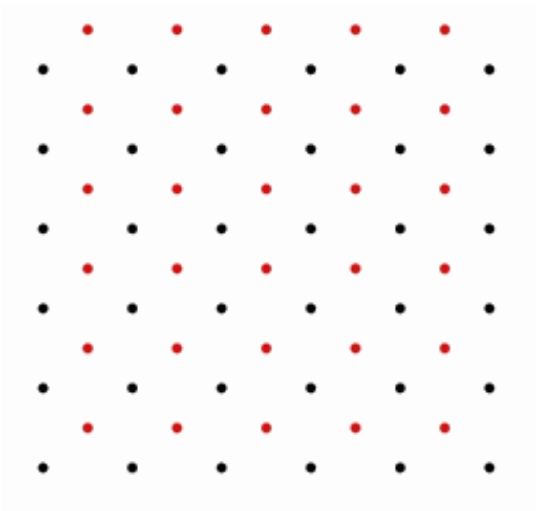
- The very simplest type of sub-pixel dither is a two-point dither offset along only one axis, i.e. one image obtained at the original pixel position of $(0, 0)$ and a second obtained at $(0, n + \frac{1}{2})$ pixels where n is an arbitrary integer. This scenario is only really relevant to STIS long-slit spectroscopy, if it is desirable to improve the subsampling along the (spatial) slit direction.
- A two-point sub-pixel dither in imaging data takes one image at the original pixel position of $(0, 0)$, and a second image offset by half a pixel in both the x and y, i.e. at $(n + \frac{1}{2}, m + \frac{1}{2})$, where n and m are arbitrary integers. This produces a substantial increase in information over non-dithered data. In the case of a square pixel, this dither pattern forms the sampling that would be produced by an array with pixels $\sqrt{2}$ smaller than the original array, rotated by a 45 degree angle from the original orientation. Setting n and m to be a few pixels (e.g. around 5-10) will also allow hot pixels to be moved by sufficient amounts to reduce their impact on objects of interest. Figure 2.1 shows the sampling of 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 additional dither is in red.

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



The sampling of the WFC3 IR detector on the sky.

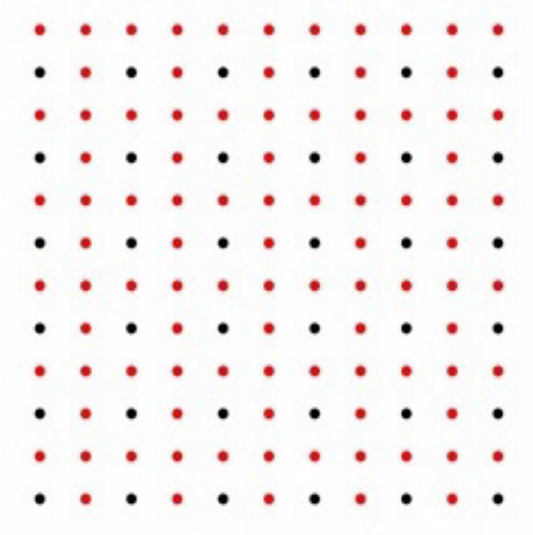
Figure 2.2: Sampling for a 2-point Dither



The sampling produced by introducing a two-point dither using the WFC3 IR detector.

- The four-point dither is a very natural option given rectangular pixels. This dither yields a total of 4 images that are offset from one another by half pixels in x and y i.e. $(0,0)$, $(0,\frac{1}{2})$, $(\frac{1}{2},\frac{1}{2})$, $(\frac{1}{2},0)$. Again integer shifts can and should be used to reduce the effect of hot pixels. This yields uniform tiling along both axes of the (*X,Y*) plane using half-pixel offsets, thereby providing more robust half-pixel subsampling of the PSF than a simple two-point dither which is only along one direction (see Figure 2.3). In fact, given the native sampling of HST instruments, an accurate four-point dither recovers nearly all of the information available in an image.

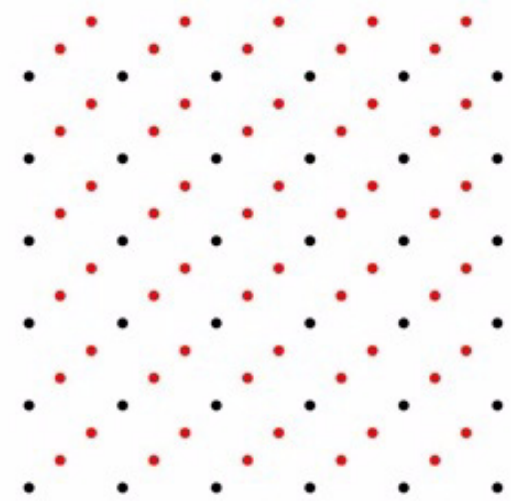
Figure 2.3: Sampling for a 4-point Dither



A four-point dither.

- There may be cases, particularly for programs with only a few orbits, where the available time breaks down more naturally into blocks of 3 exposures instead of 2 or 4. However, the best placement of a three point dither is not obvious. This is because there is no natural way to tile the plane using three placements of a rectangular CCD grid. Therefore, if a user can afford a four-point dither they may prefer to do so. However, both the natural two and four-point dither placements minimize the maximum distance from any point on the image plane to the nearest dither location. One can ask what three point dither pattern also minimizes this maximum distance. Through a (computer) calculation one can show that this is done by offsets along the diagonal at pixel offsets of $(0,0)$, $(\frac{1}{3},\frac{1}{3})$ and $(\frac{2}{3},\frac{2}{3})$ (the symmetric diagonal works just as well, of course). Again, additional integer offsets of a few pixels can (and should) be added to help remove detector defects. Figure 2.4 shows a three-point dither applied to the WFC3 IR detector.

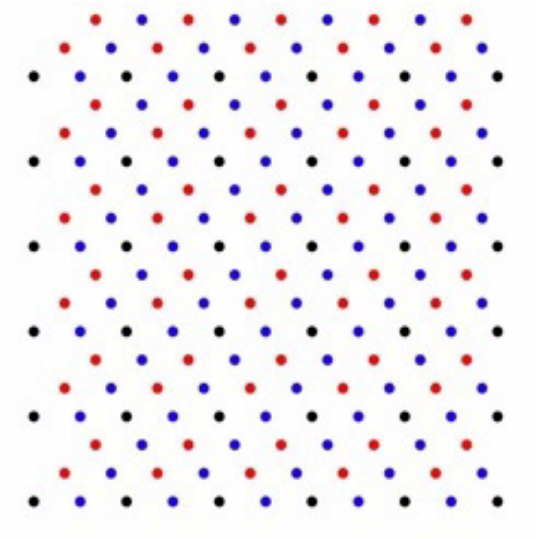
Figure 2.4: Sampling for a 3-point Dither



A three-point dither applied to the WFC3 IR detector.

- Users with multiple-orbit visits attempting to obtain extremely accurate PSFs may consider even finer subsampling of the pixel. An eight-point dither can be performed by crossing a four point dither with a two-point dither. The two point dither should be of the form $(m + \frac{1}{4}, n + \frac{1}{4})$. This places a point in the center of each of the "squares" created by the four-point dither. Users should be advised 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. Again, the four-point dither, if performed accurately with the loss of few pixels to cosmic rays or other defects, recovers nearly all the information in an HST image. Therefore, users of instruments like ACS may prefer to cross a small four point dither with a larger two or three point dither that will cover the gap between the chips. The four point dithers will insure good subsampling in the final combined image.
- A number of WFC3 users have inquired about dithers in multiples of three, as many users find three exposures fit well into a single orbit. One can easily create a nine-point dither by dividing the original pixel with a 3×3 grid. What is less clear is how to form a six-point dither. Again, a calculation suggests that crossing the linear three point dither described above with a $(0, \frac{1}{2})$ two-point dither is the optimal strategy. For square pixels, the half-pixel dither could be taken in either direction, but the WFC3 IR pixels are slightly longer in the x-direction, and so we take the dither 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 6-point Dither



A six-point dither.

2.5.3 Data with Inaccurate Offsets in Position or Roll Angle

After the observations have executed, the pointing and orientation of the telescope can be determined directly by using cross-correlation techniques, as well as through examination of the jitter files that can be requested from the HST archive as part of the data products. The HST Data Handbook contains further details on the jitter file and other data products, and how to extract their information. The most recent version can be found at the following Web page:

http://www.stsci.edu/hst/HST_overview/documents/datahandbook/

For most programs, it is sufficient to determine the translational shift between images - the chance of a spurious roll angle offset is relatively small, and many long programs have now been performed without experiencing the roll offsets originally seen in the HDF. However, the drizzle software is capable of combining the shifts in rotation as well as position.

2.5.4 How many Images to Obtain at each Dither Location

It is generally possible to successfully remove cosmic rays using only a single image at each dither location, i.e, “singly-dithered” images, using the drizzle software that is incorporated with the dither package in STSDAS. If sub-pixel dithering is desired for small programs (less than about one orbit per target), or programs where reduction of read-noise is critical (e.g., narrow-band imaging of extremely faint sources), then the best approach is likely to involve obtaining only one image at each dither location. For larger programs, however, or when read-noise is not a serious issue, the user can choose between the slightly improved sampling of a larger number of independent dither pointings, or the relative simplicity and lower overhead of multiple exposures at a given dither pointing.

2.5.5 Specific Instrument Related Issues

2.5.5.1 WFPC2

In addition to increasing information on the smallest spatial scales, dithering can be used to reduce the effect of flat-field errors in very deep images. Large dithers (tens of pixels) were used in the HDF for this purpose. Furthermore, dithers greater than one or two pixels can be used effectively to eliminate chip defects such as hot-pixels and bad columns.

The Effect of WFPC2 Geometric Distortion on Dither Offsets

The pixels near the edge of the CCD differ in size on the sky from those near the center. Thus a shift of (10,10) pixels at (400, 400) corresponds to a shift of about (10.2, 10.2) pixels at (700, 700). The default dither-line spacing produces a shift of (2.5, 2.5) WF pixels and (5.5, 5.5) PC pixels. Therefore, over nearly the entire field of view the difference in offset - even on the PC - is less than 0.1 pixels, and the shift will be essentially optimal across the whole field. However, the standard dither-box spacing offsets the telescope by as much as 0.75 arcseconds or 15.5 PC pixels. This means that at (700, 700) the shift differs from that at the center by ~0.3 pixels in x and y. While the drizzle software removes this geometric offset, it cannot change the fact that the sampling will not be optimal across the entire field of view.

The dither-box defaults were chosen to avoid repeating the placement of objects on the same columns (to reduce the effects of bad columns). However, if one is willing to live with the possibility that a given position of interest may fall twice on one of the several bad columns per chip, then one can use smaller offsets to produce a box that is more nearly perfect across the entire chip, for instance a square 2×2 box with side of 2.5 WF pixels (equivalently 5.5 PC pixels).

The Exact Relationship Between POS TARGs and WFPC2 CCD Rows and Columns

For WFPC2 an additional complication is introduced by the fact that the four chips are not precisely aligned with one another, but possess small rotational offsets (<0.5 degree) from their nominal alignments. Thus, the POSTARG axes run exactly along the CCD rows and columns on whichever aperture is specified for the observations. For example, if aperture WF3 is specified, the POSTARG axes will run *exactly* along the rows and columns on WF3, and will run only approximately along the rows and columns of the other CCDs. Note that if WFALL is specified, then the rotation for WF3 is used since WF3 is the reference chip for the WFALL aperture.

The CCD rotation misalignments lead to errors when attempting to dither by certain pixel amounts. For small dithers (<0.3 arcseconds) the rotational offsets between the CCDs are unimportant, as they imply pixel registration errors less than 3 milliarcseconds, which is roughly the nominal pointing and guiding stability for HST. But such small dithers do not allow integral pixel stepping simultaneously on the PC as well as the WF chips. A dither of 0.5 arcseconds (5 WFC pixels or 11 PC pixels) gives near-integral stepping on the PC and the WF chips, though the CCD rotations will then introduce registration errors up to 5 mas. An offset of (1.993, 0.000) arcseconds in x on WF3 would cause spurious motion in y of 0.17 pixel on WF4, due to the rotation.

Two basic types of dither patterns are defined for WFPC2, and are implemented in the APT software that is used to process Phase II observing programs. These patterns can also be used with non-default spacings when necessitated by very specific types of observations, although in general we recommend that observers use the default spacings which are optimized for a wide variety of scientific programs.

- WFPC2-LINE: a two-point dither with a single default offset of (0.25arcseconds, 0.25arcseconds), which produces an offset with half-pixel increments along both the x and y axes on all the chips: the offset corresponds to (2.5, 2.5) pixels on the WF chips and (5.5, 5.5) pixels on the PC. This pattern produces half-pixel subsampling of the PSF on all the chips, while at the same time including integer-pixel offsets to ameliorate the effects of hot pixels and other chip artifacts.
- WFPC2-BOX: a 4-point parallelogram dither, with default positions in arcseconds of (0.0, 0.0), (0.5, 0.25), (0.75, 0.75), (0.25, 0.5). This combination of integer-pixel and half-pixel shifts produces complete half-pixel subsampling of the PSF by all 4 quadrants of each pixel. Therefore this strategy is an improvement over the simple 2-point dither which only provides subsampling in two quadrants of each pixel. The disadvantage of the 4-point dither is that it involves more overhead, which has to be weighed against the relative improvement in subsampling.

2.5.5.2 ACS

For ACS, an important issue to consider in designing a dither strategy is its relatively large distortion: up to 8% across the WFC camera. Moreover, the projections of the detector pixels on the sky correspond to parallelograms with interior angles that differ from 90^{o} by up to 5 degrees, depending upon the location of the pixel on the detector. The differential distortion across the chip means that shifts of more than a few pixels produce noticeably different sub-pixel offsets across the entire chip. However, the two chips that compose the WFC are separated by a gap of order 2.5 arcseconds (~50 WFC pixels). As a result, many ACS dither strategies involve the use of offsets sufficiently large to allow the detectors to cover this gap. These will have differing sub-pixel effects across the detector. When taking several exposures of a field in a single filter, observers are generally encouraged to use dithers instead of CR-SPLIT exposures for a number of reasons: to change the placement of hot pixels on the field, to resample the point spread function and to reduce the impact of errors in the pixel-to-pixel flats.

Since dither offsets are achieved by specifying POSTARG shifts along the x and y axes of the detectors, this means that each POSTARG shift on the sky follows the edges of a parallelogram. The shifts have been defined so that displacements in rectilinear sky coordinates are aligned along the y-axes of the detectors (Mutchler and Cox 2001). Thus, for example, a displacement of one WFC pixel along the x- and y-axes of the detector is broken down as follows: the displacement of 1 pixel along the detector y-axis corresponds to 0.0497 arcseconds along the Y-POS direction; however, the displacement of 1 pixel along the detector x-axis corresponds to 0.0496 arcseconds

along the X-POS direction, plus an additional 0.0038 arcseconds along the Y-POS direction, due to the non-orthogonality of the pixels on the sky.

For the WFC, HRC, and SBC, a number of pre-defined offset patterns have been created (Mutchler and Cox 2001). These are available in the APT Phase II software, and are aimed at covering a wide range of observing requirements:

- DITHER-LINE pattern has 2-point integer pixel spacing by default to ameliorate the effect of chip artifacts. For the WFC, HRC, and SBC, the offsets are (5, 60), (5, 5), (10, 10) pixels respectively. The large y-shift for the WFC is to enable the inter-chip gap to be covered. This pattern can also be modified to subsample the PSF using half-pixel spacing, for example (2.5, 1.5) pixels, or even 1/3-pixel spacing, e.g. dither positions of (0, 0), (2.3, 1.3), (4.6, 2.6) pixels.
- MOSAIC-LINE pattern with large offsets, comparable to the size of the detectors, to increase the field of view.
- DITHER-BOX pattern, by default a set of 4 offsets consisting of integer-pixel and half-pixel offsets (0, 0), (5.0, 1.5), (2.5, 4.5), (-2.5, 3.0), aimed at providing more complete 1/2-pixel subsampling of the PSF.
- MOSAIC-BOX pattern, a 4-point pattern with large offsets, comparable to the size of the SBC and HRC detectors. It should be noted that HST ground-system limitations currently prevent this pattern from being implemented for the WFC.

The above pre-defined patterns should prove sufficient for the vast majority of scientific programs, however, other patterns can also be created simply by using a combination of POSTARG offsets.

2.5.5.3 NICMOS

A wide variety of pre-defined patterns has been created for NICMOS, to allow an easy implementation of both integer-pixel and sub-pixel dithering. These are generalized extensions to the simple line and box dithers by including spiral and chopping dithers, which are necessary to allow successful removal of a number of NICMOS artifacts. The advantages offered by dithering with NICMOS are the following:

- *Post-SAA Cosmic Ray Persistence:* The NICMOS detectors suffer from persistent after-images when exposed to a strong signal. This can arise from astronomical objects, but it also occurs due to cosmic ray bombardment during every passage of HST through the South Atlantic Anomaly (SAA). After SAA passages, a very large fraction of NICMOS pixels glow with a persistent signal that can take up to a few orbit to decay completely. Dithering can help average over the additional noise (really non-Gaussian, spatially correlated signal) that results from SAA-induced persistence. The worst effects of CR persistence can sometimes be removed by the drizzle and blot techniques. The NICMOS team has also implemented Post-SAA cosmic ray persistence removal software and dark observations which ameliorate a substantial

amount of the noise induced by traversing the contours. More information on the details of CR persistence removal can be found in the NICMOS Data Handbook (McLaughlin & Wiklind 2007).

- *Photometric accuracy:* the effects of large-scale flat-field variations and of bad-pixels can be controlled via integer-pixel dithering. In addition, for relatively bright objects, dithering can eliminate potential problems of image persistence. Geometric distortion in NICMOS is relatively small, except for the NIC3 camera in its out-of-focus position. We recommend dither steps of ~10 pixels for compact sources. The SPIRAL-DITH pattern can be used to generate dither patterns with 2 positions or more.
- *Improved sampling:* NIC3, NIC2 (shortward of 1.75 microns) and NIC1 (shortward of 1.0 microns) undersample the image. As in the case of WFPC2, the quality of the image can be improved by sub-pixel dithering. Most of the information can be recovered via a two-point dither, and virtually all the information can be recovered with four-point dithers. Since NICMOS geometric distortion is relatively small (except for NIC3 when out-of-focus), large dither steps of order ~10 pixels can be used. Telescope pointing errors, which can be of the order of 0”.02, may prevent one from obtaining an optimal dither pattern in NIC1 and NIC2, since the uncertainty corresponds to 0.43 NIC1 pixels and 0.27 NIC2 pixels; in this case more than four dither positions are advisable. For NIC3, the telescope pointing uncertainty corresponds to 0.1 pixels shift only, and four dither positions should still be viable for recovering the information. The pre-defined SPIRAL-DITH pattern can be effectively used for this purpose.
- *Background removal in uncrowded fields of compact objects:* Observations with the NICMOS long wavelength filters (central wavelength longward of 1.7 microns) are affected by variable thermal emission from the telescope (NICMOS ISR 2003-007). To remove this contribution from the images, suitable background observations must be obtained. For compact targets and uncrowded fields, observations of the background can be obtained by dithering the targets across the detector’s FOV. The use of the SPIRAL-DITH pattern with two or four positions, and a dither step of 10 pixels or more (depending on the size of the targets), may be appropriate for many cases, although the parameters may change according to the nature of the observations. The advantage of dithering in such a case (rather than chopping, for example) is that the target will remain on the chip for all observations, increasing the efficiency of the observation.

Dithering NICMOS observations may also have disadvantages that an observer should consider:

- Cosmic ray removal is not straightforward in pairs of sub-pixel dithered images. If you plan to use sub-pixel dithering to improve the image sampling, then MULTIACCUM mode or two ACCUM mode exposures per position should be obtained to help cosmic ray removal BEFORE image reconstruc-

tion. Some cosmic ray detection and removal is also performed during the calibration of Multiaccum datasets as multiple reads during the exposure allows for statistical elimination of abnormal flux values. In general, the use of ACCUM mode is discouraged because there is little on-orbit calibration done for this mode (e.g., dark frames, etc.).

- NICMOS Attached parallels: the three NICMOS cameras, NIC1, NIC2, and NIC3, have different plate scales; care should be taken in ensuring that if integer-pixel steps are desired in attached parallel (NIC1+NIC2) observations, the steps are carefully chosen to satisfy this requirement.
- Overheads: The implementation of patterns requires at least 10 - 12 seconds overhead per dither step. Large numbers of dithers can easily add up to minutes taken out of a visibility period for an entire pattern. The trade-off between the advantages offered by dithering, and the diminished amount of observing time should be considered in deciding whether or not to dither.
- Rapid dithering can impose an additional load on the full system in terms of command volume needed to execute the observations, overheads for science data buffer management, and in the volume of data that must be processed through the pipeline. In extreme cases, such as when the overheads required to execute the observations far surpass the actual exposure times, these extra loads can result in lowered overall efficiency of HST observations.

In general, the benefits of dithering greatly outweigh the disadvantages for NICMOS observations. Whenever possible without incurring excessive overhead, we recommend dithering as much as possible when taking NICMOS data. Note, however, that many NICMOS observations are significantly affected by read-out noise, especially for Cameras 1 and 2 and observations shorter than 1.8 micron. Therefore, the effects of read-out noise on multiply-dithered short exposures should always be carefully balanced against the benefits provided by extensive dithering.

2.5.5.4 STIS

The concept of dithering as applied to STIS observations is multifaceted, since STIS can be used to obtain either images or spectra, and the best method for dithering depends upon the science goals for the observing program. The goal may either be to increase the spatial resolution or to ameliorate the effect of hot pixels or uncertainties in pixel-to-pixel sensitivity with respect to the reference flat-fields.

Imaging-Mode Dithering

Observers can reduce the effect of flat-field uncertainties (particularly for the MAMA detectors) by using a small step pattern with integral pixel shifts. This stepping, or dithering, effectively smooths the detector response over the number of steps, achieving a reduction of pixel-to-pixel non-uniformity by the square root of the number of steps, assuming the pixel-to-pixel deviations are uncorrelated on the scale of the steps. This approach requires sufficient signal-to-noise to allow image registration.

Alternatively, one may improve the spatial resolution somewhat with a dither pattern that includes sub-pixel shifts. Images obtained with the STIS/CCD (0.05

arcsec/pixel), have nearly the same spatial scale as those obtained with the WFPC2/PC camera (0.045 arcseconds/pixel), so that the improvement in spatial resolution would be similar. The spatial scale of MAMA images is half that of the CCD, so the gain in spatial resolution from dithering MAMA images will be more modest, and probably insignificant in the majority of programs. Although the PSF on the MAMA detectors should be narrower than on the CCD because of the shorter wavelengths at which the MAMAs operate, in practice this advantage is offset by additional complications introduced through the instrument optics. It is important to realize that the focus varies across the field of view for STIS imaging modes, with the optical performance degrading by ~30% at the edges of the field of view. Thus, the achievable spatial resolution is significantly compromised in those regions.

Whether or not the dither pattern includes sub-pixel shifts, the effects on CCDs of bad columns, hot pixels, etc., can be reduced or eliminated if the dither pattern is greater than a few pixels. Predefined dither patterns that are available for observers to use, these include:

- STIS-CCD-BOX: This will produce a four-point parallelogram scan designed for dithering across the CCD pixels.
- STIS-SPIRAL-DITH: This produces a spiral dither pattern, starting at the center and moving outward counterclockwise. Note that a STIS-SPIRAL-DITH with four points yields a square pattern, but the optimum pattern for detector dithering to enhance resolution is either STIS-CCD-BOX or STIS-MAMA-BOX.

Spectroscopic-Mode Dithering

Dither patterns can be used with STIS spectroscopic modes for the following purposes:

- to average over pixel-to-pixel flat-field uncertainties;
- to facilitate removal of hot and cold pixels (e.g., by using integer pixel steps);
- to subsample the spatial PSF along the slit (by sub-pixel steps along the slit);
- Stepping along the dispersion direction, perpendicular to the spatial axis of the slit:
- to subsample the spectral Line Spread Function (LSF) by stepping a fraction of a pixel along the dispersion direction;
- to map out a two-dimensional region of the sky by using larger step sizes equal to or greater than the slit width.

In first-order spectroscopic modes, improved S/N ratios can be achieved by stepping the target along the slit, taking separate exposures at each location. These separate exposures will subsequently be shifted and added in post-observation data processing. This dithering smooths the detector response over the number of steps, in a manner analogous to that for imaging. For echelle modes, stepping is only possible using the long echelle slit (6x0.2 arcseconds). Note that in the high dispersion echelle modes the Doppler shifting due to spacecraft motion will cause the counts from any

output pixel to have been sampled at many independent detector pixels in the dispersion direction (for exposures comparable to an orbit visibility period and targets well away from the orbital pole of HST).

In slit-less or wide-slit mode, stepping along the dispersion would allow independent solutions for spectrum and flat-field, bearing in mind however the increased complexity due to the convolution of the spectrum with the spatial structure in the source. This technique is likely to be useful only if the constituent spectra have a good S/N ratio (perhaps 10 or better), so that the shifts between spectra can be accurately determined.

A variation on this technique involves using one of the contingent of fixed-pattern, or FP-SPLIT slits. These slits are designed to allow the wavelength projection of the spectrum on the detector to be shifted such that the fixed-pattern noise in the flat-field and the spectral flux distribution of the target can be computed simultaneously using techniques that have been successfully applied to data taken with GHRS. Note that this approach is likely to work best if the spectra have a good S/N ratio. More detailed information on the use of FP-SPLIT slits is provided in the STIS Instrument Handbook (Leitherer et al. 2001).

In many configurations the spectral line FWHM is less than two detector pixels. Possible solutions include stepping the target along the dispersion direction in a wide slit or slit-less aperture to subsample the LSF by displacing the spectrum. This technique can also be used to increase the S/N ratio. To employ this strategy, the observer will have to trade off the benefits of improved sampling with the negative impact of increased wings in the LSF when using a wide slit, particularly for MAMA observations. The use of high resolution (default) for MAMA observations may provide 15-30% better sampling, but flat-field variability may make it difficult to realize the benefits, particularly if high S/N ratio spectra are needed.

There are several pre-defined dither patterns that are available for observers to use, these include:

- STIS-PERP-TO-SLIT: This is normally used with a spectroscopic slit. It produces a scan along the POSTARGX-axis of the aperture; this is used to map a two-dimensional region of the sky (see Chapter 11 of the STIS Instrument Handbook). The target is moved perpendicular to the slit along the AXIS1 (dispersion)
- STIS-ALONG-SLIT: This is also normally used with a spectroscopic slit. It produces a scan along the POSTARGY-axis of the aperture; this is used to step a target along the long slit to dither bad pixels or improve spatial resolution (see the *STIS Instrument Handbook*). The target is moved along the slit in the AXIS2 (cross-dispersion or spatial) direction.

2.5.5.5 WFC3

WFC3/UVIS

WFC3/UVIS images are in many ways similar to ACS/WFC images. The detector comprises two rectangular CCD chips separated by a gap approximately 35 pixels wide, so that a gap-stepping dither is needed to avoid having a gap across the center of the field of view. The projection of the pixels on the sky is in the shape of a rhombus,

with an angle between the X and Y axes of 86 degrees. As with the ACS/WFC, a POSTARG in Y is along the Y axis of the aperture (along columns), and a POSTARG in X is perpendicular to the Y axis (not quite along rows). The plate scale is 0.04 arcseconds/pixel on each axis, and the FWHM of the point spread function is between 1.6 and 2.3 pixels, depending on wavelength. WFC3/UVIS images will thus benefit from half-pixel dithering, but not as much as WFC3/IR images. Non-linear distortion causes the projected area of the pixels to vary by +/-3% relative to that at the center of the detector, so POSTARGs and patterns will produce shifts in pixels that vary with location on the detector.

Five patterns have been installed in the phase 2 software to dither and mosaic WFC3/UVIS images:

- WFC3-UVIS-DITHER-LINE dithers the UVIS aperture by (2.5, 2.5) pixels to sample the point spread function with fractional pixel steps.
- WFC3-UVIS-DITHER-BOX samples the point spread function with fractional pixel steps and produces spacings of more than one column to move hot columns. The relative steps in pixels are (0, 0), (4.0, 1.5), (2.5, 4.0), and (-1.5, 2.5).
- WFC3-UVIS-MOS-DITH-LINE has a primary pattern that dithers over the gap between the two chips of the detector with relative steps of (-4.5,-60.25), (0, 0), and (4.5, 60.25) pixels. A secondary pattern adds a dither of (2.5, 1.5) pixels to the primary pattern.
- WFC3-UVIS-MOS-BOX-LRG produces a UVIS mosaic that can be executed with a single set of guide stars. It dithers the gap between the chips so that no region lies in the gap more than once. The relative steps in pixels are approximately (-1000, -997), (1000, -1001), (1000, 997), and (-1000,1001).
- WFC3-UVIS-MOSAIC-LINE is designed for observations using the full WFC3/UVIS detector for primary exposures and the full ACS/WFC detector for parallel exposures. It dithers over the inter-chip gap on both detectors. The relative steps on the WFC3/UVIS detector are (0, 0) and (36.5, 71.5) pixels.

Other patterns can be created by using POSTARG offsets or generic patterns, or by changing the spacings in the defined patterns. For programs requiring high precision small aperture photometry, observers should see WFC3 ISR 2008-10 for a discussion of features called “droplets”, caused by contamination on the outer window of the UVIS detector. Dithers ~100 pixels are recommended to improve the photometry.

The WFC3 pipeline produces cosmic ray rejected (CRJ) images from input FLT images for CR-SPLIT exposures. When MultiDrizzle is run in the pipeline, it will use the FLT images as input, tagging cosmic rays in those images with a different value of DQI (4096) from the value used by CALWF3 (8192). Observers are generally encouraged to use dithers instead of CR-SPLIT exposures for a number of reasons: to change the placement of hot pixels on the field, to resample the point spread function and to reduce the impact of errors in the pixel-to-pixel flats.

WFC3/IR

The WFC3/IR pixels are projected as rectangles on the sky, with X and Y plate scales ~0.14 and 0.12 arcseconds per pixel. A POSTARG in Y is along the Y axis of the aperture (along columns), and a POSTARG in X is along the X axis (along rows). The FWHM of the point spread function is between 1.0 to 1.25 pixels, so sub-pixel dithering is needed to recover spatial resolution. Non-linear distortion causes the projected area of the pixels to vary by +/-4% relative to that at the center of the detector, so POSTARGs and patterns will produce shifts in pixels that vary with location on the detector.

Three patterns have been installed in the phase 2 software to dither and mosaic WFC3/IR images:

- WFC3-IR-DITHER-LINE takes steps large enough for photometric accuracy and samples the point spread function with fractional pixel steps. The relative steps in pixels are (0, 0) and (3.5, 3.5).
- WFC3-IR-DITHER-BOX-MIN takes steps just large enough for photometric accuracy and samples the point spread function with fractional pixel steps. The relative steps in pixels are (0, 0), (4.0, 1.5), (2.5, 4.0), and (-1.5, 2.5).
- WFC3-IR-DITHER-BOX-UVIS is a four-point box pattern that produces an IR mosaic covering the same area as the UVIS detector. The IR imaging is intended to be accompanied by a UVIS exposure (or small dither pattern) using the aperture UVIS-CENTER.

Other patterns can be created by using POSTARG offsets or generic patterns, or by changing the spacings in the defined patterns. Note that there is a ~45 pixel diameter dead spot near the lower edge of the WFC3/IR detector, centered at ~[358,54]. A dither larger than this diameter should be used if imaging in that area is required.

WFC3/IR exposures are made with predefined timing sequences of non-destructive reads. As in NICMOS, up-the-ramp fitting of the fluxes in the sequence is used to identify and remove cosmic ray flux from each pixel. The accuracy of the procedure depends on the timing sequence and the number of frames specified in the proposal, just as the accuracy of traditional cosmic ray rejection in CR-SPLIT exposures on a CCD detector depends on the number of exposures and the exposure time. The cosmic ray rejected FLT image is used as input to MultiDrizzle. As for WFC3/UVIS images, DQI values of 4096 and 8192 are used to tag pixels with cosmic rays identified by MultiDrizzle and by CALWF3, respectively.

CHAPTER 3:

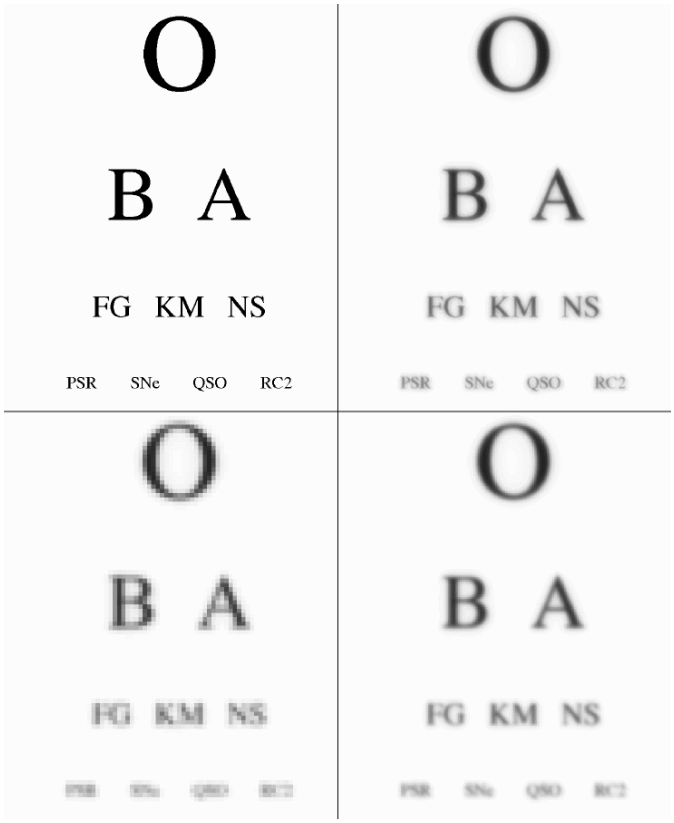
How the Drizzle Algorithm Works

In This Chapter...

3.1 Theory

While much high spatial frequency information in the image is permanently lost by smearing with response of the detector pixels, the quality of the image can be greatly improved by combining sub-pixel dithered images. Each of the pixels from the different exposures can be thought of as sampling a final, higher-resolution image, which is the "true image" of the sky convolved with the optical PSF and the pixel-response function of the CCD. The effect of undersampling is illustrated by a set of four different examples (Figure 3.1). In the upper left hand corner one sees the "true" image, as it would be seen by a telescope of infinite aperture. In the upper right, the image has been convolved with the PSF of the HST/WFPC2 and in the lower left it has been subsequently sampled by the WF2 CCD. The loss of spatial information is immediately obvious.

Figure 3.1: The Effects of Image Convolution and Subsampling

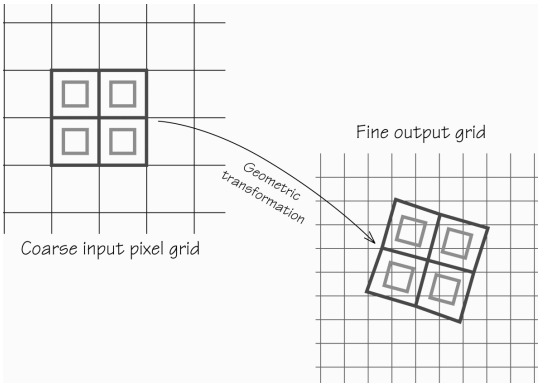


Representation of the effects of image convolution and subsampling; In the upper left hand corner one sees the “true” image, as it would be seen by a telescope of infinite aperture. In the upper right, the image has been convolved with the PSF of the HST/WFPC2 and in the lower left it has been subsequently sampled by the WF2 CCD. The loss of spatial information is immediately obvious. On the lower right the image has been reconstructed using the Drizzle algorithm.

Much of the information lost to undersampling can be recovered. This is shown in Figure 3.1, which displays the image recovered using one of the family of techniques referred to as “linear reconstruction”. However, the simple implementations of these techniques generally introduces 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. 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: How Drizzle Maps Input Pixels to Output Pixels



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 (Section 5.5.6), which is simply 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 (or psize in PyDrizzle) (Section 5.5.6), allows the user to specify the size of the output pixels. In the case of the Hubble Deep Field North (HDF-N), the drops had linear dimensions one-half that of the input pixel (i.e., pixfrac=0.5) - slightly larger than the dimensions of the output subsampled pixels. The flux value of each input pixel is divided up into the output pixels with weights proportional to the area of overlap between the "drop" and each output pixel. If the drop size is too small, not all output pixels have data added to them from each of the input images. One should therefore choose a drop size that is small enough to avoid convolving the image with too large an input pixel "footprint", yet sufficiently large to ensure that there is not too much variation in the number of input pixels contributing to each output pixel.

3.2 Image Reconstruction and Restoration Techniques

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

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

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

An implementation of the Richardson-Lucy method is in the IRAF/STSDAS task "acoadd". However, this technique is unable to handle large dithers, and is limited by typical computing capabilities to combining either small regions of many images, or the entire image of only a few dithers. Furthermore, the present task can accommodate neither geometric distortions nor the changing shape of the PSF across the field of view. This technique, like all non-linear techniques, produces final images whose noise properties are difficult to quantify. In particular, this method has a strong tendency to "clump" noise into the shape of the input PSF.

In the rest of this section we will focus on the family of linear reconstruction techniques, of which two opposite extremes are "interlacing" and "shift-and-add"; with the "drizzle" algorithm representing a continuum between these two extremes.

3.2.1 Interlacing

If the dithers are particularly well-placed, one can simply interlace the pixels from the images onto a finer grid. In the interlacing method, the 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 3x3 array of dithered images. However, due to occasional small positioning errors of the telescope, and non-uniform shifts in pixel space across the detector caused by the geometric distortion of the optics, true interlacing of images is generally not feasible.

3.2.2 Shift-and-Add

Another standard simple linear technique for combining shifted images, descriptively named "shift-and-add", has been used for many years to combine dithered infrared data onto finer grids. Each input pixel is block replicated onto a finer sub-sampled 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.

3.2.3 Drizzle

In response to the limitations of the two techniques just described, 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 the 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 allows an improvement in resolution, and a reduction in correlated noise, compared with images produced using pure 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 sub-sampled 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 (Section 5.5.6), which can be set to values ranging from 0 to 1, and 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 (Section 5.5.6) 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.

3.3 Weight Maps and Correlated Noise

When images are combined with drizzle a weight map can be specified for each input image. This image minimally contains information on the bad pixels in the image. When the final output science image is generated, an output weight map which combines the information from all the input weighting images is also saved. When a drop of value i_{xy} and user defined weight w_{xy} is added to an output image I_{xy}, with

weight W_{xy} and a fractional pixel overlap of $0 < a_{xy} < 1$, the resulting value of the image I'_{xy} and W'_{xy} is:

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

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

Drizzle has a number of advantages over standard linear reconstruction methods. Since the area of the pixels scales with the Jacobian of the geometric distortion, it is preserved for surface and absolute photometry. Therefor the flux can be measured with an aperture that is independent of its position on the image. Since the method 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 sub-pixel scales. A more practical approach is to use the weight maps generated by drizzle to calculate the expected r.m.s. noise. The weight appropriate to a given value of the scale parameter (expressed in terms of the ratio of the output to input pixel size), can be calculated in the following way (as described by Casertano et al. 2000):

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

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

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), and σ is the readnoise in DN/pixel. The quantity f represents the inverse flat field, corresponding to the way in which the HST pipeline flats are defined.

A more in-depth discussion of noise in drizzled images can be found in Section 3.3.1 Correlated Noise Details.

3.3.1 Correlated Noise Details

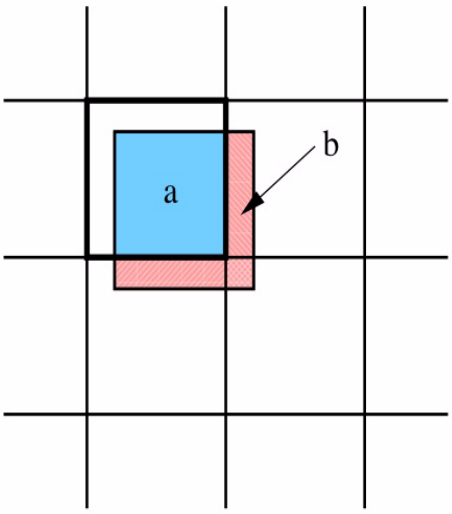
3.3.1.1 Overview

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 the statistical errors when drizzled images are analyzed using object detection and measurement programs such as SExtractor (Bertin and Arnouts 1996) and DAOPHOT (Stetson 1987).

The correlation of adjacent pixels implies that a measurement of the 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.

Figure 3.3: Schematic View of the Noise Distribution for a Pixel



A schematic view of the distribution of noise from a single input pixel between neighboring output pixels.

In Figure 3.3 we show an input pixel (broken up into two regions, a and b) 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:

$(a^2 + b_1^2 + b_2^2 + b^3)\epsilon^2 < \epsilon^2$

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

3.3.1.2 The Calculation

In general, the correlation between pixels, and thus R, depends on the choice of drizzle parameters and geometry and orientation of the dither pattern, and 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 then the situation when pixfrac, p, is set to zero. There is then 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 pixels is assumed to be independent. Let d_{xy} represent a pixel from any of the input images, and let C be in the set of all d_{xy} whose centers fall on a given output pixel of interest. Then it is simple to show that the expected variance of the noise in that output pixel, when p=0, is simply: $\sigma_c^2 = \frac{\sum_{d_{xy}\epsilon C} w_{xy}^2 s^4 \sigma_{xy}^2}{(\sum_{d_{xy}\epsilon C} w_{xy})^2}$

where σ_{xy} is the standard deviation of the noise distribution of the input pixel d_{xy}. We term this σ_c, as it is the standard deviation calculated with the pixel values added only to the pixels on which they are centered.

Now let us consider a drizzled output image where p > 0. In this case, the set of pixels contributing to an output pixel will not only include pixels whose centers fall on the output pixel, but also those for which a portion of the drop lands on the output pixel of interest even though the center does not. We refer to the set of all input pixels whose drops overlap with a given output pixel as P and note that $C \subset P$. The variance of the noise in a given output pixel is then:

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

where a_{xy} is the fractional area overlap of the drop of input data pixel d_{xy} with output pixel o. Here we choose the symbol σ_p to represent the standard deviation calculated from all pixels that contribute to the output pixels when pixfrac = p. The degree to which σ_p^2 and σ_c^2 differ depends on the dither pattern and the values of p and s. However, as more input pixels are averaged together to estimate the value of a given output pixel in P then in C, $\sigma_p^2 \leq \sigma_c^2$. When p=0, σ_p is by definition equal to σ_c.

Now consider the situation where we block average a region of NxN Pixels of the final drizzled image, doing a proper weighted sum of the image pixels. this sum is equivalent to having drizzled onto an output image with a scale size Ns. But as $N_s \gg p$, this approaches the sum over C, or, in the limit of large $N, N\sigma_c$. 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 $N\sigma_p$. Thus we see that: $R = \frac{\sigma_c}{\sigma_p}$

One can therefore obtain R for a given set of drizzle parameters and dither patterns by calculating σ_c and σ_p and performing the division. However, there is a further simplification that can be made. Because we have assumed that the inputs to each pixel are statistically equivalent, it follows that the 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, we see that the weight of an interior pixel must also be independent of p. As a result,

$\sum_{d_{xy}\epsilon C} w_{xy} = \sum_{d_{xy}\epsilon P} a_{xy}^2 w_{xy}$

Therefore, we find that:

$R^2 = \frac{\sigma_c^2}{\sigma_p^2} = \frac{\sum_{d_{xy} \epsilon C} a_{xy}^2 w_{xy}^2 \sigma_{xy}^2}{\sum_{d_{xy} \epsilon P} a_{xy}^2 w_{xy}^2 \sigma_{xy}^2}$

Although R must be calculated for any given set of dithers, there is one case that is particularly illustrative when one has many uniformly placed dithers across the pixel - one 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 above sums become integrals over the output pixels, and thus it is not hard (though somewhat tedious) to derive R. If one defines $r = p/s$ where p=pixfrac and s=scale, then in the case of a filled uniform dither pattern one finds,

if $r \geq 1$

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

and if $r \leq 1$

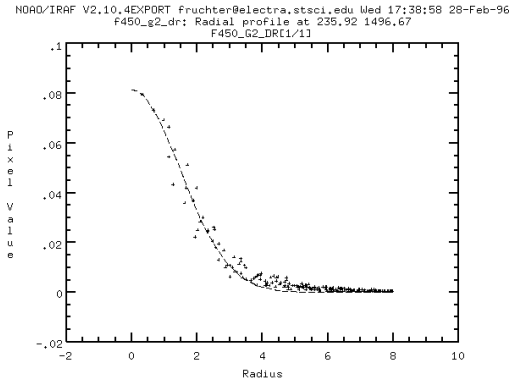
$R = \frac{1}{1 - \frac{r}{3}}$.

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

3.4 Image Fidelity After Drizzling

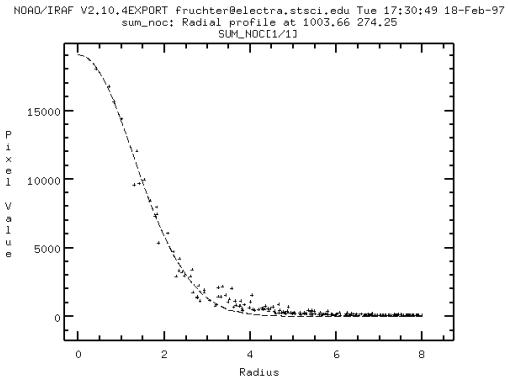
In the drizzle algorithm, the weight of an input pixel in the final output image is independent of its position on the chip. Therefore, if the dithered images do not uniformly sample the field, the center of light in an output pixel may be offset from the center of the pixel, and this offset may vary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

Figure 3.4: PSF Resulting from Non-uniform Sampling



PSF is taken directly from the HDF-N F450W drizzled image and 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: PSF of a Bright Star with Uniform Sampling



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, and clearly shows the improvement in the PSF resulting from the more uniform sampling.

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.

3.5 Photometric Accuracy After Drizzling

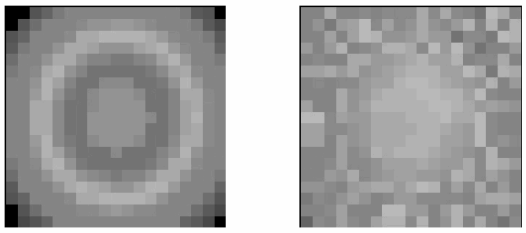
The HST instrument optics geometrically distort the images formed on the detectors: pixels at the corner of each CCD subtend less area on the sky than those near the center. Flatfields for the HST instruments are defined such that, after application of the flat-field, a source of uniform surface brightness on the sky produces uniform counts in each pixel across the CCD. Unfortunately, because of the changing pixel scale across the field, this definition of the flat-field causes point sources near the corners of the chip to be artificially brightened compared to those in the center. For example, in the WFPC2 chips, a star in the corner becomes ~4% brighter than it would have been in the center of the chip.

Fruchter and Hook (1997) carried out a set of tests to study the ability of drizzle to remove the photometric effects of geometric distortion. This involved first creating a

grid of 19x19 artificial stellar PSFs, subsampled by a factor of four, using the TinyTIM WFPC PSF modeling code. The stars were convolved with a narrow Gaussian to approximate the smearing caused by cross-talk between neighboring pixels. This image was then shifted and down-sampled onto four simulated WFPC2/WF2 frames, each with the original WF2 pixel sampling, and dithered in a 2x2 pattern of half-pixel shifts. This process also included multiplying each image by the Jacobian of the WF2 camera geometric distortion, thereby adjusting the counts to reflect the effects of geometric distortion. The amount of data and dithering patterns, therefore, resemble ones that a typical observer might produce (in contrast, the HDF-N contained 11 different pointings.) These four images were then combined using drizzle with typical parameters (output pixel 1/2 of the original WF2 pixels, and a drop size with pixfrac=0.6). The geometric distortion of the chip was removed during drizzling using the polynomial model of Trauger et al. (1995).

Aperture photometry was then obtained on the stars in one of the four simulated input images, and on the stars in the output drizzled image. The results are shown in Figure 3.6, where the photometric measurements of the 19x19 stars are represented by a 19x19 pixel image. The effect of the distortion on the photometry of the input image is very clear - the stars in the corners are up to ~4% brighter than those in the center of the chip. Figure 3.6 shows the results of aperture photometry on the 19x19 grid of stars after drizzling. The effect of geometric distortion on the photometry is dramatically reduced: the r.m.s. photometric variation in the output drizzled image is 0.004 mags. Thus, the removal of the geometric distortion by drizzle can be sufficiently effective to enable aperture photometry to be carried out successfully on resulting images, without the need to correct independently for the geometric distortion by other means.

Figure 3.6: Photometric Results of Drizzling



The figure on the left shows the stars (all of equal intrinsic brightness) as they would appear in a flat-fielded WF image. In order to compensate for the smaller area on the sky of the pixels near the edge of the chip, the flatfield has artificially brightened the stars near the edges and the corners. The image on the right shows the photometric results for these stars after drizzling, which corrects for the geometric distortion.

In practice, observers may not have four relatively well interlaced images but rather a number of almost random dithers, with each dithered image suffering from cosmic ray hits. Therefore another test was carried out, using the shifts actually obtained in the WF2 images of the HDF-N as an example of the nearly random sub-pixel phase that large dithers may produce on HST. In addition, each image was associated with a pixel mask corresponding to cosmic ray hits from one of the deep HST WFPC2 images. When these simulated images were drizzled together, the r.m.s. noise in the final photometry (which does not include any errors that could occur because of missed or incorrectly identified cosmic rays) was less than 0.015 mags.

3.6 Astrometric Accuracy After Drizzling

Drizzle has been found to impart no additional astrometric error on the images beyond our limitations on the ability to centroid on images which contain power that is not fully Nyquist sampled even when using pixels which are half the original size. Tests have been carried out to characterize the astrometric accuracy of drizzle (Fruchter and Hook 1997). The stellar images described in the previous section (Section 3.5) were again drizzled using the shifts obtained in the HDF WF2 F814W images. Both uniform weight files and cosmic ray masks were used. The positions of the drizzled stellar images were then determined with the IRAF task “imexam”, which locates the centroid using the marginal statistics of a box about the star. A box size equal to 6 output pixels, or slightly larger than twice the full-width at half-maximum of the stellar images, was used. A root mean square scatter of the stellar positions of ~0.018 input pixels about the true position was found for the images created both with uniform weight files, and with the cosmic ray masks. However, an identical scatter was produced when the original four-times oversampled images were down-sampled to the two-times oversampled scale of the test images. Thus it appears that no additional measurable astrometric error has been introduced by drizzle. Rather, we are simply observing the limitations of our ability to centroid on images which contain power that is not fully Nyquist sampled even when using pixels half the original size.

For more detailed information on astrometry, see Section 4.3.3 on Astrometric Header Information in Chapter 4.

CHAPTER 4:

Astrometric Header Information

In This Chapter...

4.1 HST Pointing Accuracy and Stability

4.1.1 HST Pointing Accuracy

A knowledge of the capabilities and limitations of HST and its complement of science instruments is of direct importance in deciding whether or not to obtain dithered observations, and what kind of strategies to use if dithering is chosen. A principal issue that must be taken into account when considering strategies for dithered observations is a knowledge of the pointing stability and offsetting accuracy of HST. Regardless of whether integer or sub-pixel dither offsets are being considered, it is important to understand the level to which positioning accuracy can be achieved by the acquisition and tracking system of HST. Specifically, the following issues must first be addressed when considering a sequence of multiple, dithered exposures of the same target with HST, which can be broadly divided into three types of observing program structures:

- 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

Our statistics on spacecraft behavior are continually improving. There have now been several large campaigns that have made extensive use of HST dithering to optimize the science, for example 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), 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 provide an excellent body of information about the precision and repeatability of HST offsets, as well as the tracking stability of the telescope when no offsets are commanded (e.g., multiple exposures at the same location). Drawing on our experience with these observing programs, we now describe in more detail the HST pointing and stability characteristics for each of the above observing modes, particularly in terms of the positional accuracy of the spacecraft when performing offsets for dithered observational programs. Gilliland (2005) contains a thorough analysis of the datasets which establish the values given in the table below.

Table 4.1: Typical HST Pointing and Stability Characteristics

Observing Scenario (with fine lock on two guide stars)	Type of Program	Typical 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

4.1.2 HST Tracking Stability at a Single Location

During each orbit, thermal variations in the telescope cause structural variations called “breathing” which leads to changes not only in the optical telescope assembly (OTA) but also in the way in which the Fine Guidance Sensors (FGS) track the guide stars. The breathing in the instrument optics manifests itself as time-dependent

changes in the shape and centroid of the PSF across the image, due to the changing focus.

The 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 both guide stars, in which case the drifts would be mainly related to thermal variations and jitter, with effects predominantly in the form of translations. Some small amount of rotation may also occur during the orbit, typically less than a few hundredths of a pixel across the science instrument. The typical r.m.s. tracking accuracy is generally of the order of 2-5 mas or less throughout an orbit, and can always be verified post-facto for a particular observation by examining the jitter files that form part of the archival dataset.

In some observations, however, fine lock is achieved successfully 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 by ~1-5 mas/sec about the guide star. The rate is typically ~1.5 mas/sec, but up to 5 mas/sec can be seen on rare occasions. This will manifest 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 will depend 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 - 20 arcminutes. For these instruments, the maximal scenario of a rotational drift of 5 mas/sec would produce a total translation during one orbit ranging between ~25 - 85 mas. For WFPC2 this maximal shift would be ~50 mas.

Thus, before proceeding with the analysis of dithered data, it is always advisable to examine the jitter data products after the observations to confirm whether two-FGS fine lock was successfully achieved during the observations. If this was the case then the expected translational shifts due to FGS drift should be less than ~3 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 further details on the jitter files and other data products, as well as how to extract the relevant information from these files.

4.1.3 Precision of Commanded Offsets

If the primary reason for dithering is to avoid bad pixels or improve the 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 offsets typically agree with the commanded offsets to an r.m.s. within ~2-5 mas during a single orbit with good lock on both guide stars, ranging up to ~10-15 mas from visit to visit over many days. Occasionally, the actual offsets can differ substantially from the commanded offsets by ~0.1-0.2 arcseconds or more, and with field rotations up to 0.1 degree, as a 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 somewhat larger dither offsets, up to a few arcseconds, are required in order to bridge inter-chip gaps between detectors, as in the multiple cameras of

WFPC2 or the two detectors in ACS/WFC. Offsets of this size are unlikely to present any problems with pointing precision, although 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 the guide stars being moved out of the FGS apertures, depending upon the exact configuration of the primary and secondary guide stars. This would necessitate a full acquisition of new guide stars, with substantial associated overhead, and loss of pointing repeatability as a result of the relative positional uncertainties in the guide star catalog (~0.2 - 0.5 arcsec). Such large offsets are more appropriate for mosaicing programs where large areas are being mapped, and would thus involve a fundamentally different observational design than programs involving small dither offsets.

4.1.4 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. At the start of the first orbit of a visit, a full guide star acquisition is performed. For each subsequent orbit in the same visit, after the telescope exits from occultation, a re-acquisition of the guide stars is carried out. Since a re-acquisition slews HST in order to force the guide stars to reside in exactly the same location in the pickles as in the previous orbit, the science instrument is typically placed successfully to within ~5 - 20 mas of its location during the previous orbit. This is generally sufficient to perform sub-pixel dithers reliably with most of the HST instruments that have pixel sizes of the order ~0.05 - 0.1 arcseconds. Thus, we generally recommend that the observing schedule be designed to fit all dithered observations of a given target into a contiguous set of orbits within a single visit, whenever possible, to provide improved relative image registration.

4.1.5 Roll Angle Repeatability Over Multiple Visits

Some observing programs are sufficiently large that they necessitate dithered observations of the same target over many orbits. In such cases, breaking up the observations into several separate visits is unavoidable, since single visits are usually constrained by scheduling limitations to contain no more than five orbits. If these multiple visits are scheduled across different dates, then some shifts may be present in the images even when specifying the same pointing, the same guide stars, and same ORIENT as previous visits.

At the start of a new visit, HST first sets up the specified roll for the observation using the gyros and carries out a full acquisition of the dominant guide star, after which it acquires the sub-dominant guide star and tracks it in fine lock. The PCS then preserves this roll angle for the remainder of the visit. In most cases the difference between the desired roll and the actual roll angle will be less than around 0.003 degrees, corresponding to a positional shift of about 73 mas at the sub-dominant guide star (assuming a separation of 1400 arcseconds between the two guide stars). At the

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 with the same 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 degrees and upward can occur (i.e., translational shifts above 100 mas).

The jitter files, which allow the roll angle to be determined based on guide star locations in the FGS, can be used in the case of visits with the same guide stars and same roll, to determine quite accurately the actual roll change that was incurred between visits.

4.2 Detector Plate scales and Geometric Distortion

4.2.1 Introduction

Apart from the uncertainties introduced by the telescope pointing accuracy when performing dither offsets, another effect which needs to be considered is the varying plate scale across each of the detectors as a result of the geometric distortion produced by the telescope optics. For example, the distortion across the WF chips in WFPC2 amounts to ~2% at the chip corners, thus a commanded telescope offset of exactly 25 pixels at the chip center would create an additional 0.5 pixel shift at the edge of the detector. In other words, objects near the edge of the chip would be subsampled by half a pixel while those near the center would not be subsampled at all, and this results in continuously varying degrees of subsampling across the image. This means that detector distortion leads to non-uniform pixel subsampling when dithering.

This additional offset introduced by geometric distortion toward the edges of the chip scales linearly with the size of the commanded telescope offset. Thus, a dither of only 5 pixels at the center (instead of 25 pixels) would produce a much less severe error of only 0.1 pixels additional shift at the edge of the chip, and the sampling would remain approximately constant across the detector. This is another reason why large dithers are discouraged - although the drizzle software is capable of dealing with changing pixel shifts and different degrees of subsampling across the image, the scientific interpretation of the data could still be impacted by the fact that not all the objects are equally subsampled. Instead, the resulting analysis is greatly simplified if small dithers are used, thereby keeping such sampling differences relatively small across the chip (e.g., < 0.1 pixel). However, the two chips that compose the WFC in ACS are separated by a gap of order 2.5 arcseconds (~50 WFC pixels). As a result, many ACS dither strategies involve the use of offsets sufficiently large to allow the detectors to cover this gap and varying degrees of subsampling across the detector will be a necessary consequence.

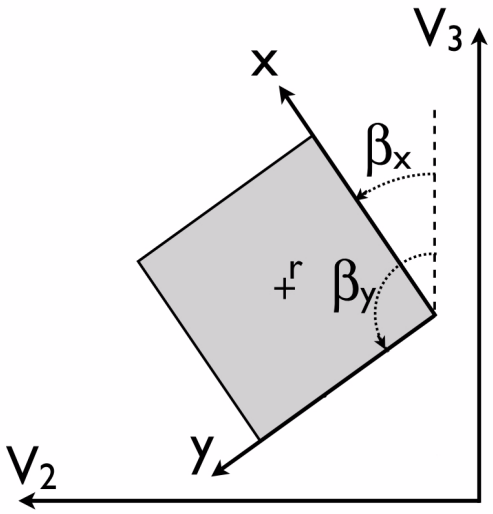
4.2.2 Summary of Detector Plate Scales

Multiple folds in the light path are used in the HST instruments to direct the light to selected optical elements and the detector(s) in a way that achieves the desired focal ratios within the confines of the optical bench envelope, while minimizing throughput loss and image degradation. In some instruments, the focal surfaces are far from normal to the principal rays. This results in an image of the sky which is distorted in a large but predictable manner, depending on the tilt of the detector plane and the curvature of the folding optics. To first order, the tilt causes a square pixel on the detector to be projected as a rectangle (for rotation of the detector about one of its axes) or as a rhombus (for rotation of a square detector about one of its diagonals) or more generally as a parallelogram. The curvature of the folding optics introduces higher order terms in the distortion, causing the projected dimensions of a pixel to vary somewhat with location on the detector. The current best-determined values of the projected pixel dimensions are stored in the Science Instrument Aperture File (SIAF), specifically defined for a single pixel at specified reference locations on the detector. The geometric distortion model appropriate to a given detector then allows the calculation of projected pixel dimensions at other locations on the detector.

The Instrument Apertures tables contain parameters relating the instrument coordinates, pixels or deflection offsets, to the telescope coordinate frame. This frame, designated V2,V3, represents the projected position on the sky. If the telescope roll is zero, then V3 points North, and V2 points East. (The reason for the unconventional axis order is that V2 and V3 are part of a three-dimensional coordinate system with V1 being approximately along the optical axis.) The (X,Y) coordinate system relates to each instrument aperture and the angles β_x and β_y are measured anti-clockwise from the V3 axis to the x and y axes. A reference point is chosen, near the center of the aperture which has the coordinates (x_r, y_r) in the instrument frame, and $V2_r, V3_r$ in the telescope frame. The reference point is the fiducial in the focal plane that HST will endeavor to place the target specified in the proposal. The scales s_x and s_y in x and y directions are not necessarily equal and are each tabulated. For most HST imaging instruments, the x and y axes are flipped with respect to the V2 and V3 axes (as shown in Figure 4.1) ($\beta_y > \beta_x$), and their projection onto the V2,V3 frame may not be orthogonal, but this is expressed by the β_x and β_y angles and the transformation formula remain valid. The transformation from any point (x,y) to V2,V3 is given to first order by:

$$V2 = V2_r + s_{xx}(x - x_r) + s_{yy}(y - y_r)$$
$$V3 = V3_r + s_{xx}(x - x_r) + s_{yy}(y - y_r)$$

Figure 4.1: Reference Angles for the SIAF Instrument File



Visual representation of the frame of reference for angles in the SIAF instrument file. $\beta_y > \beta_x$ for ACS, NICMOS, STIS, WFC3, WF/PC1, WFPC2.

The x and y dimensions of the reference pixels in the V3 coordinate system for each of the commonly used imaging detectors are tabulated in the table below. Also listed in the table are the angles of the x and y axes of each detector, corresponding to the CCD rows and columns, measured counter-clockwise from the +V3 axis of the spacecraft (beta-x and beta-y). These dimensions are measured for the reference pixel. As a result of geometric distortion the orientation and scale of the pixel axes may change somewhat across the detector. The last column in the table lists the approximate maximum change in scale from the center pixel to elsewhere on the detector (the largest changes are generally seen between the centers of the chips and the corners). While the values in this table are intended to be current, the latest positional information can always be obtained directly through the Observatory Support Group's pages on apertures:

http://www.stsci.edu/observatory/apertures

and pointing

http://www.stsci.edu/observatory/pointing

Please note that the values for WFC3 and COS in this table are preliminary, being derived purely from the pre-flight measurements, and are subject to change pending on-orbit calibration programs. All of the numbers detailed in the table below are a snapshot of the current SIAF information as of the printing of this handbook. The SIAF file itself is an internal table which is used in the operational science database at the institute, information in which the users would be the most interested is available from the Web page listed below. If this information is insufficient to meet your needs, please contact the Space Telescope Science Institute help desk.

http://www.stsci.edu/hst/observatory/apertures/siaf.html

Table 4.2: Pixel Scales of the Primary HST Instruments

Instrument	Aperture (pixels/side)	FOV (arcsec/side)	x-scale (“/pixel)	y-scale (“/pixel)	beta-x (deg)	beta-y (deg)	Detector Tilt (deg)	Distortion Center-Corner
WFPC2	PC1 (800^2)	36	0.04552	0.04550	134.908	224.908		1%
	WF2 (800^2)	80	0.09950	0.09959	224.388	314.388		2%
	WF3 (800^2)	80	0.09957	0.09948	314.698	44.698		2%
	WF4 (800^2)	80	0.09953	0.09963	45.258	135.258		2%
ACS	WFC (4096^2)	200	0.0496	0.0495	92.06	177.53	22	8%
	HRC (1024^2)	29	0.0284	0.0248	-84.12	0.08	25	11%
	SBC (1024^2)	30	0.0337	0.0300	-84.56	-0.11	25	11%
NICMOS	NIC1 (256^2)	11	0.04320	0.04302	225.312	315.312		0.7%
	NIC2 (256^2)	19	0.07600	0.07532	224.507	314.507		0.2%
	NIC3 (256^2)	52	0.20375	0.20301	224.851	314.851		0.6%
STIS	CCD (1024^2)	52	0.05076	0.05077	45.056	135.056		0.3%
	NUV (1024^2)	25	0.02475	0.02475	45	135		1%
	FUV (1024^2)	25	0.02474	0.02474	45	135		0.7%
WFC3	UVIS (4096^2)	162	0.03970	0.03952	-41.189	44.877	21	1.7%
	IR (1014^2)	136x123	0.13549	0.12108	-44.996	45.004	24	3.0%
COS	LFPSA	2.5 (diam)	0.022	0.094	135	145		
	LNPSA	2.5 (diam)	0.0258	0.0258	135	145		
	LFBOA	2.5 (diam)	0.022	0.094	135	145		
	LNBOA	2.5 (diam)	0.0258	0.0258	135	145		

4.2.3 Detector Distortion Models

To date, the most detailed empirical HST distortion geometry models are those that have been created for WFPC2, where distortion across the cameras is introduced not only by the HST OTA but also by the re-imaging optics within the instrument. However, the general form of the functional description used for WFPC2 has also been applied to the other instruments. The detector distortion models can be generally characterized in terms of the dependence of the true location (X,Y) of an object, (e.g., in arcseconds), as a function of the measured pixel position (x,y) of the object on the

detector relative to some reference pixel (x_0, y_0), usually as a polynomial of order "m" in "x", in "y", and including all their cross-terms:

$X = \sum_{i=0}^{m} \sum_{j=0}^{i} a_{ij}(x - x_0)^j (y - y_0)^{i-j}$

$Y = \sum_{i=0}^{m} \sum_{j=0}^{i} b_{ij}(x - x_0)^j (y - y_0)^{i-j}$

Explicitly, these distortion polynomials expand out into:

$X = a_{00} + a_{10}y + a_{11}x + a_{20}y^2 + a_{21}xy + a_{22}x^2 + a_{30}y^3 + a_{31}xy^2 + a_{32}x^2y + a_{33}x^3 + ...$

$Y = b_{00} + b_{10}y + b_{11}x + b_{20}y^2 + b_{21}xy + b_{22}x^2 + b_{30}y^3 + b_{31}xy^2 + b_{32}x^2y + b_{33}x^3 + ...$

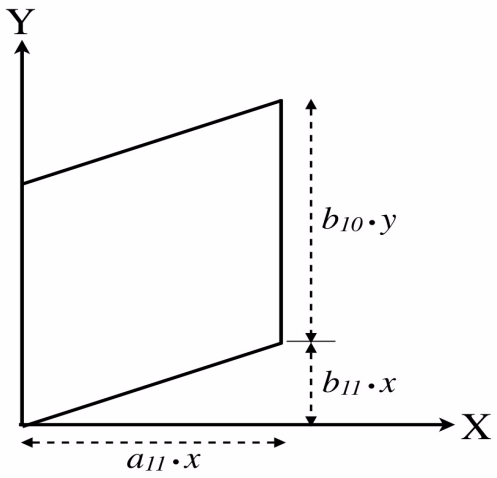
The distortion coefficients characterizing each instrument are stored in the form of tables that are maintained by STScI, and are incorporated into the IRAF/STSDAS dither software used to analyze the data. See the description of the IDC table (Section 4.2.4) for more details. The linear component of the distortion generally reduces to:

$X = a_{11}x$

$Y = b_{10}y + b_{11}x$

a_{00} and b_{00} are zero since the two coordinate systems have the same origin. a_{10} is zero when the Y axis is defined to lie along the y axis of the detector, which is generally the case. If the projected x and y axes are not perpendicular (b_{11} is non-zero), a displacement in x on the detector has an X component and a Y component, while a displacement in y on the detector has only a Y component, as shown in the figure:

Figure 4.2: Projection of Detector Coordinates



Projection of x,y detector coordinates onto the X,Y frame using first order distortion terms.

WFPC2

The distortion models for WFPC2 are generally defined in terms of polynomials expressed separately for each detector, thereby providing four transformations between the detector coordinates and a common rectilinear system.

- The Trauger (1995) solution is derived from a model of ray tracing through the camera optics, and includes wavelength dependence parameterized by the refractive index of the MgF_2 field-flattener windows. The geometric distortion in this model is defined in terms of third-order polynomials of the above form,

derived separately for each detector and expressed relative to the central pixel of each chip. The r.m.s. residuals of this solution are of the order ~5 m.a.s. across each of the cameras, with larger systematics extending up to ~20 m.a.s. toward some of the corners of the detectors.

- The solutions of Gilmozzi et al. (1995) and Holtzman et al. (1995) are based on multiple overlapping observations of globular clusters (NGC 1850 and w Centauri respectively), and are derived for a single wavelength (555nm). An important difference between these two solutions is that the Gilmozzi solution contains combined Legendre polynomials of third order, thus it includes individual terms up to 6 orders, while the Holtzman solution uses a complete third-order polynomial in *x* and *y*. The Holtzman solution is also defined in terms of a single meta-frame coordinate system, relative to the center of the PC camera. The r.m.s. residuals of this solution are of the order 2 m.a.s. across the WF chips and 5 m.a.s. across the PC, although some systematics up to 10 - 15 m.a.s. are present toward the edges of the chips.
- The solution of Casertano and Wiggs (2001) is based on more recent observations of ω Centauri, obtained in June 1997. In addition to providing an updated solution that takes into account detector motions since 1995, this work also made use of a much larger number of stars to substantially improve the sampling of the distortion across the entire area of each detector, and reduce the large systematic errors toward the corners. The resulting solution (derived at 555nm) has r.m.s. residuals of ~1.3 m.a.s. across the entire extent of each of the WF chips, and ~3 m.a.s. across the entire PC chip, thereby effectively eliminating the systematics that had been present in the earlier solutions.

These solutions have been converted into reference files (Section 4.2.4) for use in the calibration pipeline. Work to refine these solutions for a broader wavelength range has been published in a report by the WFPC2 team (Kozhurina-Platais 2003). In addition, time-dependent effects of the chip-to-chip positions (Section 4.2.4) have been calibrated and included as a separate reference file, the OFFTAB.

ACS

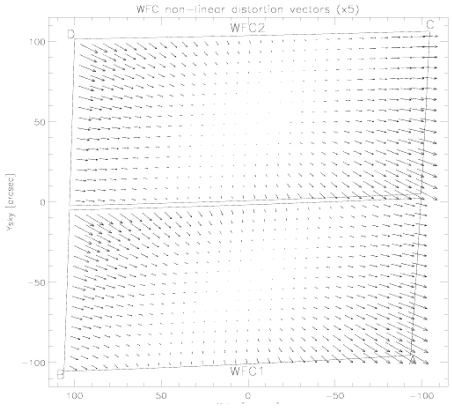
The ACS/WFC is significantly more distorted than the WFPC2, STIS or NICMOS cameras. The magnitude of the distortion is predominantly a result of the large format of the detector, together with its off-axis location in the HST focal plane. The distortion of ACS/WFC is a combination of an 8% elongation along the detector diagonal, resulting from the detector inclination with respect to the optical axis, together with an increasing radial distortion away from the center of the WFC. The ACS distortion, characterized in terms of polynomials (Hack & Cox 2000), has been described in the ACS Data Handbook which can be downloaded from:

`http://www.stsci.edu/hst/acs/documents/handbooks/`

In particular, the WFC distortion is illustrated in this figure, a vector displacement diagram which shows the contribution of the non-linear part of a quartic fit to the data. The vectors represent the degree of distortion to be expected in the WFC beyond the

directional dependence of the plate scale. For display, the vectors are magnified by a factor of 5 compared to the scale of the x and y axes. The largest displacement indicated at the top left corner of the figure is ~82 pixels or about 4 arcseconds.

Figure 4.3: Non-linear Component of the ACS WFC Detector



Non-linear component of the ACS distortion for the WFC detector using a F475W quadratic fit. Note that this figure is rotated 180 degrees with respect to the default orientation of the drizzled product, where WFC2 would be the lower half of the detector.

This distortion model has been stored in the IDCTAB reference file (Section 4.2.4) for use by the MultiDrizzle software to correct the distortion in ACS observations and allow for the combination of dithered images.

The strong distortion implies that the degree of subsampling will vary across the image even for small dithers. For example, an offset of 5 pixels at the center of the WFC already introduces an additional shift of 0.4 pixels near the edge of the detector. A larger dither, e.g. 12 pixels at the center, will correspond to an integral shift near the edge (one entire additional pixel) but will provide half-pixel subsampling midway between the center and the edge. Thus, varying degrees of subsampling across the image will be unavoidable with any single pair of dither offsets. Obtaining a larger number of offsets can help produce more uniform subsampling across the entire ACS field of view.

A small change in the skew of the ACS images has been detected over time, as have small distortions which are higher frequency than can be corrected by the low-order polynomial presently incorporated into the IDCTAB reference files. These effects are discussed in more detail in (Section 4.2.4).

NICMOS

For NICMOS the degree of distortion is somewhat less than for WFPC2, partly owing to the smaller areas covered by the NICMOS detectors (Cox et al. 1997). For all three cameras (NIC1 and NIC2, as well as NIC3 when used at the optimal focus position), the distortion is less than 1 pixel at the chip corners relative to the center and appears to be best modeled by a quadratic polynomial. However, the NIC3 camera at its nominal out-of-focus position has substantially higher geometric distortion, and the initial NIC3 geometric distortion solution reported by Cox et al. (1997) was

compromised by the effects of vignetting which forced those authors to discard stellar position measurements over a significant portion of the field of view.

The geometric distortion for NIC3 was remeasured during the January 1998 NIC3 refocus campaign, at which time the Field Offset Mirror was repositioned to substantially reduce the amount of vignetting across the field. A new set of distortion coefficients was measured using these data. These are reported in Chapter 5 of the *NICMOS Data Handbook* (Dickinson et al. 2002), and have been used to derive the in-focus NIC3 geometric distortion coefficient files.

The NICMOS plate scale has been more variable than that of the other instruments, particularly during the early part of its lifetime, due to path length changes introduced by the NICMOS dewar anomaly. More detailed information on these and other geometric effects are all available from the NICMOS Data Handbook which can be downloaded from this Web page:

`http://www.stsci.edu/hst/nicmos/documents/`

The latest distortion models for NICMOS have also been converted into reference files(Section 4.2.4) for use by PyDrizzle and MultiDrizzle to correct the distortion of NICMOS data.

STIS

The geometric distortions for the three detectors of STIS in imaging-mode have been measured on-orbit (Malumuth and Bowers 1997; Walsh, Goudfrooij and Malumuth 2001). The CCD, NUV-MAMA and FUV-MAMA display maximum distortions less than about 0.3 - 1% at the edges relative to the detector center. The distortions for these cameras have been modeled in terms of cubic polynomials, similar in their functional form to those used for NICMOS and WFPC2, and stored as IDCTAB reference files (Section 4.2.4) for use with the MultiDrizzle software.

WFC3

Wide Field Camera 3 is due to be installed during Hubble's fourth servicing mission. An initial distortion model has already been created for each channel, UVIS and IR. These models are described in the WFC3 Data Handbook, which can be downloaded from:

`http://www.stsci.edu/hst/wfc3/documents/`

The UVIS detector is tilted about one of its diagonals with respect to the light path. This tilt produces a projected rhombus with a diagonal elongation of ~7%, reminiscent of the projected parallelogram of the ACS/WFC detector. The IR detector is tilted about its x axis, creating a projected rectangle of elongation ~10%. Both WFC3 detectors have substantial non-linear geometric distortion in addition to these projected elongations. The maximum displacement from the rhomboidal projection of the UVIS detector, occurring at two opposite corners, is about 30 pixels (1.3 arcseconds) along the diagonal. The maximum displacement from the rectangular projection of the IR detector, occurring at all four corners, is about 10 pixels (1.4 arcseconds) along the diagonals.

The full distortion solutions will be updated during Servicing Mission Orbital Verification (SMOV4) in the Winter of 2008/9. For the most current information, please see the WFC3 Web page at:

http://www.stsci.edu/hst/wfc3

4.2.4 The IDC Table

The Instrument Distortion Correction table file (represented by the FITS header keyword IDCTAB) was created to support the description of the geometric distortion models for the instruments. The IDCTAB is stored as a FITS file with a table in the first extension. It contains the minimum information necessary to correct each pixel in the given science image to it's true location on the sky. This takes the form of the coefficients for the Science Instrument Aperture File (SIAF) polynomial fit. The SIAF file contains descriptions of the distorted and undistorted aperture positions. Any instrument whose distortion has been modeled using a polynomial fit can use this table as a reference file.A complete description of the IDC reference file can be found in Hack & Cox (2001).

This file can support any order polynomial, with the order of the polynomial being specified in the primary header using the NORDER keyword. The table itself consists of separate rows for each configuration which has been calibrated. The first few columns of each row indicate the chip identification (of a multi-chip detector) and filter selection for which the coefficients of this row apply. The next 6 columns specify the physical size of that chip and the position of the reference pixel on that chip both in pixels on the chip and in arcseconds on the sky relative to the center of the telescope's field of view; in other words, it's position in the V2 and V3 coordinate system. The column labeled THETA describes the orientation of the detector's Y axis relative to the telescope's V3 axis. This allows each chip of a multi-chip detector, such as WFPC2 or ACS, to be properly oriented in the final output mosaic. The remaining columns specify undistorted plate-scale and the coefficients of the polynomial which need to be applied to the chip to correct for the distortion. These reference files were originally developed for use with ACS and STIS imaging data, with IDCTAB reference files being subsequently generated for WFPC2 and (if not already available) NICMOS.

The coefficients from the IDCTAB reference file form the basis for the description of the distortion model, however, the MultiDrizzle software and the SIP convention require the model in different forms. The reference file allows for updates to the distortion model as new calibrations produce higher-quality models, updates which can be easily provided to the HST operational calibration pipeline and made available for use in archive retrievals at any time. The reference files themselves, though, only serve as the primary source of the model, while the MultiDrizzle code and "makewcs" in particular interprets those models and updates the headers with the distortion coefficients and generates ASCII files for use with IRAF's 'drizzle' task to actually perform the re-sampling. The distortion model will be written to the header using the SIP convention (Section 4.2.5) and will eventually serve as the sole source for the model without having to rely on the actual reference file itself once the SIP keywords are updated.

Time Dependent Motions of WFPC2

The IDCTAB reference files for WFPC2, however, does not contain the complete description of the model as the position of each chip varies over time relative to the other chips. Calibrations have determined the long-term movements of the chips as described in WFPC2 ISR 2001-10. The WFPC2 Data Handbook section on astrometry, reports that these motions have been tracked using K-spot imaging and started out with shifts of 2-4 pixels in the first few months of operation. Those motions settled down to a rate of about 0.1 pixels/year, which has added up to more than a pixel over the course of the operational life of WFPC2.

These motions are not accounted for in the WCS information in the raw (d0h) or calibrated (c0h) WFPC2 images generated in the archive by CALWP2. However, the generation of drizzled WFPC2 products in the archive updates the WCS information in the headers to account for these time-dependent motions. Those time dependent motions have been calibrated and recorded in an ancillary reference file, specified by the header keyword OFFTAB, which gets used along with the IDCTAB to completely define the distortion for the observation. This secondary reference file contains as its first column the date for which that particular row can first be applied to WFPC2 data. There are sets of rows corresponding to various dates when the chip positions have been determined. The remaining columns specify the chip and the V2 and V3 coordinates of the chip's reference position. The positions are then determined for any given observation date by linearly interpolating between V2,V3 positions corresponding to dates that bracket the observation date. The latest calibrations, available as of Fall 2008, will support chip-to-chip alignment to within 0.2 pixels relative to the WF3 chip for all observations up until the removal of WFPC2 during SM4.

Time Dependent Distortion for ACS

The IDC table refers to the state of the geometric distortion at a given date. Observations taken at different times and/or different orientations, however, exhibit some residual distortions. The residual distortions for ACS WFC observations have been determined to have a time-dependent component to them, as reported by ACS ISR 2007-08. The time-dependent correction to the polynomial coefficients gets derived from 2 coefficients computed using the equations reported by Anderson (2007) in ACS ISR 2007-08:

$$\alpha = 0.095 + 0.090 * (date - 2004.5)/2.5$$

$$\beta = -0.029 = 0.030 * (date - 2004.5)/2.5$$

where the *date* used in the calculation refers to the observation date in decimal years.

The values of α and β get recorded in the header keywords TDDALPHA and TDDBETA when they are derived by the "makewcs" task (described in the section on Implementation) when updating the header to be consistent with the latest IDCTAB model. The SIP coefficients written out by "makewcs" can be updated to include this time-dependence so that any SIP-aware software (like DS9) should correctly account for this effect. This is not turned on in pipeline use prior to SM4. The keyword TDDCORR will be added to the ACS/WFC primary image header with a value of

"PERFORM" to turn on this correction for pipeline use. However, "makewcs" can be run to apply these corrections to the SIP coefficients when needed by manually adding the TDDCORR keyword to the ACS/WFC image primary header.

ACS ISR 2007-08 also identified errors in the overall distortion solution at a level of a fraction of a pixel that were higher frequency than can be represented in the low order polynomial presently used by the IDCTAB or SIP header (see below). These are represented by two 32×64 tables which are linearly interpolated for the position at any given pixel in a 2048×4096 ACS chip. At present these corrections are represented as full 2048×4096 reference images in the archive, with a suffix of "_dxy". Under the present transition to the use of the fits header to represent the geometric distortion of HST images, the linearly interpolated look-up table will be placed in the fits image header and will replace the "_dxy" images.

Addition of the look-up table and time dependent distortion corrections substantially improves relative ACS astrometry; however, even with the full Anderson (2007) solution, the user can expect up to ~ 0.03 pixel or ~ 1.5 mas systematic errors in the astrometric distortion correction near the edges of the field. These errors are due to time variable changes in the distortion which have yet to be characterized. Comparison of two ACS images taken at different times or orientations may show somewhat larger offsets near the edges, since the small remaining systematic errors may add rather than cancel.

4.2.5 SIP Coefficients in the Image Headers

The distortion model used to calibrate HST images can be represented primarily by a polynomial, and in the case of ACS, a 4th-order polynomial. This polynomial has been calibrated by the instrument teams at STScI and recorded in reference files for use with HST data. This information can now be included in the headers of each image using the Simple Imaging Polynomial (SIP) convention as implemented for Spitzer data (Shupe et al., 2005), and understood by many packages, such as DS9, to report distortion-corrected positions from the image. These SIP coefficients allow MultiDrizzle to operate without requiring external reference files for describing the polynomial distortion model for each image.

SIP Convention

As described in the paper by Shupe et al. (2005), the use of the SIP coefficients can be recognized by 2 changes to the image header. The format of the coordinates described in the FITS header is provided by the CTYPE1 and CTYPE2 keywords. The extra characters -SIP are simply appended to the existing keyword values to report that the SIP transformation should be used for interpreting the coordinates of objects in the image. For HST images, this results in:

```
CTYPE1 = 'RA---TAN-SIP' / the coordinate type for the first axis
CTYPE2 = 'DEC--TAN-SIP' / the coordinate type for the second axis
```

The pixel coordinates are transformed to sky coordinates using the CD matrix specified in the header of the image. The CD matrix already includes the linear terms

of the distortion model as well as any skew, rotation and scaling. The non-linear terms, defined as “f(u,v)” and “g(u,v)”, get applied using the following transformation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_1 \end{pmatrix} \begin{pmatrix} u + f(u,v) \\ v + g(u,v) \end{pmatrix}$$

where “u” and “v” are the initial position from the image.

The non-linear coefficients are recorded in the image header using keywords “A_p_q” and “B_p_q” for the polynomial terms $u^p v^q$. The polynomial functions for use in the transformation then come out to be:

$f(u,v) = \sum_{p,q} A\_p\_q u^p v^q$ where p+q <= A_ORDER, and

$g(u,v) = \sum_{p,q} B\_p\_q u^p v^q$ where p+q <= B_ORDER

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

Implementation

The SIP coefficients have been implemented for use with HST data through the use of the Python task “makewcs”. This task has been released with STScI_Python and MultiDrizzle runs this task when the parameter “updatewcs=yes”. However, the CTYPE keywords were not being appended with the -SIP string until more software was available to interpret the SIP coefficients. This default behavior changed during the Summer of 2008 when MultiDrizzle was modified to work exclusively with the SIP convention. Any HST data processed with versions of MultiDrizzle (and “makewcs”) installed with STScI_Python 2.6 (released in Feb. 2008) or earlier can append -SIP to the CTYPE keywords in all science extensions to allow DS9 and other SIP-aware software to use the SIP coefficients recorded in the science headers.

4.3 The CD Matrix

4.3.1 Definition of the CD Matrix

The astrometric information in FITS images (also referred to as the WCS) is stored in the header using a standard set of keywords. The reference location is defined by the following keywords:

- CRVAL1: defines the right (α) ascension of the reference pixel
- CRVAL2: defines the declination (δ) of the reference pixel
- CRPIX1: the x location of the reference pixel
- CRPIX2: the y location of the reference pixel

The plate scale and rotation of the image is contained in the CD MATRIX (CD?_? keywords).

- 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

4.3.2 Computation of the CD Matrix

These keyword values get computed using the operations:

$$\begin{pmatrix} CD1\_1 & CD1\_2 \\ CD2\_1 & CD2\_2 \end{pmatrix} = scale * \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} A10 & A11 \\ B10 & B11 \end{pmatrix}$$

The plate scale, "scale", represents the average plate scale (in decimal degrees) for the reference pixel after removing all distortion. The value of θ originally comes from the PA_V3 keyword; 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 it's field of view. 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 then gets 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; for example, the 45 degree rotation of the WFPC2 field-of-view relative to the V2-V3 axes. The orientation for each chip's reference position, θ , then 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 gets recorded in the ORIENTAT keyword.

The terms A10,A11,B10,B11 refer to the linear terms of the distortion model as defined in the polynomial stored in the IDC table reference file. These terms represent the skew, change in orientation and plate scale in X and Y at the reference pixel of the image only. More information on these can be found in the section dealing with the geometric distortion and update of the SIP keywords. The right ascension (R.A.) and declination (Dec.) of the image target is stored in the header of HST images using the keywords RA_TARG and DEC_TARG, however, these are the same across all images in a dithered set since the target itself is not changing. However, the R.A. and Dec. of the image reference pixel will always be unique for each image. Therefore, the offset between the images (the x and y shift) can be determined by retrieving the values of the associated reference pixel keywords.

4.3.3 Updating the WCS Header Information

The World Coordinate System information contained in the header of each image is only initially as precise as the available catalog information for the guide stars used for pointing. The typical accuracy of the guide star positions based on Guide Star Catalog V1 was about 1 arc-second. Thus, observations taken of the same field at different orientations (requiring different guide stars to be used) can see offsets between their observations of 1-3 arcseconds. Users may choose to calculate additional shifts to align their images more accurately and these delta shifts can be incorporated back into the WCS information in the header. This facilitates proper translation of pixel to sky coordinates for science analysis, and allows for all the necessary shifting information for each image to be stored in the header of the image itself for future alignment reference.

It is also important to update the WCS keywords in the header of the image based on the provided distortion correction solutions (such as geometric distortion and velocity aberration) that have been applied to the data through the drizzle process. The full distortion solution, with velocity aberration correction, gets applied automatically inside the MultiDrizzle code-set through the use of "makewcs". The 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 the reference pixel, but also the linear terms of all distortion corrections. Higher-order terms of the distortion model then get reported in the remainder of the SIP keywords.

Users who wish to apply their own distortion corrections should be careful to update these keywords.

CHAPTER 5:

Drizzle Software

In This Chapter...

5.1 Available Software

Several software packages have been developed to implement the drizzle algorithm, to correct images for distortion, to identify cosmic rays and to align and combine images into a cosmic ray cleaned, distortion-free product. The software was originally developed as a set of Fortran routines which were available both as stand-alone executables and as **IRAF** tasks in the Dither package. The original drizzle software was developed as a collaborative effort by Andy Fruchter and Richard Hook. The main code was written in Fortran77 and incorporated into **IRAF** with **STSDAS** v2.0, inside the **Dither** package. This set of code formed the basis for a more automated task for aligning images to produce distortion-corrected images; namely, the Python task **PyDrizzle** (Section 5.3). Finally, all the steps required to identify cosmic rays using the aligned images produced by **PyDrizzle** were automated in the Python task **MultiDrizzle** (Section 5.4). The Python tasks can only be run from the PyRAF environment while the original **IRAF** tasks can be run from either **PyRAF** or the original **IRAF** command language (also known as the "cl").

5.1.1 Original IRAF Tasks

There are many tasks which are available in the **PyRAF** dither package. Many of them represent the original interface to the **drizzle** codeset in **IRAF** and are no longer supported - though their functionality has been included in the new scripts which are used by **PyDrizzle** and **MultiDrizzle**. It is important for the user to realize that use of the non-supported functions may result in inferior output products since the most

current updates to the code and methodology are only implemented in the Python functions called by PyDrizzle and MultiDrizzle.

The following functions are the original **IRAF** interface routines and are no longer being fully supported:

```
  avshift - Averages the shifts measured on the 4 WFPC chips.
crossdriz - Builds a set of cross-correlation images
            (shift + rotation).
    deriv - Take the absolute derivative of an image.
  driz_cr - Create cosmic ray mask based for dithered data.
loop_blot - Runs blot for a given list of input and output images.
loop_driz - Runs drizzle for a list of input, output,
            and mask images.
  offsets - Builds cross-correlation image (shift only).
mask_head - Attach mask names to image headers;
            invert mask if requested
   precor - cosmic ray processing.
  rotfind - Finds rotation angle from a set of cross-correlation
            images.
shiftfind - Finds X,Y shifts in a cross-correlation image.
 dunlearn - Resets all task parameters to default values.
    gprep - Transform shifts for one WFPC2 chip into
            parameters for mosaicing all four chips.
wfpc2_chips @ parameter set for 'avshift' and 'gprep' tasks.
      cdriz @ parameter set for the 'offsets' task.
```

5.1.2 Currently Supported Tasks

The latest developments in the dither package have focused on automating the functionality of the original separate **IRAF** tasks. The native **IRAF** tasks which are still fully supported include:

```
    drizzle - Perform linear image reconstruction using "drizzling".
       blot - Image sampling using interpolation
              (reverse "drizzling").
       tran - Transform coordinates taking into account distortion
   tranback - Transform an X,Y position from output to
              input pixel position
              using the same conventions as drizzle
      traxy - Transform an X,Y position from input to output
              pixel position using the same conventions as drizzle
     wcs2dr - Convert WCS header information into drizzle
               parameters for image registration
```

A majority of the new task development has been done by writing the new tasks in Python. These new tasks use **PyRAF** to provide an **IRAF** EPAR interface to use them as if they were native **IRAF** tasks. These Python tasks can only be run under **PyRAF** or directly under Python itself, and include:

```
multidrizzle - Automated Python task for performing
               cosmic ray rejection
```

```
   pydrizzle - Automated Python-based interface for
               the "drizzle" task
 tweakshifts - Compute residual shifts between images
     xytosky - Translate a 2-D image pixel coordinate to right
               ascension and declination, optionally applying the
               distortion coefficients for the detector.
      wtraxy - Transform an X,Y position from input to output pixel
               position using the same conventions as drizzle,
               with WCS and list support
   wtranback - Transform an X,Y position from output to input
               pixel position using the same conventions as
               drizzle - with WCS and list input support
       wblot - Image Sampling using Interpolation - reverse
               Drizzle version with WCS support
```

5.1.3 Supplemental Tasks

The last set of tasks involve operations which are generally useful, but not directly involved in image alignment or coordinate transformations. These are supporting **PyRAF** tasks which may still aid the user in preparing and analyzing their datasets:

```
     cgsky - Compute sky using `gsky' plus a correction for
             histogram truncation.
      gsky - Compute sky using the same mode algorithm used
             in task "crrej".
 imextreme - Locates the maximum and minimum pixels in an image.
       sky - Sky processing by "crrej-like" algorithm.\
        dq @ parameter set for the 'sky' task.
```

These tasks, however, have been written as native **IRAF** tasks and have been known to have problems with some datasets. Problems with any of the tasks in the **Dither** package can be reported to help@stsci.edu.

5.2 Drizzle

The **Dither** package consists of many separate tasks, each with targeted functionality designed to align observations and detect cosmic rays and other cosmetic defects. The drizzle code itself is actually a set of compiled Fortran code, which allows users to combine observations and improve the sampling of the overall dataset through the use of the drizzle algorithm. Supplemental tasks based on this drizzle code also provide the capability to perform coordinate transformations using the full drizzle algorithm, instead of going through all the effort of re-sampling images themselves. The coordinate transformation tasks include **traxy, tranback, wtraxy, wtranback**, and **tran**. The act of drizzling can encompass all of these features, or the user may simply choose to use the “drizzle” code itself to merely combine separate images. Using a scale and pixfrac of 1.0 is akin to performing shift-and-add combination, and there are science limitations where no additional benefit is gained through drizzling

that cannot be achieved through a simple median combination. For a more complete discussion on when and how it's appropriate to drizzle your dataset, please see Chapter 2 on *Introduction to Dithering*.

5.2.1 The IRAF Drizzle Task

The **drizzle** task in the **Dither** package provides the most basic application of the drizzle algorithm for re-sampling images. It relies on the user to provide all the inputs necessary for re-sampling the image; including, the distortion coefficients, the offsets, rotation or plate-scale changes, and the size of the output image itself. The full set of parameters for 'drizzle' are found in the following table (Table 5.1).

Table 5.1: Input Parameters for the **IRAF** Task Drizzle

Parameter	Default Value	Description	Format
input		Input data image as a single filename	string
outdata		output data image	string
outweig		output weight image	string
outcont		output context image	string
in_mask		input weighting mask	string
wt_scl	exptime	Weighting factor for input data image	string
outnx	800	X dimension for created images (if outdata not extant)	integer
outny	800	Y dimension for created images (if outdata not extant)	integer
kernel	square	Shape of the kernel function	string [square,point,gaussian, turbo,tophat,lanczos3]
pixfrac	1.0	drop size of each pixel, in pixel units form 0->1	real
scale	1.0	Linear size of output pixels in terms of input pixels	real
coeffs		Geometrical distortion coefficients file name	string [filename, 'header' or 'wcs']
lambda	555.0	Effective Wavelength (nm), Trauger coefficients only	real
xsh	0.0	X shift (in pixels) to be applied to input image	real
ysh	0.0	Y shift (in pixels) to be applied to input image	real
rot	0.0	Rotation of input image to be applied (degrees anti-clockwise)	real
shft_un	input	Units of shifts (input or output pixels)	string [input, output]
shft_fr	input	Frame in which shifts are applied	string [input, output]
xgeoim		X-shifts geometric distortion image	string
ygeoim		Y-shifts geometric distortion image	string
align	center	Reference point: corner or center of pixel	string [center, corner]
dr2gpar		Secondary geometric parameters	pset
expkey	exptime	Exposure time keyword in input data image header	string
in_un	counts	Units of the input image	string [cps, counts]
out_un	counts	Units of the output image	string [cps, counts]
fillval	INDEF	Value to be assigned to undefined output points	string

The task understands how to work with single images and subarray images for WFPC2, STIS, NICMOS, ACS, and WFC3 observations for which the distortion coefficients are available in the correct format. The distortion coefficients required as input for the **drizzle** task need to be in ASCII files with the coefficients providing the transformation from input pixels to the output frame in pixels. The output frame is defined by drizzle as the frame which includes all chips from the observation. This requires the coefficients specified in the drizzle coefficients files to perform all the transformations necessary to properly place each chip in the output image taking into account all differences in orientation, plate-scale and offset between the chips to reproduce the entire field-of-view on the sky for the observation. The primary source of distortion coefficients for ACS, STIS and WFC3 are IDCTAB reference tables (Section 4.2.4), and the latest calibrations for the distortion for WFPC2 are also stored using IDCTAB reference tables. These tables can be automatically converted into ASCII coefficients files by running **PyDrizzle** (Section 5.3) on the input image with the **PyDrizzle** generated ASCII files being intended as input to the **drizzle** task.

The **drizzle** task does not compute cosmic ray masks, or calculate additional offsets and rotations that might be needed in order to get the best pixel alignment. If you have data for which cosmic ray rejection needs to be performed, please consider using **MultiDrizzle** (Section 5.4).

5.2.2 The IRAF Blot Task

The **IRAF** task **blot** from the **Dither** package performs the inverse operation of drizzle; namely, it converts an undistorted image back into the original distorted image. This task conceptually runs the drizzle algorithm in reverse on the median image to recover the original input image. This task is primarily used for identifying cosmic rays in the original image. Like the original drizzle task, **blot** requires the user to explicitly provide all the transformation details as inputs. The full set of parameters for this task are in the following table (Table 5.2).

Table 5.2: Input Parameters for the **IRAF** Task Blot

Parameter	Default Value	Description	Format
input		Input image as a single filename	string
outdata		output data image	string
scale	1.0	Linear size of output pixels in terms of input pixels	real
coeffs		Geometrical distortion coefficients file name	string [filename, 'header' or 'wcs']
lambda	555.0	Effective Wavelength (nm), Trauger coefficients only	real
xsh	0.0	X shift (in pixels) to be applied to input image	real
ysh	0.0	Y shift (in pixels) to be applied to input image	real
rot	0.0	Rotation of input image to be applied (degrees anti-clockwise)	real
shft_un	input	Units of shifts (input or output pixels)	string [input, output]
shft_fr	input	Frame in which shifts are applied	string [input, output]
xgeoim		X-shifts geometric distortion image	string
ygeoim		Y-shifts geometric distortion image	string
align	center	Reference point: corner or center of pixel	string [center, corner]
outnx	800	X dimension for created images (if outdata not extant)	integer
outny	800	Y dimension for created images (if outdata not extant)	integer
interpol	poly5	Interpolant	string [nearest,linear,poly3, poly5,spline3]
sinscl	1.0	Scale for sinc interpolation kernel	real
fillval	INDEF	Value to be assigned to undefined output points	string
expkey	exptime	Exposure time keyword in input data image header	string
in_un	counts	Units of the input image	string [cps, counts]
out_un	counts	Units of the output image	string [cps, counts]
expout	input	Exposure time for output image	string

5.2.3 WCS-enabled Dither Tasks

Instead of explicitly providing all the alignment information, the WCS information for the input image can be used in conjunction with the distortion coefficients to place the image in the output frame. This can be done using WCS-enabled versions of the basic drizzle and blot tasks, as well as the coordinate transformation tasks **traxy** and **tranback**. These tasks allow the user to simply specify the output WCS information or use the WCS information of an existing drizzled image to define the output frame. The WCS keywords from the input image will then be fit to the WCS derived from the output frame to determine how to place the image in the output image without requiring the user to explicitly provide offsets, rotations and scale changes as required by the original **drizzle** task.

5.2.3.1 WCS-enabled Drizzle

The task **wdrizzle** provides the user with the ability to use the WCS information to apply the drizzle algorithm to an input image and place that corrected image in an output frame specified through the use of WCS information. The full parameter set for **wdrizzle** are in the following table (Table 5.3).

Table 5.3: Input Parameters for the **IRAF** Task Wdrizzle

Parameter	Default Value	Description	Format
input		Input data image as a single filename	string
outdata		output data image	string
outweig		output weight image	string
outcont		output context image	string
in_mask		input weighting mask	string
wt_scl	exptime	Weighting factor for input data image	string
outnx	800	X dimension for created images (if outdata not extant)	integer
outny	800	Y dimension for created images (if outdata not extant)	integer
geomode	user	Way of specifying geometry	string [wcs, user]
kernel	square	Shape of the kernel function	string [square,point,gaussian,turbo,tophat,lanczos3]
pixfrac	1.0	drop size of each pixel, in pixel units form 0->1	real
coeffs		Geometrical distortion coefficients file name	string [filename, 'header' or 'wcs']
lambda	555.0	Effective Wavelength (nm), Trauger coefficients only	real
xgeoim		X-shifts geometric distortion image	string
ygeoim		Y-shifts geometric distortion image	string
align	center	Reference point: corner or center of pixel	string [center, corner]
scale	1.0	Linear size of output pixels in terms of input pixels	real
xsh	0.0	X shift (in pixels) to be applied to input image	real
ysh	0.0	Y shift (in pixels) to be applied to input image	real
rot	0.0	Rotation of input image to be applied (degrees anti-clockwise)	real
shft_un	input	Units of shifts (input or output pixels)	string [input, output]
shft_fr	input	Frame in which shifts are applied	string [input, output]
outscl	0.1	Scale of output image, arcsecs/pixel	real

Parameter	Default Value	Description	Format
raref	0.0	RA of reference point on output image (CRVAL1, degrees)	real
decref	0.0	Dec of reference point on output image (CRVAL2, degrees)	real
xrefpix	0.0	Reference pixel Y position on output (CRPIX1)	real
yrefpix	0.0	Reference pixel Y position on output (CRPIX2)	real
orient	0.0	Orientation of output (PA of Y axis, N through E)	real
dr2gpar		Secondary geometric parameters	pset
expkey	exptime	Exposure time keyword in input data image header	string
in_un	counts	Units of the input image	string [cps, counts]
out_un	counts	Units of the output image	string [cps, counts]
fillval	INDEF	Value to be assigned to undefined output points	string

If the output image already exists and the **geomode** parameter has been set to *wcs*, then the WCS information from that image will be used in place of the WCS parameter values specified for **outscl**, **raref**, **decref**, **xrefpix**, **yrefpix**, **orient**.

5.2.4 Coordinate Transformation Tasks

Several supplemental tasks have also been developed using the core drizzle Fortran code base, primarily to provide the ability to apply the drizzle transformations to coordinates rather than to whole images. This capability allows the user to measure an object of interest in one frame and find it's exact position in the other frame making it possible to eliminate the effects of re-sampling of the PSF.

The tasks **traxy** and **tranback** require the user to provide the full set of transformation parameters as they would be given to **drizzle** or **blot** respectively. These tasks also have WCS-enabled versions; namely, **wtraxy** and **wtranback**. These WCS enabled versions accept the same inputs as **wdrizzle** or **wblot** to set up the parameters for the transformation, and will use the WCS from an existing image to compute the results.

The task **tran**, on the other hand, asks for the original input image as well as the output drizzled image as created by any of the drizzle tasks and a direction for the transformation. It then reads the drizzle parameters written to the header of the output image to set up the transformation, and will automatically use either **tran** or **tranback** depending on the direction of the transformation requested by the user.

All of these tasks support operations on either a single X,Y position or on a list of X,Y positions provided as an ASCII file with a column of X positions and a column of Y positions.

5.3 PyDrizzle

5.3.1 Introduction

PyDrizzle provides a semi-automated interface for computing the parameters necessary for running drizzle. **PyDrizzle** performs the task of determining the parameters necessary for aligning images based on the WCS information in the input image headers, as well as any supplemental alignment information provided in shift files (Section 5.5.2), and combines the images onto the same WCS. It does not identify cosmic rays, however, it has the ability to ignore pixels flagged as bad, such as pixels identified by other programs as affected by cosmic rays. If you have data for which cosmic ray rejection needs to be performed, please consider using **MultiDrizzle** (Section 5.4). This task was written using Python and may be run in either the Python or PyRAF environments, although **PyRAF** is still necessary in order to have access to the EPAR interface.

The task understands how to work with single images, sub-array images, dither associations and mosaic associations for WFPC2, WFC3, STIS, NICMOS and ACS observations. Processing dithered or mosaic data with **PyDrizzle** gets handled through the use of association tables (Section 5.5.2). These are simple binary FITS tables with a few columns to specify the input exposures and the rootname of the output exposure, along with any residual offset or rotation relative to the headers WCS information necessary to get precise alignment when drizzling the data together. The offset information is applied to the shifts **PyDrizzle** (Section 5.3) calculates which are based on the WCS information contained in the header of each image.

Another **PyRAF** task, **buildasn** (Section 5.5.2), was written as part of the **IRAF dither** package to help a user build an association table from a set of observations in a user's directory. This can then be used as input to **PyDrizzle** to combine all the specified observations in one step.

5.3.2 Parameters

PyDrizzle only aligns and combines images and still includes parameters for controlling the specifics of how this is accomplished, including exactly how to specify the output product's size, pixel scale, center and orientation. The full set of **PyDrizzle** parameters are described in the following table (Table 5.4).

Table 5.4: Input Parameters for PyDrizzle

Parameter	Default Value	Description	Format
input		ASN table or single image filename, wildcard suffix, or @list	string
output		output drizzled image	string
section		Specify the image chip to be drizzled as a fits extension	string
prodonly	yes	Only use the product from the ASN table as input	boolean [True, False]
kernel	square	Shape of the kernel function	string [square,point, gaussian,turbo, tophat,lanczos3]
units	cps	Units of the output image	string [cps, counts]
pixfrac	1.0	drop size of each pixel, in pixel units form 0->1	real
rotate	no	Specify whether the output image should be rotated	boolean [False, True]
orient	0.0	Absolute orientation on the sky of the output image y-axis	real
psize		Size of the output pixel in arcseconds	real
ra		right ascension of the output image reference pixel (override default)	real
dec		declination of the output image reference pixel (override default)	real
xsize		the size in x of the output image (override default)	integer
ysize		size in y of the output image (override default)	integer
bits_single	0	integer sum of all DQ bit values that should be used for single=yes, when each input image is drizzled separately	integer
bits_final	0	integer sum of all DQ bit values that should be used for single=no, when the final image is created	integer
wt_scl	exptime	weighting factor for the input image	string [exptime,expsq]
in_units	cps	the units of the input image	string [cps,counts] (in either dn or e-)
idckey	idctab	keyword to use for selecting the reference file which contains the distortion coefficients for the data	string
clean	yes	remove temporary files created by PyDrizzle	boolean [True,False]
save	no	specify whether to keep the original drizzle output files	boolean [False,True]
build	yes	combine the separate output images into a single multi-extension fits file	boolean [True,False]

5.3.3 Basic Examples

5.3.3.1 Running in Pure Python

The following serves as a basic example of how to import and use **PyDrizzle** to process a dataset. This can be done in either a Python or **PyRAF** session, but this example is explicit for Python syntax. For an example in the **PyRAF** environment see the section on *Real-World Examples* Chapter 6 or simply type "epar pydrizzle" from the **PyRAF** command line and fill in the available parameters. If you have not created an association table yet, see the section on the **buildasn** (Section 5.5.2) function. For more information on the Python environment see the Python Web page:

http://www.python.org/

```
>import pydrizzle

(then you can create the PyDrizzle object by calling the following)

>p=pydrizzle.PyDrizzle("my_asn.fits", bits_single=0,
                       bits_final=0, kernel="square",
                       pixfrac=1.)
```

Table 5.5: Input Parameters for PyDrizzle

object/parameter	Description
p	the object that is returned from the PyDrizzle function, it contains all the information necessary to start drizzling the data in the ASN table
"my_asn.fits"	the name of the ASN table that tells the function which images to use and how to apply offsets
bits_single=0	ignore all pixels (set to 0) which have bit values greater than 0 in the mask
bits_final=0	ignore all pixels (set to 0) which have bit values greater than 0 in the mask
kernel="square"	the type of kernel to be applied to the input pixels
pixfrac=1.	the fractional drop size for each pixel, no change here to original pixels, there's 1 input pixel in 1 output pixel

You can now run PyDrizzle on the data by issuing the command:

```
>p.run()
```

Or, you can view and edit any of the objects values before executing the pydrizzle command, like the WCS information:

```
>print p.observation.members[0].members[0].geometry.wcs
WCS Keywords for j8bt06nyq_flt.fits[sci,1]:
CD_11  CD_12: -7.3535609783668304e-06  1.85500817458759e-06
CD_21  CD_22: 2.8810562653901852e-06  6.6471909533171463e-06
CRVAL       : 6.0271483333329998  -72.083511111110013
CRPIX       : 512.0  512.0
NAXIS       : 1024  1024
Plate Scale : 0.0284321026605
ORIENTAT    : 15.5926361388
CTYPE       : 'RA---TAN'  'DEC--TAN'
PA Telescope: 15.79167
```

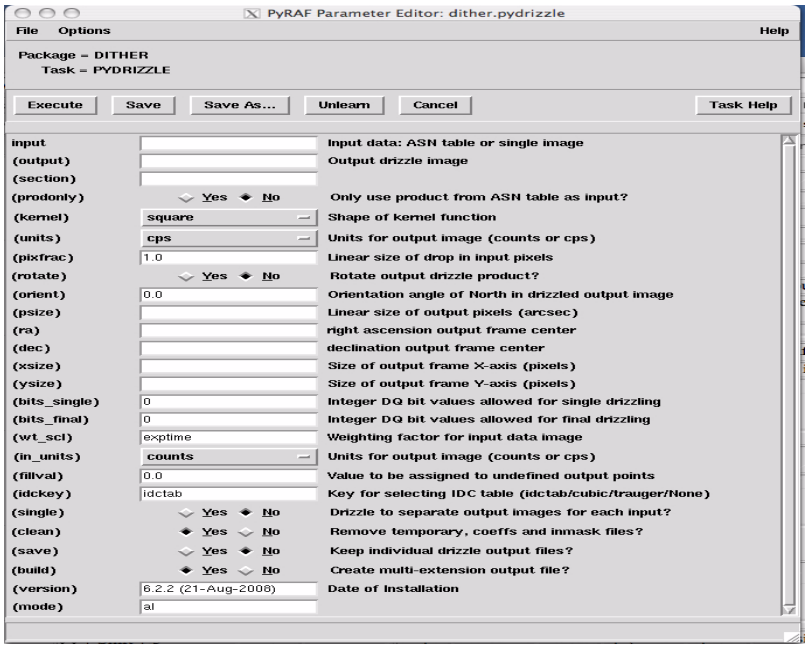
5.3.3.2 Running in the PyRAF Environment

If you prefer using the **PyRAF** environment, all you need to do is open up the task parameters using the EPAR interface. First, make sure all the necessary packages have been loaded by typing the following:

```
stsdas
dither
```

Now you can open the epar interface by typing “epar pydrizzle”. You should see a window which looks similar to Figure 5.1.

Figure 5.1: Snapshot of the PyDrizzle epar Window



Snapshot of the PyDrizzle epar window.

From here, you can enter the information necessary to drizzle your input images to a common frame.

5.3.4 Output Product File Format

The output from **PyDrizzle** is a single multi-extension FITS file with the suffix '_drz.fits'. The first extension contains the science (SCI) image which is corrected for distortion and dither-combined (or mosaiced), if applicable. The drizzled SCI image is typically in units of electrons per second, which is the default for ACS images. The user can choose to have the output in either electrons, electrons per second, data numbers (DN) or DN per second, some of these options retain the natural units of the instrument. For all options, the units chosen by the user are carried through the entire calculation. All pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image which is both photometrically and atrometricaly accurate for point and extended sources. The dimensions of the output image will be computed on-the-fly by **PyDrizzle** and the default output plate scale will be read from the IDCTAB. These parameters, however, may be chosen by the user during reprocessing to best suit the actual data.

The second extension of the output image contains the weight (WHT) image. It gives the relative weight of the output pixels and can be considered an effective exposure time map. The data quality map created by the instrument calibration pipelines is used by **PyDrizzle** in creating the weight image.

The third extension of the **PyDrizzle** output image contains the context (CTX) image which encodes information about which input image contributes to a specific output pixel. This is done using a bitmask for each output pixel, where 'bit set' means that the image, in the order it was combined, contributed with non-zero weight to that output pixel. The context image starts as a single 32-bit integer image but is extended to a cube with additional 32-bit deep planes as required to handle all the input images.

NICMOS users should note that this may seem further complicated by the nature of NICMOS data. Since the final science level NICMOS images that come out of **calnica** are really mathematical fits to the ramp of observations, the true exposure time varies across the calibrated science image. The default output from **calnica** is to have UNITCORR set to perform, which divides the pixel counts by the full exposure time for that pixel, producing an output image in units of data numbers/second. NICMOS _cal.fits images contain a TIME extension which details the true exposure level of the image, pixel-by-pixel, after the individual reads have been combined into the ramp image and after cosmic ray rejection, which includes read rejection, has been done on the image. Since the calibrated image represents the actual countrate the pixel should have reported in the absence of defects in any of the individual reads, multiplying this final image by the full exposure time of the dataset (using the EXPTIME keyword value) is an acceptable practice to convert the science image back into relative counts. The [TIME] extension in the output _cal file should only be used as a processing reference by the user.

5.4 MultiDrizzle

5.4.1 Introduction

Cosmic ray rejection using the separate tasks in **IRAF**'s dither package required the interactive use of many separate tasks; a complicated and often error-prone procedure. In addition, the geometric distortion must be removed from each input exposure in order to calculate precise shifts for alignment of the images well enough to recognize the presence of cosmic rays in each exposure. MultiDrizzle automates this entire process into a single operation based on the basic drizzle algorithm for removing the distortion and aligning the images. It then provides control over the individual steps through a separate set of parameters for each step in the process while taking into account many of the unique characteristics of the different HST cameras. As a result, a single set of dithered exposures can be combined into a single mosaic while removing the distortion and cosmic rays simply by running this one task.

5.4.2 Software Requirements

MultiDrizzle has been designed as a Python package with it's own interface and can be run from the Python environment in conjunction with **STSDAS** 3.2 or later. However, it can also be loaded into **STSDAS** under **PyRAF** to enable use of the graphical EPAR interface that comes with the **PyRAF** command interface. These packages have been designed to work together in an almost seamless manner without requiring any manual setting of the environment by the user. MultiDrizzle requires at least Python 2.4, Numpy v1.0, and F2PY to be installed for operation with the Python interfaces. In addition, **STSDAS** 3.8 and **PyRAF** 1.1 are required for use of the GUI EPAR interface.

5.4.3 Data and Header Specifics

This is a detailed example for ACS data, but will be generally applicable for any instrument. For a specific, more detailed example of the instrument you are working with please see *Real-World Example*s Chapter 6.

All ACS data calibrated using the standard HST calibration pipeline has been processed by MultiDrizzle to remove geometric distortion and cosmic rays when associated images are combined. Data taken using a standard dither pattern, using CR-SPLIT on a long exposure or using REPEAT-OBS, as specified in the observing proposal will produce associated datasets that will be automatically combined into a dither product by MultiDrizzle. There will be times, though, when the results are not sufficient for scientific analysis and the images require reprocessing off-line using MultiDrizzle. This example illustrates how to take observations associated by the HST calibration pipeline and reprocess them with MultiDrizzle.

Table 5.6: Association table for dithered observations. This example will be based on observations taken using a 2-point Dither-Line pattern with CR-SPLIT=2 at each pointing using the WFC detector of the ACS.

MEMNAME	MEMTYPE	MEMPRSNT
J8CW03A1Q	EXP-CR1	yes
J8CW03A2Q	EXP-CR1	yes
J8CW03FFQ	EXP-CR2	yes
J8CW03FJQ	EXP-CR2	yes
J8CW030X0	PROD-DTH	no
J8CW03021	PROD-CR1	yes
J8CW03031	PROD-CR2	yes

It is assumed that this association has already been processed using **calacs**. The products created by **calacs** include both cosmic ray cleaned, calibrated products with 'crj.fits' suffixes and fully calibrated input images with 'flt.fits' suffixes. This will result in having at least the following files available in the current directory:

Table 5.7: Files That Need to be Available to run MultiDrizzle

J8cw030x0_asn.fits
J8CW03A1Q_flt.fits
J8CW03A2Q_flt.fits
J8CW03FFQ_flt.fits
J8CW03FJQ_flt.fits

5.4.3.1 Reference Files

The default files which were used to process data retrieved from the HST archive are referenced in the header using an environment variable which points to the directory where they are stored. For ACS this variable is called "jref". If you retrieved the best reference files along with your data, make sure that this points to the directory in which they were saved. If you have used specialized reference files make sure that the full path to their location is also referenced correctly in the data headers. A quick way to check the current value of the variable (in **IRAF** or **PyRAF**) is to use the "show" command:

```
>show jref
```

If the variable is incorrect, it can be set using the standard **IRAF** command "set", where the directory shown in this example would be replaced by the correct one for your local system. It might, for example, be located at:

```
>set jref = /grp/hst/cdbs/jref
```

You should also set jref in your appropriate operating system environment setup file (such as .setenv on Mac X11, Unix and Linux)

```
>setenv jref /grp/hst/cdbs/jref
```

Two important file are the distortion coefficients reference file, IDCTAB (Section 4.2.4), and MultiDrizzle parameter table, MDRIZTAB (Section 5.4.7), which are referenced in the primary headers of each dataset. The MultiDrizzle parameter table provides all the default settings used by the pipeline to run MultiDrizzle on any given single image or association. For these images, using the **IRAF** syntax, we find that IDCTAB and MDRIZTAB are:

```
>hedit j8cw03a1q_flt.fits[0] idctab,mdriztab .
>j8cw03a1q_flt.fits[0],IDCTAB = jref$wfc_smov_idc.fits
>j8cw03a1q_flt.fits[0],MDRIZTAB = jref$wfc_smov_mdz.fits
```

5.4.4 Running Under PyRAF

MultiDrizzle has been incorporated into **STSDAS** not only as a Python task in its own right, but also as an **IRAF** task with the usual parameter driven interface. The Python syntax can also be used from **PyRAF**, since **PyRAF** provides full Python functionality as well as access to **IRAF** tasks.

PyRAF can usually be started by simply running the command:

```
>your_computer> pyraf
```

Once **PyRAF** has started up, load the **STSDAS**, then dither packages:

```
>stsdas
>dither
```

At this point, the **PyDrizzle** code has been loaded into **PyRAF** and is available for use. The **IRAF** EPAR interface can be used to review and set the available parameters which guide MultiDrizzle's processing routines.

5.4.5 Running under Python

MultiDrizzle can also be used directly from Python without loading **IRAF**, **STSDAS** or **PyRAF**. MultiDrizzle has been written as a Python package and can be treated just like any other Python package. Therefore, to use MultiDrizzle directly from Python, go to the directory with the data and start up Python:

```
your_computer> python
```

In order to use MultiDrizzle with the Python interface the code must be installed in a directory accessible by Python. If that directory is not already included in the PYTHONPATH or default Python system path, you can point to it inside Python using:

```
>>> sys.path.insert(1,'/directory/with/Multidrizzle/code')
```

MultiDrizzle can then be loaded using the usual import mechanism in Python:

```
>>> import multidrizzle
```

At this point, MultiDrizzle can be run on the calibrated images in the local directory. Interactive help can be obtained for MultiDrizzle using the built-in help mechanism. This will provide basic information on the methods available for use and their syntax as well as the most basic set of commands outlined in this example for running MultiDrizzle.

```
>>> multidrizzle.help()
```

5.4.6 Running MultiDrizzle

The Python syntax for running MultiDrizzle does not vary that much from the **PyRAF** syntax. Unlike the **PyRAF** syntax, it provides the user with the ability to examine the inputs and computed parameters before spending the majority of time actually doing the image combination. **MultiDrizzle** creates an instance of a MultiDrizzle object which contains all the methods necessary for editing the input parameters, computing the image combination parameters, then performing the actual cosmic ray detection and image combination. All of these functions get wrapped into one call through the **PyRAF** interface, making for simpler operation at the cost of the ability to inspect the object and its contents.

The processing starts with the creation of a MultiDrizzle object, which for the sake of this example will simply be called ‘md’:

```
>>>md = multidrizzle.Multidrizzle(input='j8cw030x0_asn.fits',
    mdriztab=yes, editpars=False,
    **input_dict)
```

The parameter ‘input_dict’ serves as the means for the user to specify any of the remaining parameters from the parameter table which should have non-default values. Setting ‘editpars’ to True here will cause the Python GUI parameter editor to start automatically. You may also choose to use the separate ‘editpars()’ method.

The specification of the MDRIZTAB parameter in the initialization demonstrates how any of the input parameters whose values need to be changed from the default value can be provided at the start. If many parameters need to be edited, the Python GUI interface can be started, simply closing this GUI saves the values in the MultiDrizzle object for use.

```
>>>md.editpars()
```

This editor allows the user to examine all the inputs for MultiDrizzle and set them as necessary to work best with the input data. See the parameter table (Section 5.4.7) for the full available set of MultiDrizzle parameters. At this point the MultiDrizzle object contains all the input parameter values necessary for the processing. However,

the computation of the parameters necessary for doing the image combination and cosmic ray detection still need to be performed using the build method:

```
>>>md.build()
```

The MultiDrizzle object (md) now contains everything necessary for the processing. It can still be inspected using simple python commands, but no further method operations are necessary to complete the setup process. Alternatively, a help method can provide a reminder of the available methods for any MultiDrizzle object already created using:

```
>>>md.help()
```

The full MultiDrizzle process can now be started:

```
>>>md.run(static=None, skysub=None, driz_separate=None,
        median=None, blot=None, driz_cr=None,
        driz_combine=None)
```

This method allows the user one last chance to specify what processing steps should be performed. If no parameters are specified, all processing steps turned on during the instantiation of the MultiDrizzle object will be performed. Generally, though, the user simply wants to run this method without any specified parameters, such as:

```
>>>md.run()
```

This represents the last step in processing images with MultiDrizzle, with the final product being a registered, cosmic ray cleaned, distortion-free, photometrically uniform image.

5.4.6.1 Session Summary

Very few commands are actually needed in order to process images or sets of images using MultiDrizzle. The total session described here simply boils down to:

```
>your_computer> cd /directory/with/data/
>your_computer> pyraf
>stsdas
>dither
>multidrizzle j8cw030x0_asn.fits mdriztab=yes mode=h
```

The Python syntax does not require many more commands:

```
>import multidrizzle
>md = multidrizzle.Multidrizzle('j8cw030x0_asn.fits',mdriztab=yes)
>md.editpars()
>md.build()
>md.run()
```

Both ways of running MultiDrizzle will produce the exact same output products if the same input files are supplied.

5.4.6.2 Intermediate Products

There are many intermediate steps during the full course of "drizzling" your input images into a mosaic. Each of these steps has particular output which informs the next stage of the proceedure. The intermediate products are all those files which get created during the process but are not necessarily considered final science products. These may include files with the extensions:

- _single_sci.fits: The individually drizzled, distortion corrected versions of each input image
- _single_mask1.fits: These are the first version of the bad pixel files that are generated
- _final_mask1.fits: These are the final masks to be used for each science image
- _bl.fits: The blotted images which were created by undistorting and shifting the part of the median image which corresponds to the FOV of the original science exposure
- _cr.fits: cosmic ray masks which are created as part of the cosmic ray detection step, accomplished in the driz_cr subroutine
- _drw.fits: pixel weighting images which are output whenever the drizzle step is run
- _med.fits: most likely these are median combined singly drizzled images
- _coeffs2.dat: this is a text file which contains information extracted from the IDC table about the distortion in the instrument. This will be replaced by the SIP header information which will also be added to the headers.

MultiDrizzle allows an option (clean) to turn on and off the saving of files which are created during intermediate steps. While it is good to keep intermediate files so that you can assess the full performance of the tasks, you must keep in mind the size of your input images and the disk capacity of the machine on which you're running, and set this parameter accordingly.

5.4.6.3 Final Products

The output from **PyDrizzle** is a single multi-extension FITS file with the suffix "_drz.fits". The first extension contains the science (SCI) image which is corrected for distortion and dither-combined (or mosaiced), if applicable. The drizzled SCI image is typically in units of electrons per second, which is the default for ACS images. The user can choose to have the output in either electrons, electrons per second, data numbers (DN) or DN per second, some of these options retain the natural units of the instrument. All pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image which is both photometrically and astrometricaly accurate for point and extended sources. The dimensions of the output image will be computed on-the-fly by **PyDrizzle** and the default output plate scale will be read from the IDCTAB. These parameters, however, may be chosen by the user during reprocessing to best suit the actual data and the users needs.

The second extension of the output image contains the weight (WHT) image. It gives the relative weight of the output pixels and can be considered an effective exposure time map. The data quality map created by the instrument calibration pipelines is used by **PyDrizzle** in creating the weight image.

The third extension of the **PyDrizzle** output image contains the context (CTX) image which encodes information about which input image contributes to a specific output pixel. This is done using a bitmask for each output pixel, where 'bit set' means that the image, in the order it was combined, contributed with non-zero weight to that output pixel. The context image starts as a single 32-bit integer image but is extended as a cube with additional 32-bit deep planes as required to handle all the input images.

NNICMOS users should note that this may seem further complicated by the nature of NICMOS data. Since the final science level NICMOS images that come out of **calnica** are really mathematical fits to the ramp of observations, the true exposure time varies across the calibrated science image. The default output from **calnica** is to have UNITCORR set to perform, which divides the pixel counts by the full exposure time for that pixel, producing an output image in units of data numbers/second. NICMOS _cal.fits images contain a TIME extension which details the true exposure level of the image, pixel-by-pixel, after the individual reads have been combined into the ramp image and after cosmic ray rejection, which includes read rejection, has been done on the image. Since the calibrated image represents the actual countrate the pixel should have reported in the absence of defects in any of the individual reads, multiplying this final image by the full exposure time of the dataset (using the EXPTIME keyword value) is an acceptable practice to convert the science image back into relative counts. The [TIME] extension in the output _cal file should only be used as a processing reference by the user.

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

ACS and WFC3 users should note that the calibrated FLT files contain additional bits in the DQ mask that identify pixels flagged as cosmic rays by MultiDrizzle. The default value for these bits is 4096, and if MultiDrizzle is re-run on the FLT files offline it will be able to use these flags and proceed directly to the final drizzle step without the need to re-do cosmic ray masking, unless that is required. Alternatively, if the cosmic ray rejection needs to be improved, then the pipeline CRs can be ignored/reset by adding 4096 to the "bits" parameter value when running MultiDrizzle.

5.4.7 MultiDrizzle Parameters

MultiDrizzle processing starts with a list of input images, then applies a number of operations on them to produce the final cosmic ray cleaned image; specifically:

1. Initialization (Section 5.4.8)
2. Bad pixel (Section 5.4.9) (static) mask creation
3. Sky Subtraction (Section 5.5.4)
4. Drizzle onto separate (Section 5.4.10), registered output images
5. Combine the separate drizzled images (Section 5.4.11) into a median
6. "blot" (Section 5.4.12), or transform back the median to each input image
7. Compare the median with original images to make cosmic ray masks (Section 5.4.13)
8. Drizzle all the images onto a final image (Section 5.4.14), using the cosmic ray masks

MultiDrizzle relies on a large number of parameters for specifying what steps get performed and for controlling the algorithm used by each step. The complete list of parameters and their default values are given in the parameter table (Section 5.4.7). It is generally recommended to try running the script first with the default parameters, which should allow the task to process nearly any set of images for an initial review. The script can then be re-run, or restarted at a particular step, with modified parameters if this is necessary. MultiDrizzle is not designed to be restarted in the middle of processing, so when computing improved cosmic ray masks, the user must turn on all steps prior to 'driz_cr'. If this is not done, the header keyword MDRIZSKY will be set to zero, and the software will flag any reasonably bright objects in the image. The result is a cosmic ray mask that looks like a map of the image, where all pixels containing sources have been flagged. When MultiDrizzle is run by the archive instrument pipelines it references the MDRIZTAB file in the header of the dataset. MDRIZTAB is a fits table file which contains carefully chosen input parameters for MultiDrizzle which are optimized for the widest range of science cases. These are the default parameters which are used in the instrument pipelines for data retrieved from the HST archive, and may be different for each instrument. Users are encouraged to examine whether these parameters are appropriate for their particular datasets and recombine their science data as necessary.

Table 5.8: Parameters for MultiDrizzle

Parameter	Default Value	Description	Format
Initialization (Section 5.4.8)			
input	*flt.fits	Input files: filename, wildcard suffix, or @list	string [*flt.fits]
output		Rootname for output drizzled products	string
mdriztab	no	Use table with MultiDrizzle parameters?	boolean [True,False]
refimage		Name of image which specifies the output WCS	string
runfile		File for logging the final drizzle commands	string
workinplace	no	Work on input files in place? (*NOT RECOMMENDED*)	boolean [False,True]
updatewcs	yes	Update the WCS keywords?	boolean [False,True]
coeffs	header	Use header-based distortion coefficients?	string [header,cubic,trauger,none]
context	no	Create context image during final drizzle?	boolean [False, True]
clean	no	Remove temporary files?	boolean [True,False]
group		Specification of extension or group to process	string
ra		right ascension of output frame center	float
dec		declination of output frame center	float
build	yes	Create multi-extension output file?	boolean [True,False]
shiftfile		Name of file containing shifts	string
staticfile		Name of the optional input static bad-pixel mask	string
Static Mask Creation (Section 5.4.9)			
static	yes	Create static bad-pixel mask from the data?	boolean [True,False]
static_sig	4.0	Sigma*rms below mode to clip for static mask	float

Parameter	Default Value	Description	Format
Sky Subtraction (Section 5.5.4)			
skysub	yes	Perform sky subtraction?	boolean [True,False]
skywidth	0.1	Bin width for sampling sky statistics (in sigma)	float
skystat	median	Sky correction statistics parameter	string [median,mode,mean]
skylower	INDEF	Lower limit of usable data for sky (always in electrons)	float
skyupper	INDEF	Upper limit of usable data for sky (always in electrons)	float
skyclip	5	Number of clipping iterations	integer
skylsigma	4	Lower side clipping factor (in sigma)	float
skyusigma	4	Upper side clipping factor (in sigma)	float
skyuser		KEYWORD indicating a sky subtraction value if done by user.	string
Create Separate Drizzled Images (Section 5.4.10)			
driz_separate	yes	Drizzle onto separate output images?	boolean [False,True]
driz_sep_outnx		Size of separate output frame's X-axis (pixels)	float
driz_sep_outny		Size of separate output frame's Y-axis (pixels)	float
driz_sep_kernel	turbo	Shape of kernel function	string [turbo,square,point,gaussiantophat ,lanczos3]
driz_sep_wt_scl	exptime	Weighting factor for input data image	float
driz_sep_scale	INDEF	Absolute size of output pixels in arcsec/pixel	float
driz_sep_pixfrac	1.0	Linear size of drop in input pixels	float
driz_sep_rot	INDEF	Orientation of final image's Y-axis w.r.t. North (in degrees)	float
driz_sep_fillval	INDEF	Value to be assigned to undefined output points	float
driz_sep_bits	0	Integer mask bit values considered good	integer

Parameter	Default Value	Description	Format
Create Median Image (Section 5.4.11)			
median	yes	Create a median image?	boolean [True,False]
median_newmasks	yes	Create new masks when doing the median?	boolean [True,False]
combine_maskpt	0.7	Percentage of weight image value below which it is flagged	float
combine_type	median	Type of combine operation	string [minmed,minimum,median,sum]
combine_nsigma	4 3	Significance for accepting minimum instead of median	string
combine_nlow	0	*minmax*: Number of low pixels to reject	integer
combine_nhigh	1	*minmax*: Number of high pixels to reject	integer
combine_lthresh	INDEF	Lower threshold for clipping input pixel values	float
combine_hthresh	INDEF	Upper threshold for clipping input pixel values	float
combine_grow	1	Radius (pixels) for neighbor rejection	float
Blot Median Image (Section 5.4.12)			
blot	yes	Blot the median back to the input frame?	boolean [True,False]
blot_interp	poly5	Interpolation function to use	string [nearest,linea,rpoly3,poly5, sinc]
blot_sinscl	1	Scale for sinc interpolation kernel	float
Remove Cosmic Rays (Section 5.4.13)			
driz_cr	yes	Perform CR rejection with deriv and driz_cr?	boolean [True,False]
driz_cr_corr	no	Create CR cleaned _cor file and a _crmask file?	boolean [True,False]
driz_cr_snr	3.5 3.0	SNR parameter for cosmic rays	string
driz_cr_grow	1	pixel grow parameter for cosmic rays	integer
driz_cr_cte_grow	0	driz_cr: length of CTE tail to mask in final product	integer
driz_cr_scale	1.2 0.7	*driz_cr:* scale parameter	string

Parameter	Default Value	Description	Format
Create Final Cleaned, Combined Product (Section 5.4.14)			
driz_combine	yes	Perform final drizzle image combination?	boolean [True,False]
final_wht_type	EXP	type of weighting for final drizzle	string [EXP,IVM,ERR]
final_outnx		Size of FINAL output frame X-axis (pixels)	float
final_outny		Size of FINAL output frame Y-axis (pixels)	float
final_kernel	square	Shape of kernel function	string [square,top,gaussian,turbo,tophat,lanczos3]
final_wt_scl	exptime	weighting factor for input data image	float
final_scale	INDEF	Absolute size of output pixels in arcsec/pixel	float
final_pixfrac	1.0	Linear size of drop in input pixels	float
final_rot	0.0	Orientation of final image's Y-axis w.r.t. North (in degrees)	float
final_fillval	INDEF	Value to be assigned to undefined output points	float
final_bits	0	Integer mask bit values considered good, bit values greater than 0 will include 0	integer
final_units	cps	Units for final drizzled image [counts,cps]	string
Override Instrument Specific Parameters (Section 5.4.15)			
gain		Detector gain in electrons per count	float
gnkeyword		Detector gain keyword in header	string
rdnoise		Detector read noise in electrons	float
rnkeyword		Detector read noise keyword in header	string
exptime		Exposure time	float
expkeyword		Exposure time keyword in header	string
crbit		Bit value for CR identification in DQ array	integer

5.4.8 Initialization

5.4.8.1 General Explanation

MultiDrizzle starts by determining what files are being specified as inputs. These files then get converted as necessary to a usable input format with appropriate units. This conversion includes making copies of each input file, if 'workinplace' is set to 'False', so that the original input SCI arrays are not altered and can be used for successive runs. A set of drizzle parameter values needed to combine the images into a final output image then get computed using **PyDrizzle**.

5.4.8.2 Parameter Details

Input: This controls all inputs for the entire **MultiDrizzle** task. In addition, it requires access to:

- The name or names of the input files to be processed which can be provided in any of several forms; namely,
    - filename of a single image
    - filename of an association (ASN) table
    - wild-card specification for files in directory
    - comma-separated list of filenames
    - '@file' filelist containing list of desired input filenames. The file list needs to be provided as an ASCII text file containing a list of filenames for all input images with one filename on each line of the file. If inverse variance maps (IVM maps) have also been created by the user and are to be used (by specifying 'IVM' to the parameter final_wht_type (Section 5.4.8.2), then these are simply provided as a second column in the filelist, with each IVM filename listed on the same line as a second entry, after its corresponding exposure filename. If the user specifies 'IVM' for the 'final_wht_type' but does not provide the names of IVM files, MultiDrizzle will automatically generate the IVM files itself for each input exposure.
- the MDRIZTAB (Section 5.4.7) reference table which specifies default parameters, contained in the input image headers
- the IDCTAB (Section 4.2.4) reference table specified in the input image headers
- any specified reference image as named in the 'refimage' parameter

Output: The rootname for the output drizzled products. This step can result in the creation of several files, including:

- copies of each input image as a FITS image, if workinplace=yes and/or input images are in GEIS format

- mask files and coeffs files created by **PyDrizzle** for use by **drizzle**

If an association file has been given as input, this name will be used instead of the product name specified in the ASN file. Similarly, if a single exposure is provided, this rootname will be used for the output product instead of relying on input rootname. If no value is provided when a filelist or wild-card specification is given as input, then a rootname of 'final' will be used.

mdriztab: Specifies whether or not to use an MDRIZTAB reference table to specify the remaining MultiDrizzle parameter settings. If 'True', the values in the table will override the settings for the remainder of the parameters.

refimage: Optional "reference image" that can be provided, in which case MultiDrizzle will create a final product with the same WCS. This reference image should be a simple FITS file (single-group, no multiple extensions), and should have been already drizzled so that all of its distortion has been removed, and its WCS is completely rectified.

runfile: This log file will contain the **IRAF** CL commands necessary for performing the final combination manually using the **drizzle** task directly.

workinplace: This parameter specifies whether to perform all processing on the original, including sky subtraction and update of the DQ array. If set to 'True', then no copy of the input will be created for processing, and the original input will be modified directly by MultiDrizzle.

updatewcs: This parameter specifies whether the WCS keywords are to be updated by running makewcs on the input data, or left alone. The update performed by makewcs not only recomputes the WCS based on the currently used IDCTAB, but also populates the header with the SIP coefficients. Also, for ACS/WFC images, the time-dependence correction will also be applied to the WCS and SIP keywords. This parameter should usually be set to 'yes', unless the WCS keywords have been carefully set by some other method, and need to be passed through to drizzle 'as is'.

coeffs: The source of the distortion coefficients gets specified using:

- header: use the 'header' for determining what distortion coefficients to use.
- cubic: use 'cubic' solutions originally provided with 'drizzle'
- trauger: use 'trauger' solutions originally provided with 'drizzle'
- none: Do not apply any distortion correction to the images

Alternatively, an arbitrary distortion coefficients file can be specified (including optionally the full pathname). This distortion file is used in computing the WCS of the header. NOTE: When the 'header' option has been selected, if the IDCTAB file is not found on disk, then the task will exit for all instruments except for WFPC2 data, which will default to the older "Trauger" model.

context: This parameter specifies whether or not to create a context image during the final drizzle combination. The context image contains the information on what image(s) contributed to each pixel encoded as a bit-mask. More information on context images can be obtained from the ACS Data Handbook.

clean: The temporary files created by **MultiDrizzle** can be automatically removed by setting this parameter to 'True'. The affected files would include the coefficients files and static mask files created by **PyDrizzle**, along with other intermediate files created by MultiDrizzle. It is often useful to retain the intermediate files and examine them when first learning how to run MultiDrizzle. But when running it routinely, or on a small disk drive, these files can be removed to save space.

group: A single FITS extension or group can be drizzled by setting this parameter. If an extension is provided, then only that chip will be drizzled onto the output frame. Either a FITS extension number or GEIS group number (such as '1'), or a FITS extension name (such as 'sci,1') may be specified.

ra: Right ascension (in decimal degrees) of the center of the output image. If this is not specified, the code will calculate the center automatically based on the distribution of image dither positions.

dec: Declination (in decimal degrees) of the center of the output image. If this is not specified, the code will calculate the center automatically based on the distribution of image dither positions.

build: MultiDrizzle will combine the separate 'drizzle' output files into a single multi-extension format FITS file when this parameter gets set to 'True'. This combined output file will contain a SCI (science), a WHT (weight), and a CTX (context) extension. If set to 'False', each extension would remain as a separate simple FITS file on its own.

shiftfile: Name of optional input file containing the shifts to be applied to the input images to improve their registration.These shifts will be added to those calculated automatically from the image headers. The file should be organized as follows:

```
# units: pixels
# frame: input
# form: delta
rootname xshift yshift rotation
```

staticfile: Name of the optical input static bad-pixel mask. The input data are multiplied by the contents of this static mask file.

gain: Value used to override instrument specific default gain values. The value is assumed to be in units of electrons/count. This parameter should not be populated if the gain keyword parameter is in use.

gainkeyword: Keyword used to specify a value to be used to override instrument specific default gain values. The value is assumed to be in units of electrons/count. This parameter should not be populated if the *gain* parameter is in use.

rdnoise: Value used to override instrument specific default readnoise values. The value is assumed to be in units of electrons. This parameter should not be populated if the *rnkeyword* parameter is in use.

rnkeyword: Keyword used to specify a value to be used to override instrument specific default readnoise values. The value is assumed to be in units of electrons. This parameter should not be populated if the *rdnoise* parameter is in use.

exptime: Value used to override default exposure time image header values. The value is assumed to be in units of seconds. This parameter should not be populated if the *expkeyword* parameter is in use.

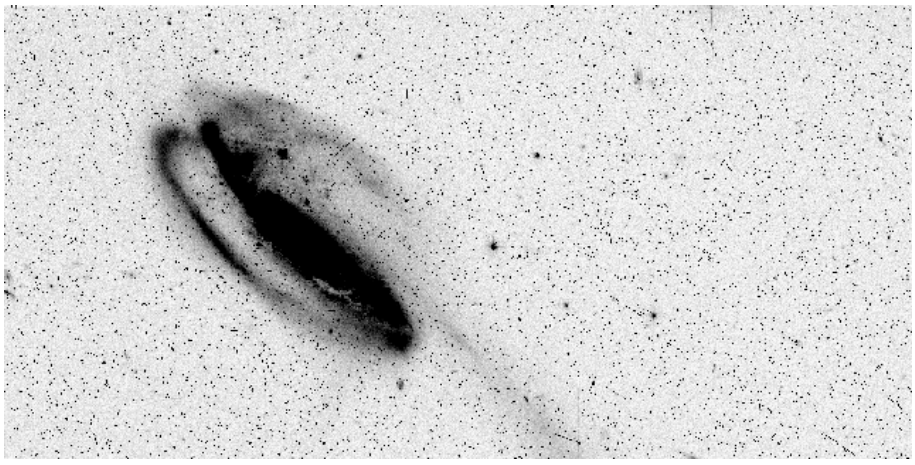
expkeyword: Keyword used to specify a value to be used to override default exposure time image header values. The value is assumed to be in units of seconds. This parameter should not be populated if the *exptime* parameter is in use.

crbit: Integer used to override instrument specific cosmic ray bit values. This value is used by MultiDrizzle to update data quality arrays when cosmic rays or other image defects are identified as "bad" in the DRIZ_CR step. To prevent the image's data quality array from being updated set the *crbit* value to 0.

5.4.8.3 Basic Example

An example of an input image that you would like to drizzle is in Figure 5.2.

Figure 5.2: Single Extension from an ACS FLT File



A single extension from a calibrated ACS FLT file, j8cw54orq_flt.fits, which still contains numerous cosmic rays, hot pixels, and other artifacts.

5.4.9 Static Mask Creation

5.4.9.1 General Explanation

If the static flag is set to "yes", a static mask will be created for all input images. The mask contains pixels which fall more than 'static_sig' *RMS below the mode for a given chip or extension. Those severely negative or low pixels might result from over subtraction of bad pixels in the dark image, or high sky levels, during calibration. For example, each ACS WFC image contains a separate image for each of 2 CCDs and separate mask will be generated for each chip.

The final static mask for each chip contains all the bad pixels that meet this criteria from all the input images along with any bad pixels which satisfied the final_bits value specified by the user and found in the images DQ mask. This static mask could also be combined with a user supplied static mask specified in the 'staticfile' parameter.

Users should consider the details of their science image and decide whether creating this mask is appropriate for the resulting science. For example, if your field is very crowded, contains mostly nebulous or extended objects, the statistics could be heavily skewed and the mask could end up containing sources.

Input: Aside from the input parameters, this step requires opening each input image to access the science (SCI) extensions for generating the static masks.

Output: The generated static masks exist strictly in memory as numarray objects that get applied during the single drizzle step. They also get used to update the input mask files for the final image combination.

5.4.9.2 Parameter Details

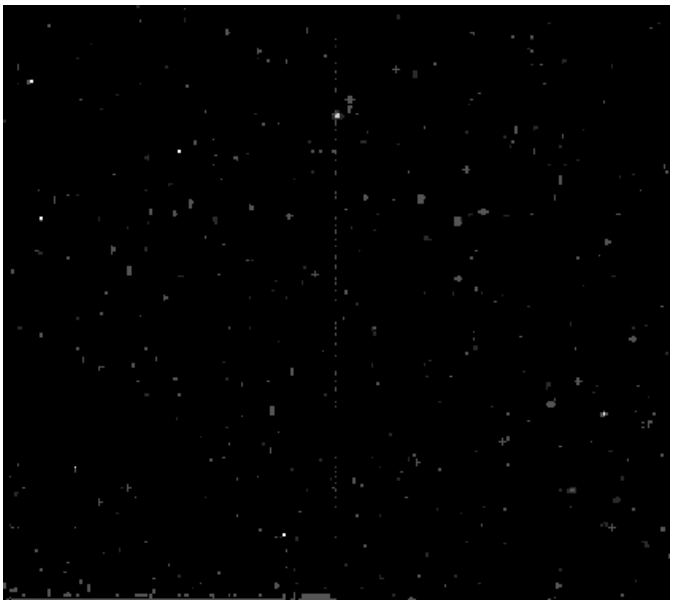
static: This parameter specifies whether to create a static bad-pixel mask from the data or not. This mask flags all pixels that deviate by more than 'static_sig' sigma below the image median.

static_sig: Number of sigma below the RMS to use as the clipping limit for creating the static mask, defaulted to 4.0

5.4.9.3 Basic Examples

An example of what a bad pixel mask might look like. (Figure 5.3) is a badpixel mask from a NICMOS camera 3 image where the differing gray levels signify different bits which have been set in the Data Quality (DQ) extension of the science image with which it is associated.

Figure 5.3: An Example Bad Pixel Mask Taken from a NICMOS Image



Bad pixel mask from NICMOS camera 3. There are multiple shades of grey because the DQ image contains the logical AND of all pixel processing flags generated during **calnica**.

5.4.10 Create Separate Drizzled Images

5.4.10.1 General Explanation

Each input image gets drizzled onto a separate copy of the output frame. When stacked, these copies would correspond to the final combined product. As separate images, they allow for treatment of each input image separately in the undistorted, final WCS system. These images provide the information necessary for refining the image registration for each of the input images. They also provide the images which will later be combined into a median image used for the later blot and cosmic ray detection steps.

Input: Aside from the input parameters, this step requires:

- valid input images with SCI extensions
- valid distortion coefficients tables
- any optional secondary distortion correction images
- numarray object (in memory) for static mask

Output: This step produces:

- singly drizzled science image (simple FITS format)
- singly drizzled weight images (simple FITS format)

These images all have the same WCS based on the original input parameters and those provided for this step; specifically, output shape, pixel size, and orientation, if any have been specified at all.

5.4.10.2 Parameter Details

driz_separate: This parameter specifies whether or not to drizzle each input image onto separate output images. The separate output images will all have the same WCS as the final combined output frame. These images are used to create the median image, needed for the cosmic ray rejection step further on.

driz_sep_outnx: Size of the X axis of the output images, in pixels, on which each input will be drizzled onto. If no value is specified, it will use the smallest size that can accommodate the full dithered field.

driz_sep_outny: Size of the Y axis of the output images, in pixels, on which each input will be drizzled onto. If no value is specified, it will use the smallest size that can accommodate the full dithered field.

driz_sep_kernel: For the initial separate drizzling operation only, this specifies the form of the kernel function used to distribute flux onto the separate output images. The options are currently:

- *square*: original classic drizzling kernel
- *point*: this kernel is a point so each input pixel can only contribute to the single pixel which is closest to the output position. It is equivalent to the limit pixfrac -> 0 and is very fast.

- *gaussian*: this kernel is a circular gaussian with FWHM equal to the value of pixfrac, measured in input pixels.
- *turbo*: this is similar to kernel="square" but the box is always the same shape and size on the output grid and always aligned with the X and Y axes. This may result in a significant speed increase.
- *tophat*: the kernel is a circular "top hat" shape of width pixfrac. It effects only output pixels within pixfrac/2 of the output position are affected.
- *lanczos3*: a Lanczos style kernel extending 3 pixels from the center. The Lanczos kernel is a damped, bounded form of the "sinc" interpolator and is very effective for resampling single images when scale=pixfrac=1. It leads to less resolution loss than the other kernels, and also less correlated noise in outputs. It is however much slower. It should never be used for pixfrac not equal to 1.0 and is not recommended for scale not equal to 1.0.

The default for this step is "turbo" since it is much faster than "square", and it is quite satisfactory for the purposes of generating the median image. More information about the different kernels can be found in the help for the task 'drizzle'.

driz_sep_wt_scl: Weighting factor for input image. If driz_sep_wt_scl=exptime then the scaling value will be set equal to the exposure time found in the image header. This is the default recommended behavior. It is also possible to give wt_scl=expsq for weighting by the square of exposure time, which is optimal for read-noise dominated images.

driz_sep_scale: Linear size of the output pixels in arcseconds/pixel for each separate drizzled image (to be used in creating the median for cosmic ray rejection). The default value of INDEF specifies that the undistorted pixel scale for the first input image, as computed by **PyDrizzle**, will be used as the pixel scale for all the output images.

driz_sep_pixfrac: Fraction by which input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, "dropsize", of a pixel in units of the input pixel size. If pixfrac is set to less than 0.001, the kernel will be reset to "point" for more efficient processing. For the step of drizzling each input image onto a separate output image, the default value of 1 is best in order to ensure that each output drizzled image is fully populated with pixels from the input image. For more information, see the help file for the drizzle task.

driz_sep_rot: Position Angle of output image's Y-axis relative to North. A value of 0.0 would orient the final output image with North up. The default of INDEF specifies that the images will not be rotated, but will instead be drizzled in the default orientation for the camera, with the x and y axes of the drizzled image corresponding approximately to the detector axes. This conserves disk space, since these single drizzled images are only used in the intermediate step of creating a median image.

driz_sep_fillval: Value to be assigned to output pixels that have zero weight or did not receive flux from any input pixels during drizzling. This parameter corresponds to the 'fillval' parameter of the **drizzle**. If the default of 'INDEF' is used and if the weight in both the input and output images for a given pixel are zero, then the output

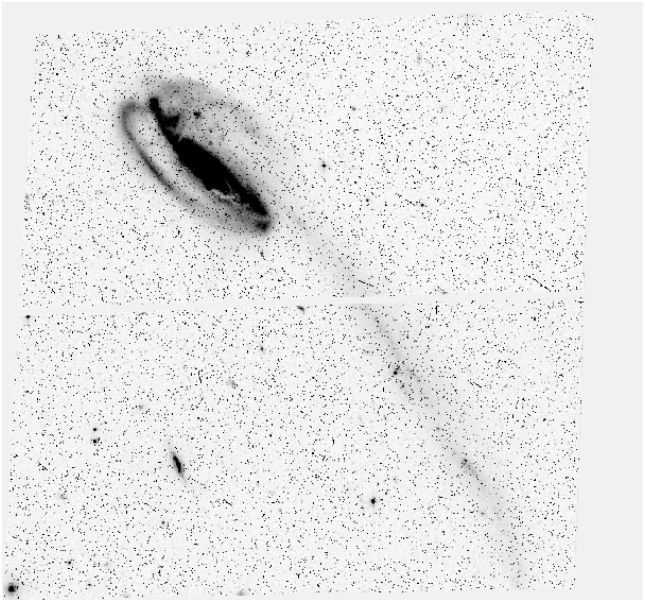
pixel will be set to the value it would have had if the input had a non-zero weight. Otherwise, if a numerical value is provided (e.g. 0), then these pixels will be set to that value.

driz_sep_bits: Integer sum of all the DQ bit values from the input image's DQ array that should be considered 'good' when building the weighting mask. This can also be used to reset pixels to good if they had been flagged as cosmic rays during a previous run of MultiDrizzle, by adding the value 4096 for ACS and WFPC2 data. Please see the section on Selecting the 'Bits' Parameter (Section 5.5.7) for a more detailed discussion.

5.4.10.3 Basic Example

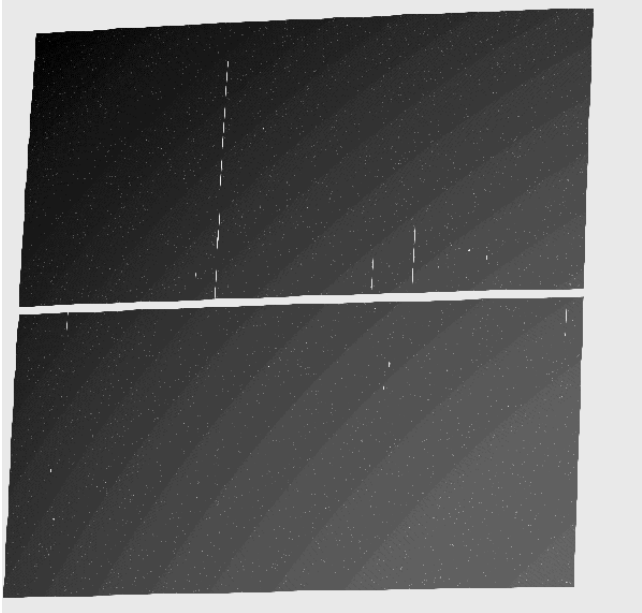
Figure 5.4 and Figure 5.5 contain two basic examples.

Figure 5.4: A Singly Drizzled ACS Image



The singly drizzled FLT image 'j8cw54ovq_flt_single_sci.fits'. The two chips have been combined and the geometric distortion has been removed.

Figure 5.5: Weight Map Corresponding to the Singly Drizzled ACS Image



The weight image 'j8cw54ovq_flt_single_wht.fits' corresponding to the singly drizzled image in Figure 5.5 where white indicates zero weight.

5.4.11 Create Median Image

5.4.11.1 General Explanation

The singly drizzled science images are combined to create a single median image. This median combination gets performed section-by-section from the input single drizzle images. Each section corresponds to a contiguous set of lines from each image taking up no more than 1Mb in memory, so that combining 10 input images would only require 10Mb for these sections. The goal of this step is to created an estimate of what the fully cleaned image should look like in order to enable better bad pixel detection as well as a chance to improve the alignment of the image stack. Creating a median image from the aligned and undistorted input images allows for a statistical rejection of bad pixels.

Input: Aside from the input parameters, this step requires access to the single drizzled images on disk.

Output: The final median image serves as the only output from this step.

5.4.11.2 Parameter Details

median: The user can specify whether or not to create a median image with this parameter. This median image will be used as the comparison 'truth' image: in the cosmic ray rejection step.

median_newmasks: The user can specify whether or not to create new mask files when creating the median image. These masks are generated from the weight files produced previously by the "driz_separate" step, and would contain all the bad pixel information. These pixels will be excluded when calculating the median. Generally

this step should be set to “yes”, unless it is desired to include bad pixels in generating the median.

combine_type: This parameter allows the user to choose which method is used to create the median image. Valid options are:

- average
- median
- sum
- minmed

The ‘average’, ‘median’, and ‘sum’ options set the mode of operation for using numcombine, a numarray method for median-combining arrays, to create the median image. The “minmed” option will produce an image that is generally the same as the median, except in cases where the median is significantly higher than the minimum good pixel value, in which case it will choose the minimum. The sigma thresholds for this decision are provided by the “combine_nsigma” parameter.

combine_nsigma: Sigmas used for accepting minimum values instead of median values when using the ‘minmed’ combination method. If two values are specified, then the first value will be used in the initial choice between median and minimum, while the second value will be used in the “growing” step to reject additional pixels around those identified in the first step. If only one value is specified, then it is used in both steps.

combine_nlow: When using a ‘minmed’ rejection method, this parameter sets the number of low value pixels to reject.

combine_nhigh: When using a ‘minmed’ rejection method, this parameter sets the number of high value pixels to reject.

combine_lthresh: Sets the lower threshold for clipping input pixel values during image combination. This value gets passed directly to ‘imcombine’ for use in creating the median image. If None, no thresholds are used at all.

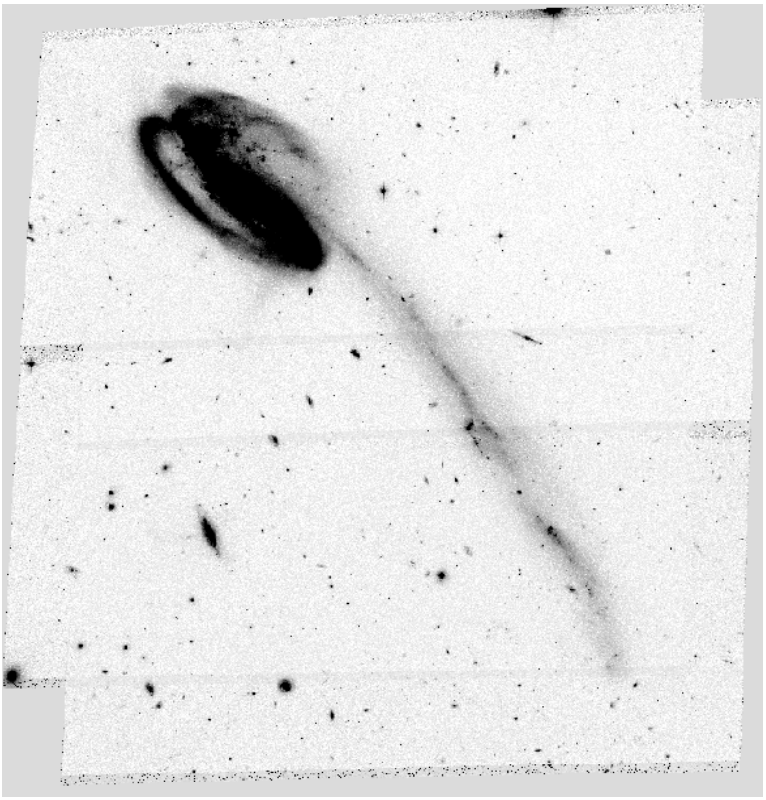
combine_hthresh: Sets the upper threshold for clipping input pixel values during image combination. This value gets passed directly to ‘imcombine’ for use in creating the median image. If None, no thresholds are used at all.

combine_grow: Width in pixels for additional pixels to be rejected in an image with a rejected pixel from one of the rejection algorithms. This parameter is used to set the ‘grow’ parameter in **imcombine** for use in creating the median image.

5.4.11.3 Basic Example

This is a cleaned median image created using six separately drizzled images and their associated bad pixel masks.

Figure 5.6: ACS Example of a Cleaned Median Image



The cleaned median image created using the 6 separately drizzled Tadpole images and their bad pixel masks.

5.4.12 Blot Median Image

5.4.12.1 General Explanation

The median image is the combination of the WCS aligned input images which have already had the distortion model applied. Taking the median of the aligned images allows for a statistical rejection of bad pixels from the image stack. The resulting median image can then be input to the blot task with the goal of creating 'cleaned' versions of the input images at each of their respective dither locations. These "blotted" images can then be directly compared to the original distorted input images for detection of bad-pixels, hot pixels, and cosmic rays whose locations will be saved to the output badpixel masks.

Input: Aside from the input parameters, this step only requires opening the single median image created from all the input images.

Output: A distorted version of the median image corresponding to each input 'chip' (extension) gets written out as output from this step as separate simple FITS images.

5.4.12.2 Parameter Details

blot: Perform the blot operation on the median image. The output will be median smoothed images which match each input chips location, these are used in the cosmic ray rejection step.

blot_interp: Type of interpolation to use when blotting drizzled images back to their original WCS. Valid options are:

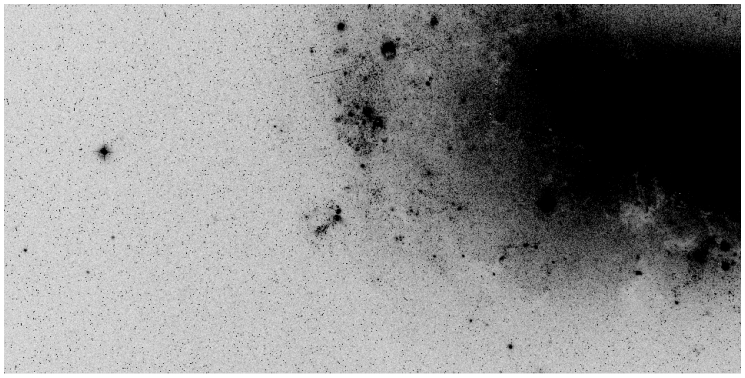
- nearest: Nearest neighbor
- linear: Bilinear interpolation in x and y
- poly3: Third order interior polynomial in x and y
- poly5: Fifth order interior polynomial in x and y
- sinc: Sinc interpolation; accurate but slow

The poly5 interpolation method has been chosen as the default because it is relatively fast and accurate. If 'sinc' interpolation has been selected, then it will use the value of the parameter 'blot_sinscl' to specify the size of the sinc interpolation kernel.

blot_sinscl: Size of the sinc interpolation kernel in pixels.

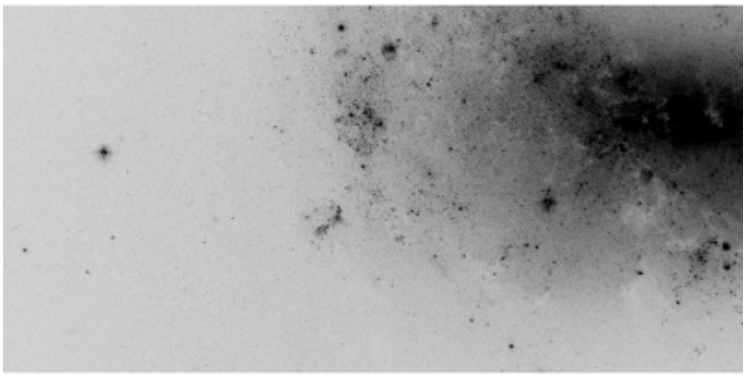
5.4.12.3 Basic Example

Figure 5.7: Original ACS FLT Image Fed into the Drizzle Step



The original FLT image that was fed into the single drizzle step.

Figure 5.8: The Returned Corresponding Blotted Imager



The blotted image, this is a cutout of the median image at the location of the original image. Note the absence of cosmic rays because they were statistically rejected.

5.4.13 Remove Cosmic Rays

5.4.13.1 General Information

The blotted median images which are now transformed back into the original reference frame, get compared to the original input images to detect any spurious pixels (which may include pixels impacted by cosmic rays) in each input. Those spurious pixels then get flagged as 'bad' in the output mask files, which get used as input for the final combination, so that they do not show up in the final product.

Input: Aside from the input parameters, this step requires:

- the blotted median images, and
- the mask files.

Output: The identified bad pixels get flagged by updating the input mask files. Optionally, copies of the original images with the bad pixels removed can be created through the use of the 'driz_cr_corr' parameter.

5.4.13.2 Parameter Details

driz_cr: Perform cosmic ray detection? If set to "yes", it will detect cosmic rays and create cosmic ray masks using the algorithms from **deriv** and **driz_cr**.

driz_cr_corr: Create a cosmic ray cleaned input image? The cosmic ray cleaned _cor image will be generated directly from the input image, and a corresponding _crmask file will be written to document the pixels detected as affected by cosmic rays.

driz_cr_snr: These values specify the signal-to-noise ratios for the **driz_cr** task to use in detecting cosmic rays. This parameter value gets passed directly to **driz_cr**;

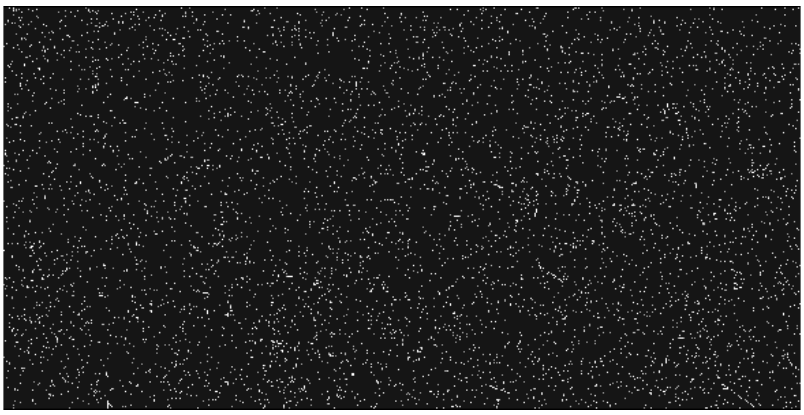
driz_cr_scale: Scaling factor applied to the derivative in **driz_cr** when detecting cosmic rays. This parameter gets passed directly to **driz_cr**.

driz_cr_grow: Radius (in pixels) around each detected cosmic ray to use more stringent detection criteria for additional cosmic rays.

driz_cr_cte_grow: Length (in pixels) of CTE tail to mask in drizzled output.

5.4.13.3 Basic Example

Figure 5.9: Single chip ACS Cosmic Ray Mask



A single cosmic ray mask 'j8cw54orq_flt_sci2_crderiv.fits'. This mask should be blinked with the original image 'j8cw54orq_flt' input file to assure that optimal parameters were chosen in the **driz_cr** task.

5.4.14 Create Final Cleaned, Combined Product

5.4.14.1 General Explanation

This step performs the final image combination of the original input images (or their copies) using the updated mask files to remove any cosmic rays. The output frame, just like the single drizzle step, can be redefined here using some parameters for this step, otherwise it will use the defaults to (ideally) include all the input pixels from all the input images after registering them according to their header's WCS information.

Input: Aside from the input parameters, this step requires:

- all input image SCI arrays
- updated mask files
- all distortion coefficients files

Output: The final product of MultiDrizzle is a registered, cosmic ray cleaned, distortion-free, photometrically flat science image with associated weight and context images. By default, these will be written out as a single multi-extension FITS file, but the user could simply have them written out as separate simple FITS images.

5.4.14.2 Parameter Details

driz_combine: This parameter specifies whether or not to perform the final drizzle image combination. This applies the bad pixel masks to the input images and creates a final, cleaned, distortion-corrected product.

final_wht_type: Specify the type of weighting image to combine with the bad pixel mask for the final drizzle step. The options are:

- EXP

  The default of 'EXP' indicates that the images will be weighted according to their exposure time, which is the standard behavior for drizzle. This weighting is a good approximation in the regime where the sky noise dominates. This option is provided as the default since it produces reliable weighting for all types of data, including older instruments (e.g., WFPC2) where more sophisticated options may not be available.

- ERR

  Specifying 'ERR' is an alternative for ACS and STIS data, in which case the final drizzled images will be weighted according to the inverse variance of each pixel in the input exposure files, calculated from the error array data extension that is in each calibrated input exposure file. This array encapsulates all the noise sources in each exposure, including read-noise, dark current and sky background, as well as Poisson noise from the sources themselves, and this also includes a dependence upon exposure time. For WFPC2, the ERR array is not produced during calibration, therefore this option is not available. But for ACS and STIS datasets this option is generally recommended to be the most accurate type of weighting for producing the final drizzled image.

- IVM

  In this case the user can either supply their own inverse-variance weighting map or let MultiDrizzle generate one on-the-fly automatically during the final drizzle step. This may be necessary for specific purposes, for example to create a drizzled weight file for software such as SExtractor, which expects a weight image that contains all the background noise sources (sky level, read-noise, dark current, etc.) but not the Poisson noise from the objects themselves. The user can create the inverse variance images and then specify their names using the 'input' parameter for MultiDrizzle to specify an '@file'. This would be a single ASCII file containing the list of input calibrated exposure filenames (one per line), with a second column containing the name of the IVM file corresponding to each calibrated exposure. Each IVM file must have the same file format as the input file, and if given as multi-extension FITS files (for example, ACS or STIS data) then the IVM extension must have the EXTNAME of 'IVM'. If no IVM files are specified on input, then MultiDrizzle will rely on the flat-field reference file and computed dark value from the image header to automatically generate an IVM file specific to each exposure.

final_outnx: Size of the X axis of the final drizzled image (in pixels). If no value is specified, it will use the smallest size that can accommodate the full dithered field.

final_outny: Size of the Y axis of the final drizzled image (in pixels). If no value is specified, it will use the smallest size that can accommodate the full dithered field.

final_kernel: Shape of the kernel used by 'drizzle' in the final image combination. The supported choices are:

- *square*: original classic drizzling kernel

- *point*: this kernel is a point so each input pixel can only contribute to the single pixel which is closest to the output position. It is equivalent to the limit pixfrac -> 0. It is very fast.
- *gaussian*: this kernel is a circular gaussian with FWHM equal to the value of pixfrac, measured in input pixels.
- *turbo*: this is similar to kernel="square" but the box is always the same shape and size on the output grid and always aligned with the X and Y axes. This results in a significant speed increase in some cases.
- *tophat*: the kernel is a circular "top hat" shape of width pixfrac. In effect only output pixels within pixfrac/2 of the output position are affected.
- *lanczos3*: a Lanczos style kernel extending 3 pixels from the center. The Lanczos kernel is a damped, bounded form of the sinc interpolator and is very effective for resampling single images when scale=pixfrac=1. It leads to less resolution loss than the other kernels, and also less correlated noise in outputs. It is however much slower. It should never be used for pixfrac not equal to 1.0 and is not recommended for scale not equal to 1.0.
- The default for this step is "square"

final_wt_scl: Weighting factor for input image. If final_wt_scl = exptime, then the scaling value will be set equal to the exposure time found in the image header. This is the default recommended behavior. It is also possible to give wt_scl=expsq for weighting by the square of exposure time which is optimal for read-noise dominated images.

final_scale: Linear size of the output pixels in arcseconds/pixel for the final combined product. The default value of INDEF specifies that the undistorted pixel scale for the first input image, as computed by **PyDrizzle**, will be used as the pixel scale for the final output image.

final_pixfrac: Fraction by which input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, "dropsize", of a pixel in units of the input pixel size. If pixfrac is set to less than 0.001, the kernel is reset to 'point' for more efficient processing. If more than a few images are being combined, values smaller than 1 (e.g. 0.7 or 0.8) can be specified, which result in a slightly sharper output image. For more information, read the help for the task 'drizzle'.

final_rot: Position Angle of output image's Y-axis relative to North. The default of 0.0 would orient the final output image with North up. A value of INDEF would specify that the images will not be rotated, but will instead be drizzled in the default orientation for the camera, with the x and y axes of the drizzled image corresponding approximately to the detector axes.

final_fillval: Value to be assigned to output pixels that have zero weight or did not receive flux from any input pixels during drizzling. This parameter corresponds to the 'fillval' parameter of the drizzle task. If the default of INDEF is used and if the weight in both the input and output images for a given pixel are zero, then the output pixel will be set to the value it would have had if the input had a non-zero weight.

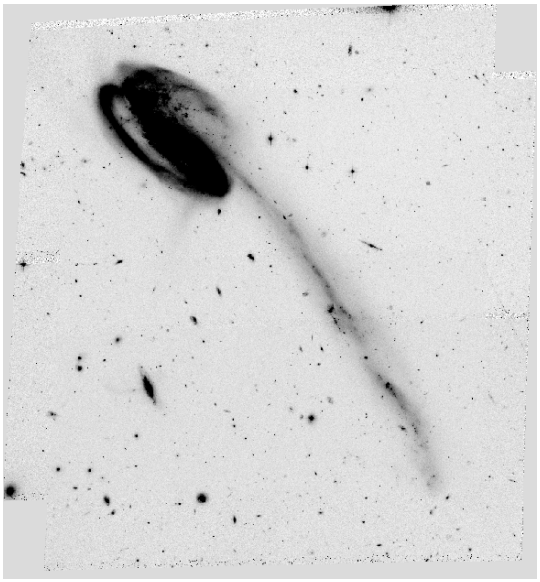
Otherwise, if a numerical value is provided (e.g. 0), then these pixels will be set to that value.

final_bits: Integer sum of all the DQ bit values from the input image's DQ array that should be considered 'good' when building the weighting mask. This can also be used to reset pixels to good if they had been flagged as cosmic rays during a previous run of MultiDrizzle, by adding the value 4096 for ACS and WFPC2 data. The section on selecting the bits parameter (Section 5.5.7) provides more details on which bits should be included.

5.4.14.3 Basic Example

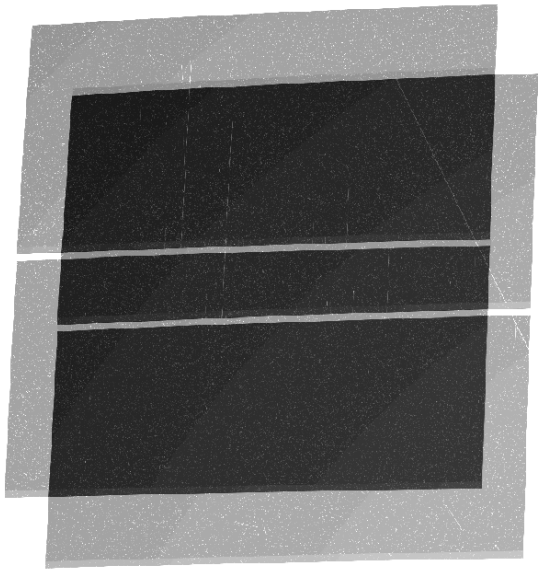
Figure 5.10 shows the SCI extension, Figure 5.11 shows the WHT extension, and Figure 5.12 shows the CTX extension of the drizzled product.

Figure 5.10: Science Extension of an ACS Drizzled Output Product



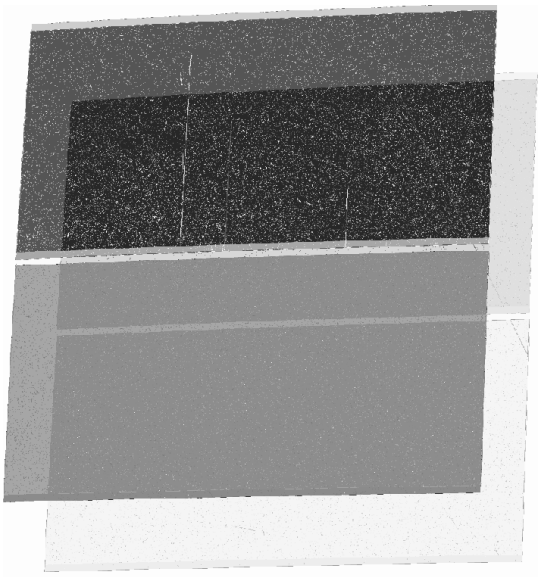
The science (SCI) extension of the drizzled product. This image has been corrected for distortion and drizzled onto a single mosaic using the six images in the dither pattern.

Figure 5.11: Corresponding Weight Extension of the ACS Drizzled Output



The corresponding weight (WHT) extension of the drizzled product.

Figure 5.12: Corresponding Context Extension of the ACS Drizzled Output



The corresponding context (CTX) extension of the drizzled product.

5.4.15 Override Instrument Specific Parameters

5.4.15.1 General Information

The user may choose to override the information contained in the headers of the images by setting these parameters directly.

5.4.15.2 Parameter Details

gain: Detector gain
gnkeyword: Detector gain keyword in header
rdnoise: Detector read noise
rnkeyword: Detector read noise keyword in header
exptime: Exposure time
expkeyword: Exposure time keyword in header
crbit: Bit value for CR identification in DQ array

5.5 Optimizing the Drizzle Parameters for Your Data

5.5.1 Creating Custom Association Tables

Association tables for dithered, REPEAT-OBS, or CR-SPLIT observations are generated by the calibration pipeline. These tables keep track of the input exposure filenames and the output product filenames. Some types of observations, however, will not have association tables. Others will have multiple tables from different visits which need to be combined into one. In the following discussion, we present the methodology for creating custom association tables, either by merging association tables or creating them from scratch. Users also have the capability to manually edit association tables to include any known image offsets. This can be done using the **ttools** task **tedit** in **IRAF**, where the user adds the columns XOFFSET, YOFFSET and/or ROTATION to the association table. Alternately, an ascii file with the known image offsets may be specified and the association automatically updated.

5.5.1.1 Merging Association Tables

Observing programs which cover a large portion of the sky will generally be made up of multiple pointings. Each of these pointings may be dithered or split for cosmic ray rejection and will possess their own association table. In order for **PyDrizzle** to produce a single combined product, a composite association table must be built from the separate association tables. Users can easily create this composite by merging the individual tables using the **ttools** task **tmerge** with "option = append".

The default product rootname will be taken from the first listed DTH-PROD in the composite association table. This rootname can be modified by the user (with tedit) to suit the desired naming convention. Generally, this rootname should match the rootname of the composite association table. A detailed example of merging association tables is given in step 1 of Example 4 from the ACS Data Handbook, Section 3.5.2.

5.5.1.2 Creating Association Tables from Scratch

In some cases, observations of the same target will not have accompanying association tables, for example observations taken in different visits or dithered observations taken using POS-TARG offsets. The **PyRAF** task **buildasn** (Section

5.5.2) has been developed for the **dither** package to create association tables which may be used for reprocessing with **PyDrizzle** or **MultiDrizzle**.

The following illustrates how to use Buildasn (Section 5.5.2) to create an association table from all calibrated input files with the FLT suffix found in a single directory. Within **PyRAF** and using the **IRAF** syntax, we type:

```
pyraf> buildasn mymosaic suffix='flt.fits'
```

Alternately, the same result may be obtained using the Python syntax:

```
pyraf> iraf.buildasn('mymosaic',suffix='flt.fits')
```

The association table will have the name 'mymosaic_asn.fits', the product name will be 'mymosaic_drz.fits', and all files with the FLT suffix found in the working directory will be included in the association. To specify a subset of images within a given directory, the user may specify the 'suffix' parameter to be a filelist ('@filename'), a wild-card list ('*8cd*flt*'), or any portion of the filename ('crj' or 'f555w').

If user determined offsets are available, **buildasn** (Section 5.5.2) has the capability of incorporating them into the association table. These offsets (XOFFSET, YOFFSET, and/or ROTATION) are specified by the file listed in the "shiftfile" parameter.This option allows users to fine-tune the final image combination by providing corrections to the header WCS information, if required.

```
pyraf> buildasn mymosaic suffix='flt' shiftfile='shift.txt'
```

MultiDrizzle utilizes the **buildasn** (Section 5.5.2) task and creates association tables automatically when a file suffix (i.e. 'flt.fits') or a file list and/or a shift file is specified in the input parameters.

5.5.2 Buildasn

5.5.2.1 Parameters

The **PyRAF** task **buildasn** was written to help users create an association table for data which was not originally considered part of an association. It is a member of the **IRAF dither** package and requires the following inputs:

Table 5.9: Parameters for Buildasn

Parameter	Default Value	Description	Format
asnroot		Rootname for the ASN table	string
suffix	crj	input file description for association members, may contain wildcards and "@" list	string
shiftfile		optional file containing user defined shifts in order of x,y,rotation	string
product		output filename for the ASN table	string
verbose	no	Print additional information while building the table?	boolean [False,True]

5.5.2.2 Example Usage

In order to create a new association table, you must first make sure that the correct packages are loaded. In the Python shell (this will work for both **PyRAF** and a regular Python shell) you can use the asnUtil package in one of three ways:

```
python>from glob import glob
python>from pytools import asnutil
python>input=glob("*flt.fits")
python>asnt=asnutil.ASNTable(input)

or

python>asnt=asnutil.ASNTable(input,output='myasntable')


or

python>asnt=asnutil.ASNTable(input,shiftfile="shifts.txt")

and then:

python>asnt.create()
python>asnt.write()
```

Or from the **PyRAF** shell:

```
pyraf>dither
pyraf>buildasn mymosiac suffix='flt'
```

both of these command sets build an association table called mymosaic_asn.fits using all the *_flt.fits images that are available in the current directory. If you were to input this asn file into **PyDrizzle** or **MultiDrizzle**, then the output product would be called mymosaic_drz.fits. **MultiDrizzle** can take as input any combination of asn tables, filenames or wildcard selection criteria and will use the asnutil package to generate the association table to feed to **PyDrizzle**.

5.5.2.3 Shift Files

The shift file is an ASCII file containing the user-computed shifts for a set of images and is used for updating the offsets in the association table. Observers may use either the association (ASN) table delivered by the pipeline or create their own. The shift file can contain shifts for files other than those contained in the ASN table you wish to update, only those entries found in the ASN table will be used to update the table. Thus, a single shift file can be generated for a whole set of observations which are represented by multiple association tables. For more detailed information on how to calculate shifts for your images, please see the section on Alignment (Section 5.5.3).

The shift file uses the following format:

```
# units: pixels
# frame: input
# form: absolute
# refimage: <filename>
filename1  xshift1 yshift1  [rotation1  [scale1]]
filename2  xshift2 yshift2  [rotation2  [scale2]]
```

The refimage filename requires the specification of the extension from which the WCS information will be read, but only in the case where the refimage is a multi-extension FITS image.

The specification of the filenames for the input images need to include the full filename, with suffix and extension; for example, j94f05bgq_flt.fits.

The entries for rotation and scale for each input image are optional. However, if a scale difference is noted for an image, a value must be given for the rotation as well.

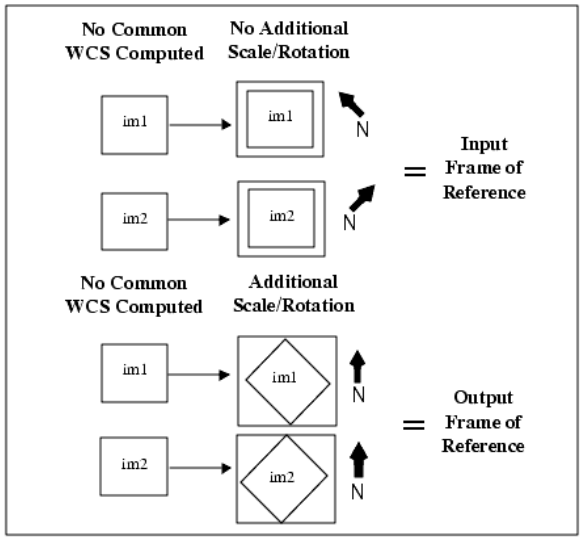
The units for the items specified in the first three lines are:

```
units                    Values: pixels (default), arcseconds
frame                    Values: input (default), output
form                     Values: absolute (default), delta
```

The first three lines specify the units of the shifts; the frame of reference, whether the shifts were computed using the input images (input) or singly drizzled images (output); the form of the shifts, whether they should be added to the header information as additional incremental shifts (delta) or the explicit shift that should be applied in total (absolute). The '#' symbol is required to signify that these are commentary lines in the file. When shifts are given in the 'output' frame of reference, it is necessary to specify the name of the reference image used. This reference image contains a copy of the WCS information from the image that was used as a reference to calculate the shifts and is only required when 'frame = output'. If shifts were calculated from single drizzled images which have been distortion corrected, rotated or altered in scale then this image must be one of the drizzled files so that the corrections are taken into account when the shifts are added to the original WCS information.

Absolute Shifts

Figure 5.13: Absolute Shifts in the Input and Output Reference Frames



Absolute Shifts in the 'input' and 'output' frame of reference. The direction of north is indicated by the bold arrows.

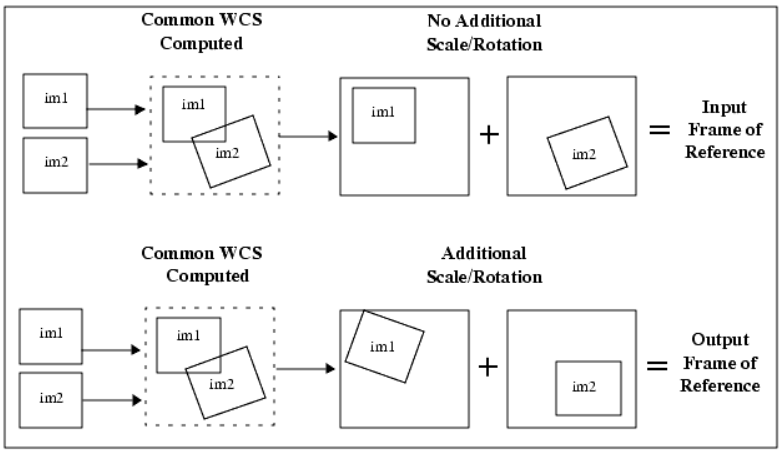
Figure 5.13 shows an absolute shift which is the total shift between two images and includes offsets implied by the header WCS. Absolute shifts may be determined by separately drizzling images to their own unique WCS frame. PyDrizzle will use the header information to optimally place the drizzled image in the frame. If absolute offsets are provided in the shift file, PyDrizzle will populate the OFFSET columns in the association and fill the DELTA values with zeroes.

To derive absolute shifts in the 'input' frame of reference, each distorted image is drizzled separately, and PyDrizzle uses the unique WCS information from each frame to choose the central RA/Dec, the image orientation, and final image size. No additional rotation or scaling is applied while drizzling. (Alternately, the user may catalog sources in each distorted image and then apply the distortion model to the X/Y positions. However, this requires tasks to transform coordinates from distorted to undistorted space)

Absolute shifts in the 'output' frame of reference may be computed by separately drizzling each distorted image, with the same rotation and scaling, onto a unique WCS frame. Target lists in each drizzled image may then be matched to derive the absolute shifts.

Delta Shifts

Figure 5.14: Delta Shifts in the Input and Output Reference Frames



Delta Shifts in the 'input' and 'output' frame of reference.

Figure 5.14 shows a delta shift defined as the residual shift required to align sources, **after** applying the offsets implied by the header WCS. Delta shifts may be determined by separately drizzling images onto a common WCS frame with the same central right ascension and declination. This is performed in the 'driz_separate' step of MultiDrizzle (Section 5.4) or by setting the **PyDrizzle** parameter 'single=yes'. Object lists derived for each drizzled image may then be matched and fit to derive a single delta shift for each image.

Delta shifts are defined to be in the 'input' frame of reference when the distortion-corrected, drizzled image has the same orientation and scale as the distorted image. Shifts will be in the 'output' frame of reference if any rotation or scaling was applied while drizzling. When the shifts are given in the 'output' frame of reference, PyDrizzle requires the specification of the reference image to properly account for any orientation and scale changes when interpreting the shifts. It is recommended that shifts be specified in the output frame with a reference image to insure the proper interpretation of the shifts.

If delta shifts are provided, those values will be used to populate the DELTA columns and the OFFSET columns will be zero. If shifts are given in both the OFFSET and DELTA columns, the OFFSET column will be applied and the DELTA columns will be ignored. This convention used by **buildasn** eliminates any ambiguity as to which values are applied to the data. Editing the ASN table directly will then allow the user to further update the offsets using either delta or absolute shifts.

It should be noted that shift files can generally be provided directly to MultiDrizzle without having to edit the association tables directly. In fact, MultiDrizzle allows the user to specify the input files without the need for an association table at all, and it will create one as needed on its own for use in combining the images.

The following is an example of a shift file in units of pixels, measured in the 'input' frame of reference, with the form specified as 'delta' shifts for an ACS image set. An x

and y shift and a rotation (in degrees) is specified for each dataset. If only a simple shift is required, then the rotation column need not be specified.

```
#units: pixels
#frame: input
#form: delta
j8c0b1skq_flt.fits  -2.4 -1.2 -0.002
j8c0b1snq_flt.fits  -4.3 -2.3  0.001
j8c0b1sqq_flt.fits  -6.9 -3.5  0.003
```

This is what the output ASN table from **buildasn** should contain if the above shift file was used as input:

Table 5.10: Output ASN Table from **buildasn**

MEMNAME	MEMTYPE	XOFFSET	YOFFSET	XDELTA	YDELTA	ROTATION
j8c0b1skq	EXP-DTH	0	0	-2.4	-1.2	-0.002
j8c0b1smq	EXP-DTH	0	0	-4.3	-2.3	0.001
j8c0b1sqq	EXP-DTH	0	0	-6.9	-3.5	0.003

5.5.3 Alignment

5.5.3.1 Alignment Error Sources

Users are encouraged to reference the section on HST Pointing Accuracy and Stability (Section 4.1.1) for a discussion on error sources that derive from the stability of the telescope. Additional errors may arise from the nature of the science data itself such as:

- the ability to get accurate centroids on the objects in the image, which may include extended sources, dust obscuration, or merely faint objects with low signal to noise
- small rotation angles in addition to shifts between images which may complicate the offsets
- optical distortion which has not been fully accounted for
- long exposures which may suffer from blurring due to the changing nature of the 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, which can be downloaded from the Web at:

http://www.stsci.edu/hst/HST_overview/documents/calworkshop/workshop2002/CW2002_Papers/

5.5.3.2 Visit Alignment

The astrometry of images taken within the same visit and orbit of the telescope are generally limited to the accuracy of the telescope pointing which is controlled by the Fine Guidance Sensors (FGS) and the Guide Star Catalog. If you have external knowledge of the field (for example other ground or space based images that the current dataset can be linked to) then you can improve the inherent absolute astrometry of your image. If your images are populated with sufficient well exposed point sources, then the relative astrometry of each image can be improved through source detection and centroiding. These updates can then be folded back in to the astrometry information in the header of each image and used to combine all the exposures into a single well aligned and drizzled mosaic.

5.5.3.3 Inter-visit alignment

If you have more than one visit for your entire dataset, enough time has probably elapsed that the telescope has used different guide stars for the observations. This could result in alignment errors of many pixels depending on the platescale of the instrument that was used. There are two paths you can take to get the best alignment for the final mosaic. If you cannot improve the inherent astrometry of the images inside a single visit, then you can calculate shifts for all the images at the same time using one image from the entire dataset as a reference, as described in the previous section for visit alignment. You might choose the exposure with the longest observing time (this would provide the best SNR and number of available objects for matching) or some other image based on the requirements of the dataset.

If you have improved the alignment of the images inside each visit, you can then calculate the inter-visit offsets between the combined visit level mosaics by following these proscribed steps:

- Create drizzled images for each of the visits you would like combined
- Choose one visit to be the reference visit; the first image in this mosaic could be the final reference image used in your final drizzle
- Compute the offsets that exist between each visit and the reference visit
- Apply the offset for each visit to all the images inside that visit by either updating the headers or adding the additional shifts to your shift file
- Drizzle all the images together into a final mosaic

5.5.3.4 Combining large mosaics or data from multiple programs

The same methods are used to drizzle and align large mosaics and multiple programs as smaller ones. Every effort has been taken to ensure the drizzle algorithms are structured to provide the fastest computation and memory management. However, the user should consider limitations which exist due the size of the data, and the amount of memory available on the processing computer. Processing large datasets on a 32-bit system will be limited by the OS restrictions for addressing memory, so only 2Gb of memory can be used for all the output arrays. This results in a limiting size of about 16000 x 16000 for the combined image, if no context image is generated. This

limitation can be avoided by running the MultiDrizzle code on a 64-bit system where the OS can address much larger blocks of memory.

5.5.3.5 User Defined Shifts

The ability of PyDrizzle and MultiDrizzle to properly align images depends on the accuracy of the World Coordinate System (WCS) and any other keywords that may deal with astrometry or distortion information in the header of the image. Unfortunately this does not always provide the most precise alignment, even for images taken within the same visit. Incorrect shifts between exposures can degrade image quality and corrupt cosmic ray rejection when left uncorrected. To resolve this, users can specify additional shifts that can be added to the header calculation. By deriving residual (delta) shifts based on the position of objects in the data after aligning the images based on the WCS information, the user may refine the alignment based on the WCS header information to create a precisely-aligned drizzled product, especially for images taken in multiple-visits. Many observers have developed their own methods of comparing images and computing offsets. The conventions described here should support the majority of users. In practice, the shifts are applied to the data using a shiftfile (Section 5.5.2) which can be incorporated into the association table for the dataset using the buildasn function (Section 5.5.2).

Deriving these shifts generally follows the same set of steps; namely,

- Identify sources in each of the images
- Identify which image will serve as the reference to which all other images will be aligned
- Cross-match sources from each image with the sources in the reference image
- Perform a fit on the cross-matched list of sources for each image relative to the reference image

Use of **IRAF** tasks for the fit, such as **geomap**, assume that all rotations occur about (0.,0.), which by default falls on the corner of each image. However, the drizzle algorithm rotates all input images around the center of the image, not the corner. This requires that the positions for all sources need to be specified relative to the center of the output frame instead of the corner. This can be done by simply subtracting the coordinates of the output frame's center pixel from all the sources position prior to performing the fit. If this is not done, a residual offset will be introduced after applying the computed fit. The task **tweakshifts**, as described in the next section, demonstrates how to determine the shifts using **IRAF** based tasks while taking into account the correct center of rotation when performing the fit between source positions.

These shifts, regardless of how they were computed, should describe the offset which needs to be applied to the image to shift the objects so that they align with the same objects in the reference frame.

5.5.3.6 Tweakshifts

Tweakshifts provides an automated interface for computing residual shifts between input exposures being combined using MultiDrizzle or PyDrizzle.The shifts computed by Tweakshifts correspond to pointing differences after applying the WCS information

from the input image's headers. Such errors would, for example, be due to errors in guide-star positions when combining observations from different observing visits or from slight offsets introduced upon re-acquiring the guide stars in a slightly different position. This task was written using Python and relies on the Python interface to the **IRAF** environment, **PyRAF**, for its operation. As a result, this task can only be run under **PyRAF**. The primary implementation of **tweakshifts** involves using existing **IRAF** tasks such as **daofind**, **geomap**, and **crossdriz** to compute the shifts. Tweakshifts automates the operation of these tasks to produce a shiftfile and reference WCS file which can serve as input to MultiDrizzle. The shiftfile will contain the residual shifts for each input image, while the reference WCS file consists of a FITS header with all the WCS keywords appropriate for the reference image without any image data. The reference WCS will then be used by MultiDrizzle to properly interpret the shifts from the shiftfile for the final alignment of the images.

Tweakshifts supports multiple methods for determining the residual shifts between images: primarily, catalog matching and cross-correlation. Each mode of operation has its own requirements for input file formats in order to properly compute the shifts between the images.

Catalog Matching

A widely utilized method for computing offsets between images consists of identifying sources in each image, matching them up and solving the matched sets of positions to find offsets. This technique requires that the image contain recognizable sources which are point-sources, or similar enough to point-sources so that they can be identified by the software as a source. In addition, there has to be enough overlap between the sources identified from each input exposure to positively identify the same target in each image and allow the production of a sorted matched list of targets. Tweakshifts relies on either the **daofind** task or Source Extractor as the object identification routine. The first input image gets selected as the reference image for the final product. This results in each of the remaining image's source lists being matched to the reference image source list using **xyxymatch** producing an output matched list for each image relative to the reference image positions. These matched lists will only contain sources that are found in common between each image and the reference image. The matched lists will then be used as input to **geomap** to compute the final shift, rotation and scale change for each input image relative to the reference image.

Normally, distortion-free input images would be required as input in order to allow positive identification of sources relative to the reference image. For ACS, this would require the use of MultiDrizzle (or PyDrizzle) to generate distortion-corrected images, such as the singly-drizzled images from MultiDrizzle. However, tweakshifts supports the use of calibrated, distorted images (such as FLT images for ACS, or.c0h images for WFPC2) as input images. The use of distorted input images requires that the undistort parameter be set to yes to turn on distortion correction of the source objects positions from each image. This removes the necessity to actually run MultiDrizzle, PyDrizzle, or drizzle on each input.

Several other parameters act to directly control the operation of the underlying **daofind**, **xyxymatch** and **geomap** tasks to allow the user to fine-tune the results. The "computesig" parameter, for example, sets Tweakshifts so that it will compute a global

value for the sigma based on the sky value found in the reference image. This value will then be used in **daofind** to tune the object-finding algorithm for the images being combined. Since it is a global value, all input exposures need to be normalized to the same units, either count rates or counts or electrons, in order for the object identification to work the same on all input images.

The source lists from each image generally will include cosmic rays as detected sources, sometimes significantly confusing object identification between images. Long-exposure observations often have more cosmic ray events that source objects, so weeding them out in those cases would improve the efficiency of identifying common sources between images. One such method for trimming potentially bad or confusing sources from each source list would be to set a flux limit and only use sources above that limit. The fluxes reported in the default **daofind** source object lists are given as magnitude values. Thus, setting a limit based on the **daofind** magnitudes for the reference image as the fluxmax or fluxmin parameters and setting the ascend parameter to yes would allow the source lists to be trimmed of all sources fainter than the provided limit. This new trimmed source list would then be used in matching sources between images and for the final fitting for the shifts.

Cross-Correlation

The use of cross-correlation to determine shifts can be selected by setting the parameter findmode to "cross-corr". This technique allows shifts to be computed between images which consist primarily of large extended sources with few or no point-sources. The algorithm implemented by Tweakshifts relies on running the tasks **crossdriz** to perform the cross-correlations between the images and then **shiftfind** to determine the shifts from the cross-correlation images. As with the catalog finding method, several parameters have been provided to directly control the operation of these underlying **IRAF** tasks. The inputs for this step, however, MUST be distortion-free images which all share the same WCS. These can be generated as the single-drizzle products from MultiDrizzle by turning off all steps past the single drizzle processing.

Available Options

Table 5.11: Available Parameters for Tweakshifts

Parameter	Default Value	Description	Format
Basic			
input		ASN table, ASCII file single image	string
shiftfile		input shiftfile with initial values	string
reference	tweak_wcs.fits	filename of the OUTPUT reference WCS	string
output	shifts.txt	output shift file	boolean [True, False]
findmode	catalog	mode for finding shifts	string [catalog, cross-cor]
gencatalog	sextractor	generate catalog with this task	string [sextractor, daofind]
sextractpars	parameter listing	sextractor parameters	
undistort	yes	apply distortion correction to input image positions	boolean [False, True]
computesig	yes	automatically compute sigma for all inputs	boolean [False, True]
idckey	idctab	key for selecting idc table	string [idctab,cubic,trauger,none]
clean	yes	remove intermediate files	boolean [True, False]
verbose	no	print extra messages during processing	boolean [False, True]
Coordinate File Description			
catfile		file containing coordinate filenames for input files	string
xcol	1	column number for x positions	integer
ycol	2	column number for y positions	integer
fluxcol	3	column number for flux/magnitude values	integer
fluxmax	INDEF	maximum value for valid objects	real
fluxmin	INDEF	minimum value for valid objects	real
fluxunits	counts	units of flux values used for sorting	string [counts,cps,mag]
refnbright	INDEF	number of brightest objects to keep after sorting	int

Parameter	Default Value	Description	Format
Object Detection Parameters			
minobj	15	minimum number of objects acceptable for matching	int
nmatch	30	maximum number of objects to match	int
matching	tolerance	the matching algorithm	string [tolerance,trianges]
tolerance	1.0	the matching tolerance in pixels	real
fwhmpsf	2.5	the fwhm of the psf in scale units	real
sigma	0.0	standard deviation of the background in counts	real
datamin	INDEF	minimum good data value	real
datamax	INDEF	maximum good data value	real
threshold	4.0	threshold in sigma for feature detection	real
nsigma	1.5	width of convolution kernel in sigma	real
fitgeometry	rscale	fitting geometry	string [shift,xyscale,rotate,rscale,rxyscale,general]
function	polynomial	surface type	string [checbyshev,legendre,polynomial]
maxiter	3	maximum number of rejection iterations	int
reject	3.0	rejection limit in sigma units	real
Cross-correlation Parameters			
crossref		reference image for cross correlation	string
margin	50	margin to strip down	int
tapersz	50	edge region to taper	int
pad	no	pad working area to prevent wraparound affects?	boolean [False, True]
fwhm	7.0	cross correlation peak fwhm	real
ellip	0.05	cross correlation peak ellipticity	real
pa	45.0	cross correlation peak position angle	real
fitbox	7	box size used to fit cross correlation peak	int

5.5.4 Sky Subtraction

5.5.4.1 Introduction

The default behavior of MultiDrizzle in the pipeline is to estimate the sky by means of the statistical distribution of pixels in the image, and subtract that from copies of the input files that are used to create the final drizzled image. The input files that are delivered to the user from the archive are not modified, ie they are not sky subtracted, but the sky values calculated by MultiDrizzle are contained in the header keyword MDRIZSKY.

For cameras with multiple detectors (such as ACS/WFC, WFPC2, or WFC3), the sky values in each exposure are first measured separately for the different detectors. Then these different values are compared, and the lowest measured sky value is used as the estimate for all the detectors for that exposure. This is based on the premise that for targets consisting of large extended or bright sources, the pixel intensity distribution in one or more of the detectors may be significantly skewed toward the bright end by the target itself, thereby overestimating the sky on that detector. If the other detector is less affected by such a target, then its sky value will be lower, and can therefore also be substituted as the sky value for the detector with the bright source.

The default behavior for MultiDrizzle in the pipeline is to perform iterative sigma-clipping, starting with the full range of pixel intensity values and calculating the standard deviation, then rejecting pixels with values deviating from the mean by more than 4 sigma either way, repeating this for a total of 5 iterations, and then using the median value of the final distribution as the sky value. This gives good results for a wide range of datasets, but occasionally the sky is still slightly overestimated and can be improved by post-pipeline processing.

We recommend sky subtraction to be turned on for broad-band data or other data with significant amounts of sky. This is because the sky background is retained in the science data header files; the pipeline MultiDrizzle calculates a sky background value and subtracts it on-the-fly before creating the pipeline drizzled product, but does not modify the science pixel values in the original science files. Therefore, turning on this step when running MultiDrizzle offline will ensure that the sky background keywords are read from the header and are applied before drizzling the data.

Of course, it should be kept in mind that if the background of a field is this strongly affected by the bright sources within it, then the purpose of the background measurement is primarily to ensure that there are no more varying offset levels in all the exposures prior to combination, due to sky variations from one exposure to the next; the true background may be difficult or impossible to determine in such cases.

This is also why background subtraction is turned off by default when running MultiDrizzle in the pipeline on narrow-band data and UV observations that are “dark” (i.e., have no geocoronal emission in the filter bandpass), since: 1) the sky background through such filters is much lower than through an optical broad-band filter, and (2) such observations are often of extended diffuse emission-line targets whose flux is much higher than the background, thus any automated attempt to measure the background may introduce errors that are larger than the background itself. However,

if the user is able to determine an accurate background level in such cases, then the above mechanism may be used to propagate those values directly to MultiDrizzle.

Sky subtraction is generally recommended for optimal flagging and removal of CR's, when the sky background is more than a few electrons. However, some science applications may require the sky to not be removed. Thus, the final drizzle step can be performed with no sky subtraction. If you turn off sky subtraction, you should also set drizzle.pixfrac =1. Otherwise variations in sky between images will add noise to your data.

5.5.4.2 Methodology

The clipped mode is computed for each input chip and scaled to a reference plate scale as an estimate of the sky background. The lowest scaled value for each chip of an observation (file) is then re-scaled to the chip's plate scale and subtracted as the sky value. The primary header of each input image is updated with this value. In lieu of having MultiDrizzle compute the sky value, the user can supply their own sky value as a keyword in the input file header. This keyword name would then be given to MultiDrizzle in the 'skyuser' parameter.

Input: Aside from the input parameters, this step requires opening each input image to access the science (SCI) extensions for computing the sky values.

Output: The input file primary headers get updated with the computed sky value, and each input image's SCI array (or copy, if 'workinplace' is set to False), gets sky subtracted.

5.5.4.3 Parameter Details

skysub: Turn on or off sky subtraction on the input data.

skywidth: Bin width, in sigma, used to sample the distribution of pixel flux values in order to compute the sky background statistics.

skystat: Statistical method for determining the sky value from the image pixel values. Valid options are:

- median
- mode
- mean

skylower: Lower limit of usable pixel values for computing the sky. This value should be specified in units of electrons.

skyupper: Upper limit of usable pixel values for computing the sky. This value should be specified in units of electrons.

skyclip: Number of clipping iterations to use when computing the sky value.

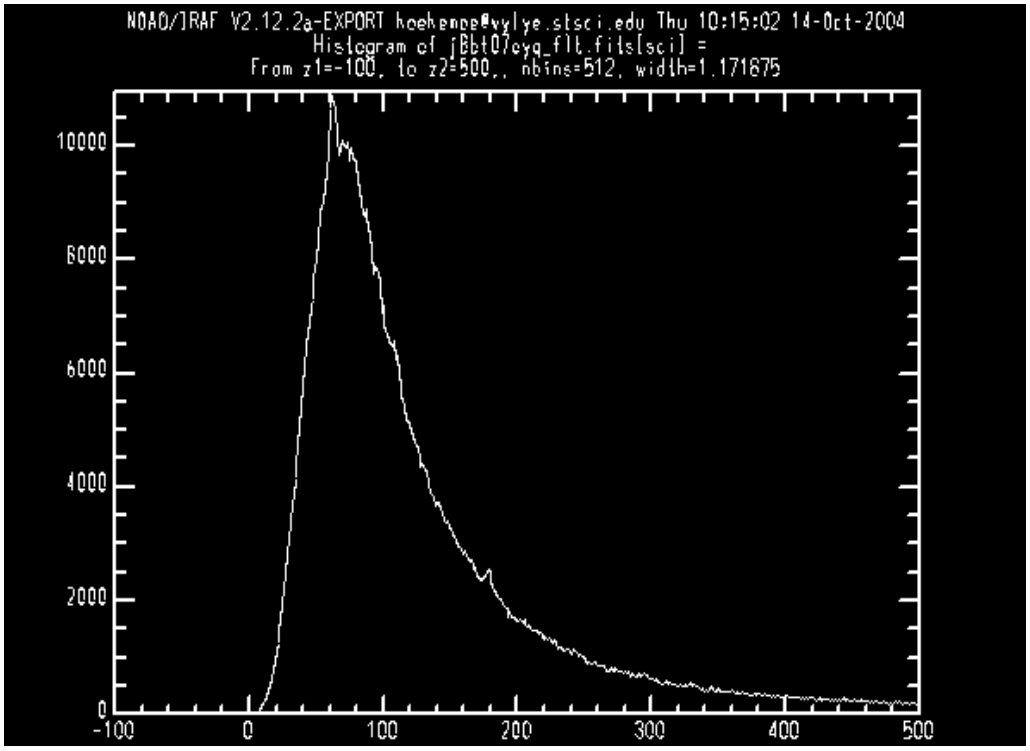
skylsigma: Lower clipping limit, in sigma, used when computing the sky value.

skyusigma: Upper clipping limit, in sigma, used when computing the sky value.

skyuser: Name of header keyword which records the sky value already subtracted from the image by the user.

5.5.4.4 Basic Example

Figure 5.15: Histogram Resulting from an Image Dominated by Scattered Light



Histogram of pixel intensity values in the calibrated science data of j8bt07oyq_flt.fits, which is dominated by scattered light from bright stars, introducing a significant bright skew to the distribution of pixel intensities near the background, thereby strongly affecting any automated estimate of the background. In cases such as this, the choice median, mean or mode can significantly affect the value assigned to the sky, and users may wish to try the various options, or carefully examine the statistics of their image before combining them.

5.5.5 Cosmic Ray Rejection

Few HST observing proposals have sufficient time to take a number of exposures at each of several dither positions. Therefore, if dithering is to be of wide-spread use, one must be able to remove cosmic rays from data where repeated images at the same position on the sky are not taken. We have therefore adapted Drizzle to the removal of cosmic rays, and automated this entire process in the task MultiDrizzle (Section 5.4). As the techniques involved in cosmic ray removal are also valuable in characterizing the image fidelity of drizzle, we will discuss them first. Here then is a short description of the method we use for the removal of cosmic rays:

1. Drizzle each image onto a separate sub-sampled output image using pixfrac = 1.0 where each image has the same WCS as the final output image.
2. Take the median of the resulting aligned drizzled images. This provides a first estimate of an image free of cosmic rays.

3. Map the median image back to the input plane of each of the individual images, taking into account the image shifts and geometric distortion. This can be done by interpolating the values of the median image using the **IRAF** program named **blot**.
4. Take the spatial derivative of each of the blotted output images. This derivative image is used in the next step to estimate the degree to which errors in the computed image shift or the blurring effect of taking the median could have distorted the value of the blotted estimate.
5. Compare each original image with the corresponding blotted image. 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 suspect pixel is masked.
6. Repeat the previous step on pixels adjacent to pixels already masked, using a more stringent comparison criterion.
7. Finally, drizzle the input images onto a single output image using the pixel masks created in the previous steps. For this final combination, a smaller pixfrac than in the first step will usually be used in order to maximize the resolution of the final image.

5.5.5.1 Image Alignment Requirement

One of the primary complicating factors in accurately determining what pixels are affected by cosmic rays remains the alignment of the images. Any misalignment of a star from one image to the next by more than 0.1 pixel will generally lead to a misidentification of a cosmic ray for the misaligned image. This level of error can result from pointing errors or inaccurate distortion models, both of which must be addressed prior to accurate identification of cosmic rays. The images generated in the first step of this process, the images drizzled to the final output WCS, can be used to verify the alignment and accuracy of the distortion-correction for each of the input images. Any processing which cross-matches sources from one image to the next can be used to compute the offset (shift) between the images. The task Tweakshifts (Section 5.5.3) demonstrates how these images can be used to determine the shifts using either catalog matching of sources or cross-correlation based on **IRAF** tasks such as **daofind**, **xyxymatch** and **geomap**.

5.5.6 Selecting the Optimal Scale and Pixfrac

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 re-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. The new shrunken pixels, or "drops", rain down upon the sub-sampled output. The value of an input pixel is averaged into an output pixel with a weight proportional to the area of overlap between the "drop" and the output pixel. If the drop size is sufficiently small not all output pixels will have data

added to them from each input image. One must therefore choose a drop size that is small enough to avoid degrading the image, but large enough so that after all images are drizzled the coverage is reasonably uniform. The drop size is controlled by a user-adjustable parameter called pixfrac, which is simply the ratio of the linear size of the drop to the input pixel (before any adjustment due to the geometric distortion of the camera). Thus interlacing is equivalent to drizzling in the limit as pixfrac $\rightarrow 0.0$, while shift-and-add is equivalent to pixfrac = 1.0. The degree of subsampling of the output is controlled by the user through the scale parameter, s, which is the ratio of the linear size of an output pixel to an input pixel.

This can be formally represented through the following relations. When a pixel (x_i, y_i) from an input image i with data value $d_{x_i y_i}$ and user defined weight $w_{x_i y_i}$ is added to an output image pixel (x_0, y_0) with the value $I_{x_0 y_0}$, weight $W_{x_0 y_0}$ and fractional pixel overlap $0 < a_{x_i y_i x_0 y_0} w_{x_i y_i} < 1$, the resulting values and weights of that same pixel, $I^{'}$ and $W^{'}$ are:

$$W^{'}_{x_0 y_0} = a_{x_i y_i x_0 y_0} w_{x_i y_i} + W_{x_0 y_0}$$

$$I^{'}_{x_0 y_0} = \frac{d_{x_i y_i} a_{x_i y_i x_0 y_0} w_{x_i y_i} s^2 + I_{x_0 y_0} W_{x_0 y_0}}{W^{'}_{x_0 y_0}}$$

where a factor s^2 is introduced to conserve surface intensity, and there i and o are used to distinguish the input and output pixel indices. In practice, drizzle applies this iterative procedure to the input data, pixel-by-pixel, image-by-image. Thus, after each input image is processed, there is a usable output image and weight, I and W. The final output images, after all input have been processed, can be written as:

$$W_{x_0 y_0} = a_{x_i y_i x_0 y_0} w_{x_i y_i}$$

$$I_{x_0 y_0} = \frac{d_{x_i y_i} a_{x_i y_i x_0 y_0} w_{x_i y_i} s^2}{W^{'}_{x_0 y_0}}$$

In nearly all cases $a_{x_i y_i x_0 y_0} = 0$, since very few input pixels overlap a given output pixel. When the dithered positions of the input images map directly onto the centers of the output grid, and pixfrac and scale are chosen so that p is only slightly greater than s, one obtains the full advantages of interlacing: because the power in an output pixel is almost entirely determined by input pixels centered on that output pixel, the convolutions with both p and G effectively drop away. Nonetheless, the small overlap between adjacent drops fills in missing data.

Noise considerations must also be taken into account and users are strongly encouraged to examine the section on Weight Maps and Correlated Noise (Section 3.3).

5.5.6.1 Choosing the drizzle.scale Parameter

The values used for the drizzle.pixfrac and drizzle.scale parameters depends on the number of input images and the size of the shifts. In general, the full-width at half-maximum of a PSF in the final image should be around 2.5 pixels for a well-sampled output image. In most cases choosing an output pixel size equal to 0.6 or 0.5 the input pixel works well with HST instruments. The output pixel size specified with the drizzle.scale parameter is given in arcseconds. This is because some instruments, such as WFPC2, have more than one pixel size associated with them. The drizzle.pixfrac parameter, however, is given as a fraction of the input pixel size.

5.5.6.2 Choosing the drizzle.pixfrac Parameter

For observers who generally have three to four dither pointings with 0.5 pixel shift increments, the drop size should be larger than the output pixel size. (for example, for WFPC2, scale=0.05 arcsec and pixfrac=0.6) — remember, the scale is now given in arcseconds while the pixfrac is relative to the input pixel. This choice of parameters allows some of the "drop" to spill over to adjacent pixels, thus recovering some resolution to the image. For offsets that are close to integer values, it is difficult to recover any resolution, and a large drop size is recommended (like pixfrac=0.8 or 0.9 with output pixel one-half the input).

One test of the choice of drizzle.scale and drizzle.pixfrac is to measure the r.m.s of the weight image away from the edges of the output image (preferably under a region of sky). The r.m.s. of the final drizzle weight image should be less than 20% to 30% of the median (use imstat to get these numbers). A larger r.m.s. suggests that the weights are so variable that the ignoring the weight image when doing photometry will add noticeably to the final photometric errors.

Since the behavior of the final image statistics will depend significantly upon the exact number of offsets, as well as the degree of sub-pixel sampling, observers are encouraged to experiment by re-running drizzle a few times with different values of pixfrac, to see which yields the best results for the data.

5.5.7 Selecting the 'Bits' Parameter

5.5.7.1 Data Quality Flags for Bad Pixels: The 'Bits' Parameter

The data quality flags which were set during calibration can be used as bit masks when drizzling, and the user may specify which bit values should actually be considered 'good' and included in image combination. This is done via the parameters 'driz_sep_bits' and 'final_bits'. Any pixel flagged otherwise will be considered 'bad' and will be given zero weight. If a pixel is flagged as bad but has non-zero weight in any other image, the pixel will be replaced with the value from that image during image combination. Otherwise, the pixel will be replaced with the 'fill value'. The default 'fill value' used by MultiDrizzle during pipeline processing is INDEF. If the 'fill value' is set to INDEF during reprocessing with MultiDrizzle, and if there are no pixels with non-zero weight, the pixel will retain its original value.

The flags for the DQ array are presented in the Data Handbook for each instrument. For ACS, for example, the bits value used during pipeline processing is 96 which is the sum of 32 and 64 (two flag values for warm pixels). Note that these pixels were flagged during calibration as "bad" or "suspect", but may have been corrected in later processing steps. The bits parameter indicates which "suspect" pixels to keep. For ACS, for example, when the total counts in a given pixel have exceeded the full well, the DQ flag is 256. However, testing shows that counts still accumulate in a highly linear manner and that ACS photometry of saturated stars is quite practical if using a gain that samples the full well depth.

The default bits value in the off-line MultiDrizzle software available for reprocessing is zero, but can be reset by the user prior to reprocessing. The value chosen for this parameter is completely up to the user and should be selected based on

the specific calibration needs. For ACS/HRC processing, the user may want to add 8 (masked by aperture) to the value so that pixels which lie behind the occulting finger mask are not given zero weight. (In the ACS Data Handbook, Figure 4.7 of Section 4.5.1, bit 8 is not included in the bits parameter prior to drizzling. Pixels behind the occulting finger were therefore given zero weight and replaced with the default fill value=zero.) Of course, the photometry in this region will be inaccurate due to improper flat fielding, so this bit would likely only be set for aesthetic reasons.

5.5.7.2 Using the 'Bits' Parameter to Update the Masks

Before executing any of the 7 processing steps, MultiDrizzle completes several initialization steps which are described in Section 5.4.8. During this initialization process, two separate mask files are created for each chip.

The first of these files is called the single mask, and every pixel which has been flagged in the input image DQ array will be assigned a value of zero in the mask. All other pixels are assumed to be good and will be assigned a value of 1.0. 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 MultiDrizzle to ignore specific DQ flag values by specifying the parameter 'driz_sep_bits.' This can be particularly helpful when objects have been erroneously flagged as cosmic rays during pipeline processing. For ACS data, for example, the user would set 'driz_sep_bits=4096', and all pixels originally flagged as cosmic rays will be assumed to be 'good', and these pixels will instead be given a value of 1.0 in the single mask.

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 MultiDrizzle which 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.

The two mask files are created during the software initialization and subsequently updated when running MultiDrizzle steps 1 and 6 to compute the static mask and the cosmic ray mask. In step 1, the static mask is retained in memory and is combined with the previous versions of the single mask and final mask to create updated versions of these images. In step 6, the cosmic ray mask is computed, and when the parameter 'driz_cr_corr=yes', this mask will be written to a file called '*_crmask.fits'. This mask should be inspected and blinked with the original input image to verify the quality of the rejection. The final mask is then updated with those pixels flagged in the cosmic ray mask. In addition, the DQ array of the original input image is updated with the pixels flagged as cosmic rays.

During final drizzle combination in step 7, the DQ array of the input image is used in combination with the 'final_bits' parameter to produce a new version of the final mask which will be used to create the drizzled product '*_drz.fits'.

CHAPTER 6:

Real-World Examples

In This Chapter...

6.1 Limitations of Pipeline Processing

The goal of instrument pipelines are to provide data calibrated to a level suitable for initial evaluation and analysis for all users. Improvements to the calibration methods are immediately applied to the data when they are retrieved through the HST archives On-The-Fly reprocessing (OTFR) system. These same calibration script updates are propagated in releases of the IRAF/PyRAF STSDAS software package. If it has been a long time since your data was retrieved from the archive, it is advisable that you request them again to ensure that they contain the most up-to-date header information and calibrations.

There are presently two fundamental separate steps in the OTF process. The first is the calibration of individual data sets, and the second is combination of dithered datasets into a single output image using MultiDrizzle.

The second step cannot succeed without good results in the first. In some cases pipeline calibration of individual images will be insufficient. There are several occasions when On-The-Fly reprocessing is not ideal and when off-line interactive processing by the user is required. For instance the user may with to

- use personal versions of reference files,
- use non-default calibration switch values
- change default parameter values which do not completely remove hot pixels, cosmic rays, image persistence or other additional sources of noise.

NICMOS data in particular may require special attention by the user. Datasets often contain additional signal in the sky, persistence or pedestal (differing bias levels between quadrants in the chip) that require processing to be removed after the data have been calibrated through CALNICA. For more detailed information on recognizing and removing these effects in your NICMOS data see the data analysis chapter of the NICMOS Data Handbook which can be downloaded from:

http://www.stsci.edu/hst/nicmos/documents/handbooks/

Even when their individual datasets have been well calibrated, many users may wish to rerun the standard MultiDrizzle performed by the pipeline. The pipeline uses coarse values for both the output pixel size (scale) and drizzling kernel (pixfrac). This speeds up processing of the pipeline, and is sufficient to give the user a very good quick view of the field. However, when the user has well dithered data, he or she may wish to rerun MultiDrizzle at his or her home institution using parameters better suited for optimal cosmic ray removal and good final image resolution. In this chapter the user can find numerous examples showing how she or he may benefit from running MultiDrizzle on data from the archive.

The following sections illustrate specific examples of using the drizzle software with each of the instruments that it currently supports: WFPC2, ACS, WFC3, STIS/CCD, and NICMOS.

The intermediate and final drizzled images can take up a lot of space, and running the tasks in the dither package can be quite CPU-intensive, so care should be taken to make sure enough disk space is available before retrieving the images and running through the examples. All of the initial input images for each of these examples can be retrieved from the Hubble archive at

http://archive.stsci.edu/hst/search.php.

6.2 ACS

6.2.1 Introduction

The MultiDrizzle software resides in the stsdas.analysis.dither package and can only be run from within PyRAF. It contains an extensive set of parameters for user modification, but the default values should allow the processing of nearly any set of images for an initial review. From within the 'epar' window, the MultiDrizzle parameters are separated according to which drizzle task they control, making them easier to interpret. In default mode, MultiDrizzle performs each of its 7 steps in order:

1.) static mask
2.) sky subtraction
3.) drizzle separate
4.) median
5.) blot
6.) cosmic ray rejection
7.) final drizzle

MultiDrizzle may be executed from the command line, as part of a script, or run using the 'epar' facility which allows the user to view all parameters and turn particular steps on and off. We recommend that beginners use the 'epar' facility to become familiar with each step and to fine-tune the parameters in each step before running MultiDrizzle from the command line. For more details, a help document describing each task and its parameters may be accessed by typing 'help MultiDrizzle' from within PyRAF. In the examples below, several of the commands are too long to fit on a single line with the current document formatting. When a command is meant to continue on the same line, it will be followed by a backslash.

To understand the processing which took place in the pipeline, it can be helpful to inspect the MultiDrizzle parameter table which is given by the image header keyword MDRIZTAB. To query this information, the IRAF tasks 'hselect' and 'tread' may be used as illustrated below. It is important to note that default MultiDrizzle parameters (within the 'epar' facility) are not necessarily the same as the parameters specified in the MDRIZTAB during pipeline processing.

```
hselect *flt.fits[1] $I,mdriztab yes
tread table_mdz.fits
```

Drizzled products which were obtained from the archive have been processed using the parameters specified in the MDRIZTAB. These parameters work best for observations which were obtained as part of a pre-defined observing pattern and thus are 'associated' in the pipeline via an association table (*_asn.fits). For example, images which were obtained using a sub-pixel dither box pattern are usually aligned to better than 0.1 pixels and have accurate cosmic ray flags. When a sub-pixel dither pattern has been obtained, the final drizzle sampling can often be improved via manual reprocessing. The steps required to improve the drizzle sampling are described in Example 1. When images are obtained in separate visits, the image alignment and cosmic ray flagging can usually be fine-tuned, and these are described in Examples 2 and 3, respectively.

The calibrated products from CALACS (either the association table '_asn.fits' or the calibrated files with the extension '_flt.fits' or '_crj.fits') should be used as input for drizzling. The calibrated files have been corrected for bias, dark current and flat fielding. They are not corrected for distortion and may contain numerous cosmic rays, hot pixels, and other artifacts. These images appear square in detector space, but the pixels actually do not cover equal area on the sky. Once ACS images have been corrected for geometric distortion via drizzling, they will appear 'rhombus'-shaped, as shown in the examples below.

When manual reprocessing is desired, the calibrated data products and the distortion reference files should be placed on the user's local disk. The user is advised to retain a copy of the original archival data and place it in a separate directory. This is because MultiDrizzle will modify the input files, for example, when fine tuning the sky subtraction or cosmic ray rejection. The drizzle reference files include the geometric distortion table (IDCTAB= *_idc.fits), the distortion correction image (DGEOFILE= *_dxy.fits), and an optional MultiDrizzle parameter table (MDRIZTAB= *_mdz.fits). When combining images which were taken on different

dates (and therefore at different orientations), the software will automatically correct for time-dependent distortion and will print a message to the screen "Computing ACS Time Dependent Distortion Coefficients" during processing.

The drizzle reference files are assumed to be in a directory called 'jref'. This path must be defined before starting PyRAF and should be set to point to the user's local directory where the data resides. For example:

```
'setenv jref /mydata/jref/'
```

It is usually a good idea to make sure that you request that all the calibration files which were used to calibrate your data be delivered from the archive when you make the request for your science datasets.

The following three ACS examples are intended to be read consecutively, since each builds on information presented in the previous example. Testing was performed using the following software versions, which were current at the time of writing this document:

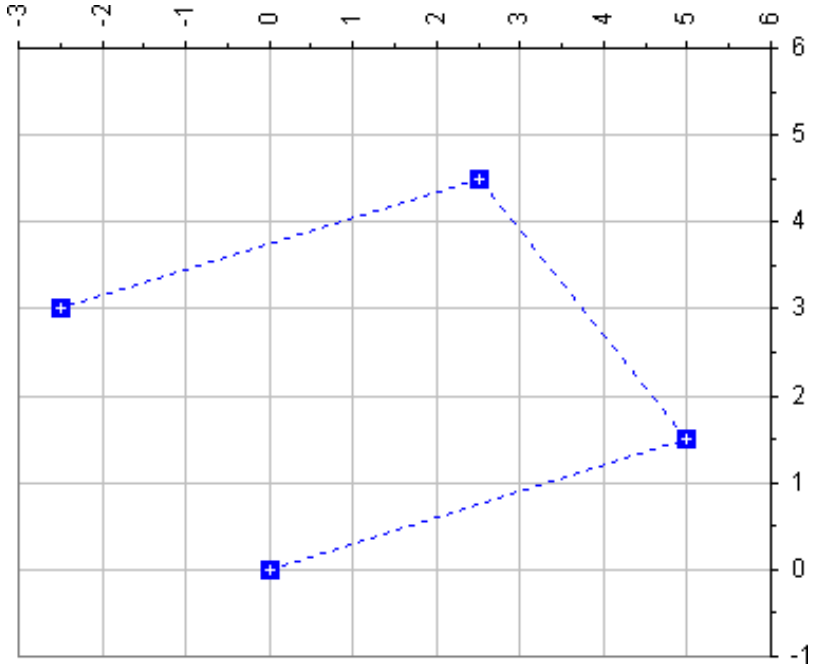
```
MultiDrizzle 3.2.1
NUMPY Version 1.0.4
PyFITS Version 1.4
PyDrizzle Version 6.3.0 (3-Sep-2008)
Python Version 2.5.1 (Mar 19 2008)
```

Users may obtain the latest public release of the STSDAS software (which includes the drizzling code) from STScI. Release notes for the latest version may be found on the STScI software Web page.

6.2.2 Example 1: Optimizing the Image Sampling for Single Visit

The following example describes the combination of four ACS/HRC images of the globular cluster M15 (Program 10401, PI Chandar) which were obtained with the F435W filter. These images were acquired in a single visit and are all at the same orientation. This program uses the default HRC dither pattern ACS-HRC-DITHER-BOX which has relative pixel coordinates (0, 0), (5.0, 1.5), (2.5, 4.5), (-2.5, 3.0). It is a parallelogram pattern designed for half-pixel sampling in both x and y, with overall dimensions large enough to help reject the larger detector artifacts.

Figure 6.1: A 4-point Dither Pattern for ACS-HRC



The ACS-HRC-DITHER-BOX is 4-point dither pattern designed for half-pixel sampling in both x and y.

The data described in this example are available from the HST archive for those who would like to repeat this example. The search parameters are 'Dataset' = 'j95z02010'. Under the archive 'Retrieval Options', the box for 'Science Files Requested' should be set to 'Calibrated' and the box for 'Reference Files' should be set to 'Best Reference Files'.

The archive will deliver the reference files required for running MultiDrizzle, including the geometric distortion table 'IDCTAB' = '_idc.fits', the distortion correction image 'DGEOFILE' = '_dxy.fits', and the MultiDrizzle parameter table 'MDRIZTAB' = '_mdz.fits'. Also delivered are the calibrated data products produced by CALACS which will be used as input to MultiDrizzle. These can be one of the following: 1.) the association table '_asn.fits', 2.) the flat-fielded images '_flt.fits', or 3.) the cosmic ray rejected, flat-fielded images '_crj.fits' (when available). Finally, the archive will deliver one or more drizzled data products '_drz.fits' which were created by running MultiDrizzle in the pipeline with a default set of parameters. The '_drz.fits' files may be saved in a separate directory for comparison with the drizzled products from this example.

The table below replicates the contents of the association table, where the rootnames of the four dithered images are given under the column 'MEMNAME'.

```
tprint j95z02010_asn.fits
```

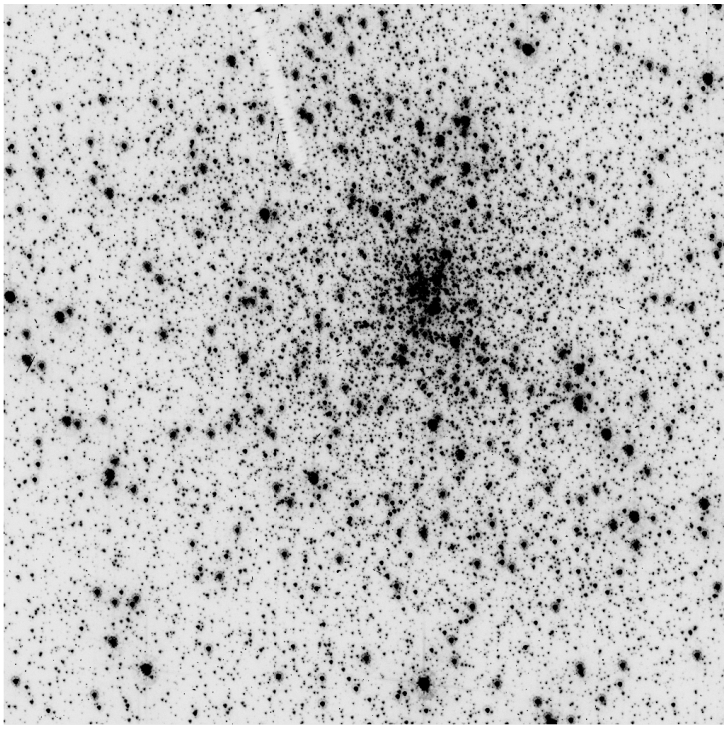
Table 6.1: Contents of the Association Table for the Dithered Images in this Example

MEMNAME	MEMTYPE	MEMPRSNT
J95Z02AQQ	EXP-DTH	yes
J95Z02ARQ	EXP-DTH	yes
J95Z02ASQ	EXP-DTH	yes
J95Z02ATQ	EXP-DTH	yes
J95Z02010	PROD-TARG	yes

Since these data were obtained as part of a sub-pixel dither box pattern, the default MultiDrizzle parameters applied during pipeline processing are adequate for aligning images (to better than 0.1 pixels) and for providing cosmic ray masks (where the 4096 flag is written to the DQ array of the calibrated image). To examine the quality of the pipeline cosmic ray masking, the user may blink the science and DQ extensions of the calibrated files, where the display range is chosen so that only flags with a value greater than 4000 are shown. The SCI and DQ extensions are shown in the figures below for the first image in the association.

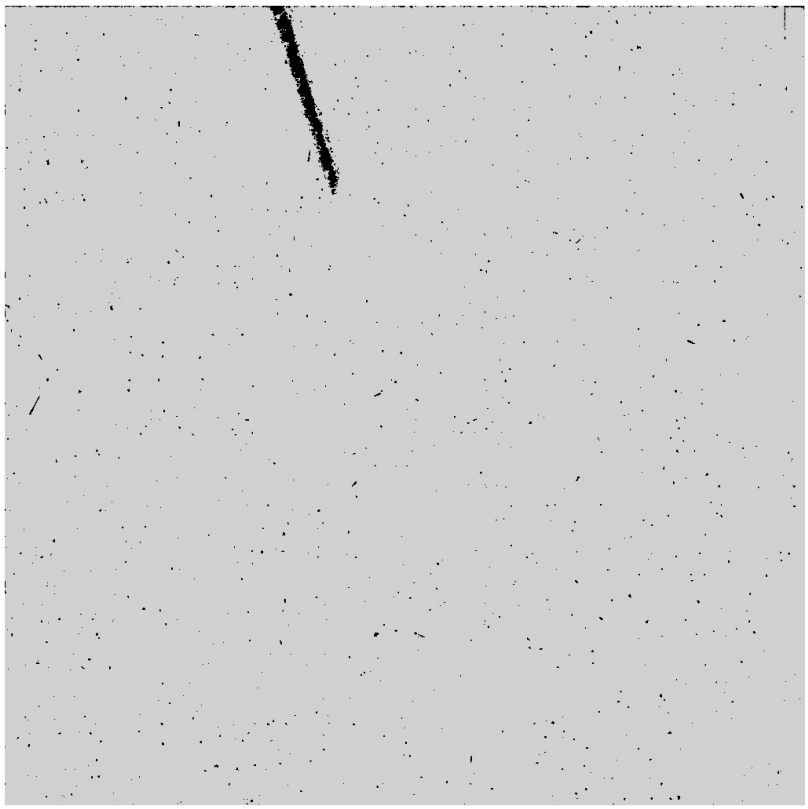
```
display j95z02aqq_flt.fits[1] 1 zs- zr- z1=0 z2=1000 fill+
display j95z02aqq_flt.fits[3] 2 zs- zr- z1=4000 z2=6000 fill+
```

Figure 6.2: Science Portion of the Calibrated ACS Image



The science portion of the calibrated image: ‘j95z02aqq_flt.fits[1].

Figure 6.3: Data Quality Portion of the Calibrated ACS Image



The data quality portion of calibrated image ‘j95z02aqq_flt.fits[3].

In default mode, MultiDrizzle performs each of its 7 steps in order. In this example, however, steps 1-6 can be turned off since the pipeline processing was adequate for aligning images and creating cosmic ray masks. To optimize the parameters for the final ‘driz_combine’ step, the user is encouraged to experiment with different combinations of the parameters: 1.) ‘final_scale’ (the size of the output pixels) and 2.) ‘final_pixfrac’ (the linear size of the ‘drop’ in input pixels). One must choose a ‘pixfrac’ value that is small enough to avoid degrading the final image, but large enough that when all images are dropped onto the final frame, the flux coverage of the output frame is fairly uniform. As suggested in the HST Dither Handbook, statistics performed on the drizzled weight image should yield an rms value (standard deviation) which is less than 20% of the median (midpoint) value. This threshold is a balance between the benefit of improving the image resolution at the expense of increasing the noise in the background from resampling the pixels. In general, the ‘pixfrac’ should be slightly larger than the scale to allow some spill over to adjacent pixels.

Before running MultiDrizzle, the package stsdas.analysis.dither must be loaded. The commands below show how to run a ‘test grid’ of varying ‘final_scale’ and ‘final_pixfrac’ values, where the default HRC scale is 0.025”/pixel. When specifying parameters from the command line within PyRAF, as illustrated below, the parameter must be preceded by the word ‘iraf’. Since the output file ‘hrc_drz.fits’ will be overwritten with each successive run, the example renames the drizzled product with a unique name between each separate trial.

```
   unlearn multidrizzle
   iraf.multidrizz.static=no
   iraf.multidrizz.skysub=no
   iraf.multidrizz.driz_separate=no
   iraf.multidrizz.median=no
   iraf.multidrizz.blot=no
   iraf.multidrizz.driz_cr=no
   iraf.multidrizz.driz_combine=yes
   iraf.multidrizz.final_rot=INDEF

multidrizzle *flt.fits output='hrc' final_scale=0.0250 final_pixfrac=1
imrename hrc_drz.fits hrc_drz_test1.fits

multidrizzle *flt.fits output='hrc' final_scale=0.0200 final_pixfrac=0.9
imrename hrc_drz.fits hrc_drz_test2.fits

multidrizzle *flt.fits output='hrc' final_scale=0.0175 final_pixfrac=0.8
imrename hrc_drz.fits hrc_drz_test3.fits

multidrizzle *flt.fits output='hrc' final_scale=0.0150 final_pixfrac=0.7
imrename hrc_drz.fits hrc_drz_test4.fits

multidrizzle *flt.fits output='hrc' final_scale=0.0125 final_pixfrac=0.6
imrename hrc_drz.fits hrc_drz_test5.fits
```

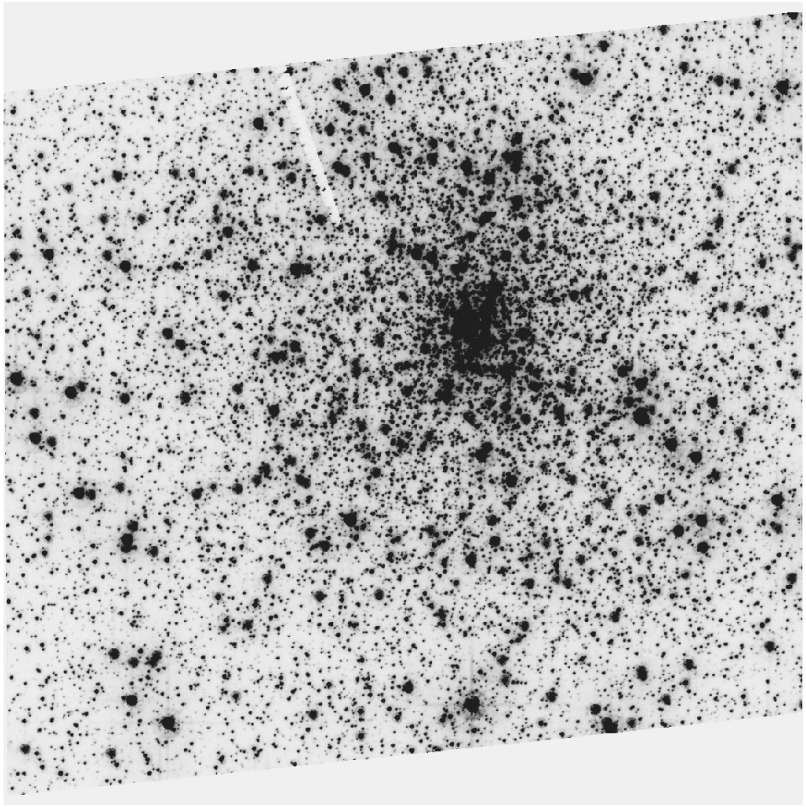
When the initial setup parameters are set: 'build=yes' (default) and 'context=yes' (non-default), the final MultiDrizzle output image will be a single multi-extension FITS file named 'hrc_drz.fits'. This file contains the science image in extension 1, the weight image in extension 2, and the context image in extension 3. When 'build=no', these files will be written to separate output files. When the default value 'context=no' is used, no context image is created.

The MultiDrizzle parameter 'final_rot' is the position angle of final drizzled image's Y-axis relative to North. The default of 0.0 would orient the final output image with North up. A value of INDEF would specify that the images will not be rotated, but will instead be drizzled in the default orientation for the camera, with the x and y axes of the drizzled image corresponding approximately to the detector axes.

The first extension of the drizzled product 'hrc_drz.fits[1]' contains the science (SCI) image which has been corrected for distortion which represents the combination of all eight dithered images. All pixels cover an equal area on the sky and have an equal photometric normalization across the field of view, giving an image which is both photometrically and astrometrically accurate for both point and extended sources. The dimensions of the output image are computed on-the-fly by MultiDrizzle and the default output plate scale is read from the 'scale' column in the IDCTAB. These parameters, however, may be chosen by the user to best suit the actual data. The SCI portion of the final drizzled product is presented in the figure below and is in units of electrons/sec. Changing the 'final_units' parameter from the default value 'cps' (counts per second) to 'counts' will produce a drizzled image in units of electrons.

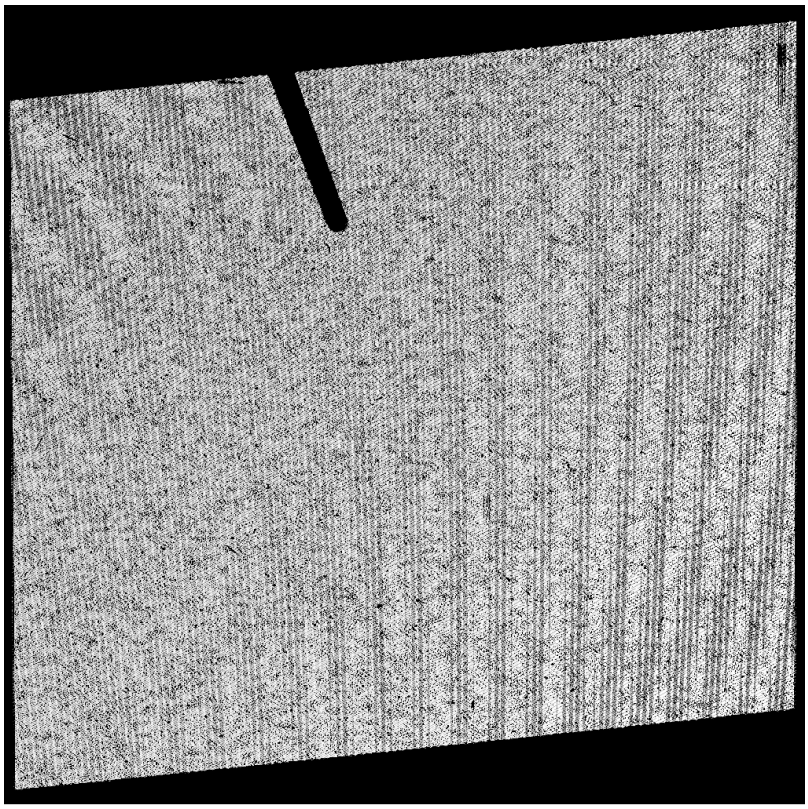
The second extension of the output image contains the weight (WHT) image. When the 'final_wht_type' is set to 'EXP', the weight image can be considered an effective exposure time map. The weight image from our example is shown in the figure below, where darker areas have higher weight.

Figure 6.4: Science Extension of the Final ACS Drizzled Product



The science extension of the final drizzled product.

Figure 6.5: Weight Extension of the Final ACS Drizzled Product



The weight extension of the final drizzled product for trial number 4 (scale=0.6, pixfrac=0.7).

In the table below, the statistics in the weight image (rms/median) are reported for each trial (1-5) using the task ‘imexam’ for a 500x500 pixel box in the center and in the lower right corner of each weight image. The PSF FWHM was measured using an isolated star at coordinate (878,966) in the first trial image, where scale=0.025” and pixfrac=1.0. Note that the table gives the value of ‘scale’ in two different ways, for illustration: as a fraction of the default plate scale and as an absolute size in arcseconds/pixel. MultiDrizzle uses the latter for defining the parameter ‘final_scale’.

Table 6.2: Weight Image Statistics and PSF FWHM for Various Final Drizzle Scale/Pixfrac Combinations

Trial Number	Pixfrac (fraction)	Scale (fraction)	Scale (arcsec)	RMS/Median (center)	RMS/Median (corner)	PSF FWHM (pixels)	PSF FWHM (arcsec)
1	1.0	1.0	0.0250”	0.078	0.162	2.06	0.0515”
2	0.9	0.8	0.0200”	0.100	0.154	2.58	0.0516”
3	0.8	0.7	0.0175”	0.125	0.133	2.83	0.0495”
4	0.7	0.6	0.0150”	0.129	0.164	3.23	0.0485”
5	0.6	0.5	0.0125”	0.143	0.251	3.69	0.0461”

The statistics of the weight image for the central portion of the chip meet the general requirement ‘rms/median < 20%’ for all five trials. In the lower right corner, however, trial #5 has a ratio greater than 20%. This is an indication that the ‘scale’ and ‘pixfrac’ parameters have been ‘shrunk’ too far (i.e. increasing numbers of output pixels which have no contribution from any of the four input pixels will result in ‘holes’ in the final image, and this is reflected by the increased rms of the weight image.) Looking at the weight image in the above figure, it is clear how the rms varies over the field of view. The majority of this variation is due to the change in geometric distortion over the chip, where the input pixels cover significantly different areas on the sky. (For more information, refer to the discussion on Pixel Area Maps.

Determining which is the best solution will depend on the position of the target on the detector. For a compact source in the center of the chip, trial #5 would be the best choice because the PSF FWHM is the narrowest, while the weight image ratio ‘rms/median’ is still less than 20%. For this example, however, stars are evenly distributed over the entire chip and trial #4 provides the best result, where ‘rms/median’ does not exceed 20% on any portion of the detector. The drizzled product for trial #4 is presented in the above figure, where the final ‘scale’ is equal to 0.6 times the default pixel scale and the final ‘pixfrac’ equal to 0.7. The resulting image has a plate scale of 0.0150”/pixel and the FWHM of the PSF in the optimized version of the image (in arcseconds) is 0.0485” compared to 0.0515” in the version created by the pipeline. Because the HRC detector is already well sampled at blue wavelengths, this is a relatively minor improvement. For the WFC, however, the detector pixels are significantly undersampled, and optimizing the final ‘scale’ and ‘pixfrac’ parameters will produce a much more dramatic improvement. For example, for the ACS-WFC-DITHER BOX, the PSF FWHM can usually be improved from ~0.10” to approximately ~0.06”.

6.2.3 Example 2: Optimizing the Image Alignment for Multiple Visits

The following example describes the combination of three single ACS/WFC images of the globular cluster NGC104 (Program 10737, PI Mack) which were obtained with the F606W filter. These images were acquired in separate visits and are all at unique orientations. Note that the second image was taken as part of a cosmic ray split association, so the ASN product rootname 'j9irw4041' should be selected from the archive query, and the two cosmic ray split '*_flt.fits' images will be delivered as part of the dataset.

The IRAF task 'hselect' may be used to query the image headers and compare the observing parameters.

```
hselect*flt.fits[1]'rootname,proposid,date-obs,filter*,
     aperture,orientat,exptime,ccdgain' yes
```

Table 6.3: Header Keywords for the Exposures of NGC104 Taken in Subsequent Visits

Rootname	Proposal ID	Observation Date	Filter Name	Aperture	Orientation	Exposure Time	Gain
j9irw3fwq	10737	2006-05-30	F606W	WFCENTER	-125.2	339.0	2
j9irw4b1q	10737	2006-07-08	F606W	WFCENTER	-88.6	339.0	2
j9irw5kaq	10737	2006-08-31	F606W	WFCENTER	-31.4	339.0	2

MultiDrizzle uses the information stored in the World Coordinate System (WCS) of the image header to align images. When the parameter 'driz_separate=yes', the input images will be corrected for geometric distortion and drizzled onto separate output frames which have a common WCS. Any required shifts, rotations, or scale changes are computed from the image headers. The output image dimensions are calculated on-the-fly and the pixel scale is taken from the column 'scale' from the IDCTAB reference file, where the default values are 0.05 arcsec/pix for the WFC and 0.025 arcsec/pix for the HRC and SBC. If drizzling is not done, the effects of distortion must be removed from the photometry by applying the Pixel Area Maps.

The distortion reference file is read from the image header via the IDCTAB and DGEOFILE keywords which specify the name and location of the appropriate reference files (*_idc.fits and *_dxy.fits). The distortion coefficients for each chip are written to temporary ascii files named '*_coeffs?.dat'. These files are useful for running several tasks in the drizzle package. If the user wishes to retain these files, the parameter 'clean' should be set to 'no'. In future versions of the software, the distortion information will be maintained in the image header via SIP coefficients, rather than external reference files.

By default, MultiDrizzle will look for all files in the working directory with the '_flt.fits' extension. Alternately, a file 'suffix' may be used to define the input image list, or a subset of images in a given directory may be specified via the 'filelist' parameter. The first image in the list will be defined as the reference image, and all the

others will be registered with respect to this WCS. The size of the output images will be chosen such that all the input images are contained within the array.

A reference image may be specified, and the input images will be drizzled to match the WCS of this image. Alternately, the central RA and Dec (ra, dec) of the reference pixel and the dimensions of the output frame (outnx, outny) may be specified, though reasonable values will be automatically determined from the images' WCS if these parameters are left blank. The central RA and Dec are specified in the initial setup parameters, and the output image dimensions are specified in both the 'driz_separate' and 'driz_combine' parameters in MultiDrizzle steps 3 and 7, respectively.

Within a single visit, small dithers are usually accurately reflected by the WCS. This is not always the case for multiple visits which often require guide star re-acquisitions and may utilize different guide stars. Even for back-to-back exposures that are part of an association, residual offsets on the order of a few tenths of a pixel can be significant enough to degrade final combinations. It is therefore essential to verify the presence of image-to-image shifts, rotations, and/or scale variations before combining data with MultiDrizzle. These residual offsets may be determined by separately drizzling onto a common output frame and then matching source lists to derive a single correction for each image.

The input files are:

1. the calibrated images '*_flt.fits'
2. the reference files '*_idc.fits' & '*_dxy.fits' (obtained from the archive)

The output products are:

1. the single science and weight images: '*_single_sci.fits' & '*_single_wht.fits'
2. the coefficient files '*_coeffs?.dat' (one for each chip)
3. the mask files '*_single_mask?.fits' & '*_final_mask?.fits' (one for each chip)

MultiDrizzle uses the asnUtil package to create an association table whose name is defined by the MultiDrizzle parameter 'output'. This association will be used to define the data set, and the header WCS from the entire set is used to define a common output frame. If user defined shifts are available, these may be specified via the 'shiftfile' parameter, and the association table will be updated. A shift file is not usually available until after separately drizzling the images onto a common WCS and source lists have been matched.

By default, the 'driz_separate' step uses the drizzle kernel 'turbo', the linear drop size 'pixfrac=1', and an output pixel scale equal to the input scale. For more information on setting these parameters, refer to the HST Dither Handbook. These parameters can be changed when the images are obtained as part of a sub-pixel dither pattern; for example, masks can be substantially improved by specifying a smaller value of scale, with the primary trade-off being increased computation time and much larger images (the size increases as the inverse square of the value of 'scale').

In this example, we have left the 'driz_separate' parameters to their default values. While the final drizzled image would contain 3 extensions in a single file (the science,

weight, and context images), the 'driz_separate' products are separate science and weight images named '*_single_sci.fits' and '*_single_wht.fits'. By default, ACS drizzled images are corrected for exposure time and are output in units of electrons per second.

The separately drizzled science images may be used to improve the image registration prior to final drizzle combination. While the WCS information for images taken within a single visit are usually adequate to align them, there can be significant residual offsets between visits. Shifts which are determined from separately drizzling images onto a common WCS are by definition 'delta' shifts, and will be applied in addition to any offsets from the WCS when a shift file is provided.

The images have been renamed as shown below for the purposes of illustration within this example. Next, the images are separately drizzled onto a common output frame which is free from distortion.

```
imrename j9irw3fwq_flt.fits f606w_01_flt.fits
imrename j9irw4b1q_flt.fits f606w_02_flt.fits
imrename j9irw5kaq_flt.fits f606w_03_flt.fits

unlearn multidrizzle
iraf.multidrizz.clean = no
iraf.multidrizz.build = no
iraf.multidrizz.static = no
iraf.multidrizz.skysub = no
iraf.multidrizz.driz_separate = yes
iraf.multidrizz.median = no
iraf.multidrizz.blot = no
iraf.multidrizz.driz_cr = no
iraf.multidrizz.driz_combine = no

multidrizzle *_flt.fits output=f606w
```

Next, the user should measure the positions of stars in the separately drizzled images, and compute a shift file which defines the residual offsets between images which were not accounted for using the header WCS. To find stars using the task 'daofind', it is necessary to first measure the standard deviation of the sky background using the task 'imexamine'. In this example, setting the object detection threshold to 50 sigma above the local background results in ~16K sources detected over the field of view which can be used to perform catalog matching. When specifying IRAF task parameters from the command line within PyRAF, the parameter must be preceded by the word 'iraf.' as shown below. When specifying these parameters from the command line within IRAF, the prefix 'iraf.' is not necessary.

```
apphot
iraf.findpar.thresh=50
iraf.datapar.sigma=0.02
daofind f606w_01_single_sci output=f606w_01.cdt inter- verif-
daofind f606w_02_single_sci output=f606w_02.cdt inter- verif-
daofind f606w_03_single_sci output=f606w_03.cdt inter- verif-
```

To match star lists using 'xyxymatch', an initial guess of the residual image offsets may be determined by displaying each single drizzled image and measuring the xy-shift of a single star with respect to the first (reference) image.

```
displ f606w_01_single_sci 1 zs- zr- z1=-0.1 z2=1
displ f606w_02_single_sci 2 zs- zr- z1=-0.1 z2=1
displ f606w_03_single_sci 3 zs- zr- z1=-0.1 z2=1
```

In this case, we find that the stars in the second visit are offset from the first visit by approximately (-29,-40) pixels. The tolerance for matching has been set to 2 pixels in this example; however, if sufficient numbers of stars are not matched, the tolerance can increased slightly.

```
iraf.xyxymatch.matching='tolerance'

  xyxymatch inp="f606w_02.cdt" refer="f606w_01.cdt"\
      out="f606w_match2.cdt" xin=-29 yin=-40 xrot=0 yrot=0 toler=2

  xyxymatch inp="f606w_03.cdt" refer="f606w_01.cdt"\
      out="f606w_match3.cdt" xin=-8  yin=-30 xrot=0 yrot=0 toler=2
```

The resulting matched coordinate lists now contain ~6K stars, since cosmic rays and saturated star spikes have now been removed by matching coordinate lists. Using the task 'tvmark', the user may overplot the original coordinate list in blue, the stars matched with image two in red, and the stars matched with image three in green.

```
tvmark 1  f606w_01.cdt mark=point nx=0 ny=0 points=3 color=203 inter-
tvmark 1  f606w_match2.cdt mark=point nx=0 ny=0 points=3 color=204 inter-
tvmark 1  f606w_match3.cdt mark=point nx=0 ny=0 points=3 color=205 inter-
```

To compute a more accurate estimate of the residual offsets, it is necessary to compute the geometric transformation between images. This can be done using the matched star lists which contain 4 columns each: xref, yref, xin, yin. The task 'geomap' allows the user to specify the fitting geometry to be used. In this case, we use 'fitgeom=rscale' which tells 'geomap' to solve for an x and y shift, a single rotation, and a single scale correction. (See the help file for 'geomap' for more details on the fitting geometry.)

It is important to note that 'geomap' computes the transformation about the pixel (0,0), while MultiDrizzle applies the shifts with respect to the center of the drizzled image. Therefore, it is necessary for the user to subtract the image center from the star coordinates in order to compute the correct offsets. The reference center of the drizzled output images can be determined with the task 'hselect' as follows:

```
hselect *sci.fits $I,D001OUXC,D001OUYC yes
```

The reference center in this example is at the position (2732.5, 2958.5), and these x and y values must be subtracted from the matched star lists before computing the transformation.

```
tcalc *match*.cdt  c7  "c1-2732.5"
tcalc *match*.cdt  c8  "c2-2958.5"
tcalc *match*.cdt  c9  "c3-2732.5"
tcalc *match*.cdt c10  "c4-2958.5"
```

Finally, the corrected columns for each image can be extracted and piped to a new output file which will be used by 'geomap' to compute the transformation.

```
fields f606w_match2.cdt 7,8,9,10 > f606w_match2_sub.cdt
fields f606w_match3.cdt 7,8,9,10 > f606w_match3_sub.cdt

geomap f606w_match2_sub.cdt datab=cdt.db fitgeom=rscale inter=no\
    reject=3 maxiter=5 xmin=INDEF xmax=INDEF ymin=INDEF ymax=INDEF
geomap f606w_match3_sub.cdt datab=cdt.db fitgeom=rscale inter=no\
    reject=3 maxiter=5 xmin=INDEF xmax=INDEF ymin=INDEF ymax=INDEF
```

The resulting offsets were recorded to the nearest 0.01 pixel and are listed below in the form of a shift file. The ascii file below is called 'shift.dat' and contains the residual offsets which were derived by the user using the results from 'geomap'. The columns are as follows: image, x_resid, y_resid, rotation, scale. For observations which are taken in different visits (and therefore at different orientations), residual rotations and scales are usually required.

```
# units: pixels
# form: delta
# frame: output
# refimage: refimage.fits
f555w_01_flt.fits   0.00   0.00 0.0000 1.00000
f555w_02_flt.fits -29.25 -40.58 0.0046 1.00008
f555w_03_flt.fits  -7.61 -29.87 0.0008 0.99996
```

Before moving on to further processing steps, the single drizzled image should be copied to a file called 'refimage.fits'. The residual offsets given in the shift file are defined with respect to this reference image. MultiDrizzle may now be called again, this time specifying the name of the file with the residual offsets, and the single drizzled images will now be accurately aligned.

```
imcopy f606w_01_single_sci.fits refimage.fits

multidrizzle *_flt.fits output=f606w shiftfile='shift.dat'
```

6.2.4 Example 3: Optimizing the Cosmic Ray Rejection for an ACS Mosaic with Sub-pixel Dithers

The following example describes the combination of eight ACS/WFC images of the galaxy NGC 4449 (Program 10585, PI Aloisi) which were obtained with the F555W filter. These images were acquired in two separate visits and are all at the same orientation. A combination of two ACS-WFC-DITHER-LINE patterns were used for this program. The primary pattern has two points shifted roughly 5x60 pixels apart to

span the WFC interchip gap, so that combined data will not have missing data there. The secondary pattern is a 2-point sub-pixel dither pattern which, when used in conjunction with the gap dither, allows the user to sub-sample the PSF. The two image associations j9cd01020 and j9cd02020 correspond to the left and right halves of the galaxy, and this is reflected in the target RA and Dec information. These data are available from the HST archive for those who wish to repeat this example. Approximately 5 GB of free disk space is required when intermediate products are not removed.

An outline of the drizzling process for the NGC4449 dataset is described below. While MultiDrizzle may be run from start to finish in a single pass, we have chosen to break up this example into several sections so that intermediate products can be examined and fine-tuned before moving on to later processing steps.

Table 6.4: Calibrated datasets used in the accompanying MultiDrizzle example. The two sets of images have a significant shift to image the left and right halves of the galaxy. Within a given visit, small sub-pixel dithers have been specified in accordance with the WFC-DITHER-BOX pattern.

Association	Dataset	Observation Date	RA/Dec (degrees)	POS-TARG1 (arcsec)	POS-TARG2 (arcsec)	Exposure Time (seconds)
j9cd01020	j9cd01kqq_flt	2005-11-10	187.0656 +44.1148	0.000	0.000	608
	j9cd01l5q_flt	2005-11-10	187.0656 +44.1148	0.123	0.084	615
	j9cd01ldq_flt	2005-11-10	187.0656 +44.1148	0.247	2.984	616
	j9cd01mfq_flt	2005-11-10	187.0656 +44.1148	0.370	3.068	621
j9cd02020	j9cd02phq_flt	2005-11-11	187.0128 +44.0854	0.000	0.000	608
	j9cd02ppq_flt	2005-11-11	187.0128 +44.0854	0.123	0.084	615
	j9cd02pxq_flt	2005-11-11	187.0128 +44.0854	0.247	2.984	616
	j9cd02q5q_flt	2005-11-11	187.0128 +44.0854	0.370	3.068	621

When large volumes of data are being processed, it can be helpful to rename the input files in the working directory in a meaningful way. The IRAF task ‘hselect’ may be used to query the image headers to assist in defining a naming convention. For example:

```
 hselect images="*flt.fits[1]" expr=yes \
   fields='rootname,date-obs,targname,ra_targ,dec_targ,postarg*,\
exptime,filter*'
```

The output from 'hselect' tells us that the first four images are centered on the left half of the galaxy 'POSA' and the second four images are centered on the right half of the galaxy: 'POSB'. In this example, the images have been renamed 'f555w_pos?_0?_flt.fits', where the first index refers to the target position, and the second index refers to the sub-pixel dithers 1-4 in the box pattern. For illustration, only two images have been renamed in the box below, but in practice, the user should rename all eight images.

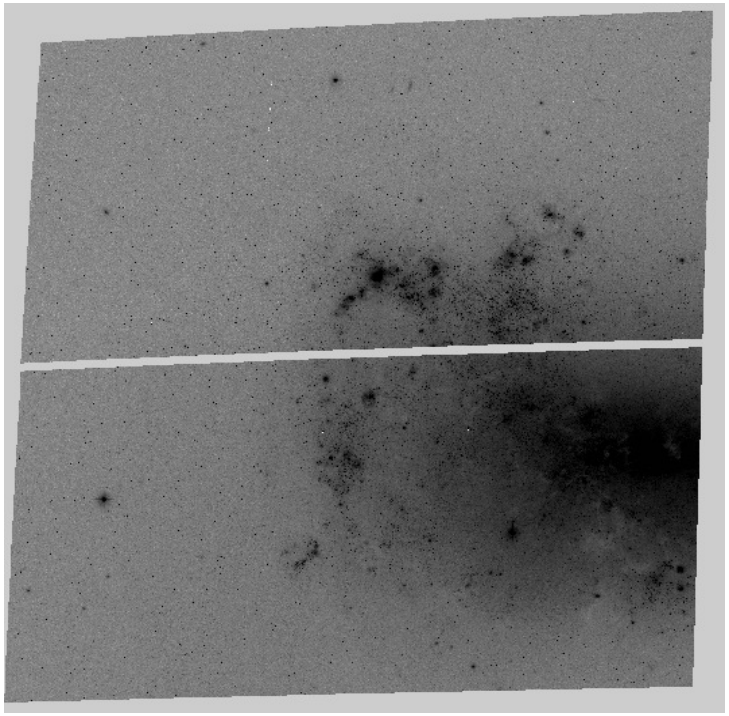
```
imrename j9cd01kqq_flt.fits f555w_posa_01_flt.fits
imrename j9cd02phq_flt.fits f555w_posb_01_flt.fits
```

6.2.4.1 Improving the Alignment

The images should first be singly drizzled by running MultiDrizzle step 3 (driz_separate), as described in Example 2. The separately drizzled science images may be used to improve the image registration prior to final drizzle combination. In this example, the images form two groups of three. While the WCS information for images within a single group (visit) are adequate to align them, there is a small residual offset between visits. Shifts which are determined from separately drizzling images onto a common WCS are by definition 'delta' shifts, and will be applied in addition to any offsets from the WCS when a shift file is provided.

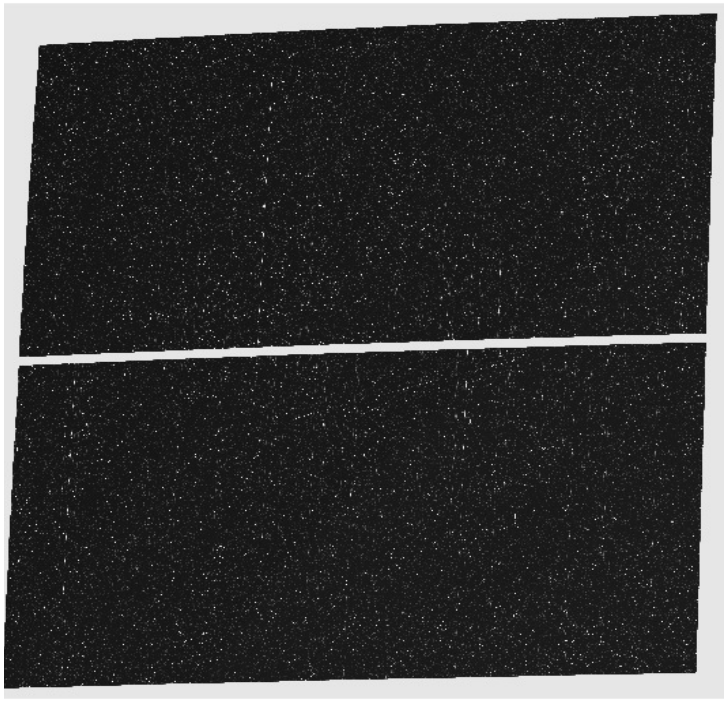
One of the singly drizzled FLT images, 'f555w_posa_01_flt_single_sci.fits', is shown in Figure 6.6. This image still contains numerous cosmic ray events, hot pixels, and other artifacts. The 'rhombus' shape is a result of correcting the geometric distortion. The corresponding weight image, 'f555w_posa_01_flt_single_wht.fits', is shown in Figure 6.7, where white indicates pixels with zero weight. Due to the effects of distortion and varying pixel area in the '_flt.fits' images, the weight image changes gradually by ~10% across the detector. Because the association table was used to define a common WCS for all images, the drizzled image is 'padded' with zeros outside the boundary of the original array. The weight image is set to zero in these regions, allowing these pixels to be rejected during median combination.

Figure 6.6: Single ACS Science Image



The single science image.

Figure 6.7: Single ACS Weight Image



The weight image for the single science image.

Next, the user should measure the positions of stars in the separately drizzled images, and compute a shift file which defines the residual offsets between images which were not accounted for using the header WCS. This process is described in detail in Example 2 and will not be addressed in this example. After separately drizzling, the '_single_sci.fits' images at POSB are offset from the first image at POSA by approximately (-20,+19) pixels. The tolerance for matching has been set to 2 pixels in this example; however, if the images were obtained at different orientations, small residual rotations on the order of 0.005 degrees may also exist. In this case, the tolerance should be set slightly higher, for example 10 pixels, to ensure that stars are matched.

The resulting matched coordinate lists contain ~6K stars for POSA and ~2K stars for POSB. The fewer stars matched for POSB are due to the fact that there is only a small overlap between the two pointings.

The residual offsets were recorded to the nearest 0.01 pixel and are listed below in the form of a shift file. The transformation results for this example using 'fitgeom=rscale' show no significant rotation or scale terms (i.e. the largest residual rotation is less than 0.001 degrees and the largest scale correction, 1.000009, times the detector size, 4096 pixels, is less than 0.1 pixels). Thus, the 'geomap' steps described above may be rerun, but with 'fitgeom=shift' to obtain slightly more accurate results. (Note that the fit rms given in the geomap output is approximately the same (0.1 pixels) whether 'shift' or 'rscale' was used. If indeed a residual rotation or scale was required, the fit rms would be significantly smaller when 'fitgeom' was set to 'rscale' compared to when it was set to 'shift'.)

The ascii file below is called 'shift.dat' and contains the residual offsets which were derived by the user using 'geomap'. The columns are as follows: image, x_resid, y_resid, rotation, scale. In this case, no significant rotation or scale terms were found, so these columns have been left blank. For observations which are taken in different visits (and therefore at different orientations), residual rotations and scales are usually required.

```
# units: pixels
# form: delta
# frame: output
# refimage: refimage.fits
f555w_posa_01_flt.fits   0.00   0.00
f555w_posa_02_flt.fits  -0.05  -0.05
f555w_posa_03_flt.fits   0.10   0.08
f555w_posa_04_flt.fits   0.02   0.06
f555w_posb_01_flt.fits -19.99  18.63
f555w_posb_02_flt.fits -19.97  18.65
f555w_posb_03_flt.fits -20.00  18.69
f555w_posb_04_flt.fits -19.98  18.70
```

Before moving on to further processing steps, the single drizzled image should be copied to a file called 'refimage.fits'. The residual offsets given in the shift file are defined with respect to this reference image.

```
imcopy f555w_posa_01_single_sci.fits refimage.fits
```

6.2.4.2 A Quick-way to Produce a Combined Image

All the necessary information for successfully combining the data should be in hand at this point. The 'shiftfile' provides the corrections to the alignment of the images to allow the rest of the MultiDrizzle processing to correctly identify and remove cosmic rays. The final combined image can be generated using one more call to MultiDrizzle, this time turning on all the steps:

```
unlearn multidrizzle
multidrizzle *_flt.fits output=f555w shiftfile='shift.dat'
```

The resulting product will be the final combined image named, in this case, 'f555w_drz.fits' which can now be used for science. Because the default MultiDrizzle parameters may not provide optimal data products, and the user is strongly advised to inspect the quality of the sky subtraction and cosmic ray rejection and to experiment with the output 'scale' and 'pixfrac' parameters for the final drizzle step. The following sections will walk the user through each of the remaining processing steps.

6.2.4.3 Creating the Median

Once the appropriate offsets have been determined, MultiDrizzle should be rerun from the beginning with only steps 1-4 (static mask, sky subtraction, single drizzle, and median) turned on and with the 'shiftfile' parameter pointing to the ascii file containing the residual offsets. MultiDrizzle will use the header WCS information in combination with the shift file to appropriately align images.

When 'static=yes', MultiDrizzle goes through each of the input images, calculates the rms value for each chip, and identifies pixels that are below the median value by more than 'static_sig' times the rms. This step is used for identifying pixels that may have high values in the dark frame, which is subtracted during calibration, but may not necessarily have high values in the images, and thus subtraction gives them large negative values. Such pixels are not always flagged in the DQ file, and this step allows them to be identified. Sometimes such pixels fall on bright objects so instead of being negative, they would be positive but lower than surrounding pixels. If the images are dithered, then they should land on blank sky at least some of the time, in which case they will appear negative and will be flagged.

When 'skysub=yes', MultiDrizzle will subtract the sky from each drizzled exposure. Since the input images have different exposure times, it is important to remove the sky prior to creation of the median image. Otherwise, deviant pixels will not be optimally rejected from the median, and this will impact the accuracy of the cosmic ray rejection which is performed in step 5 and which compares each input pixel with the blotted median image. Sky subtraction is recommended for effective cosmic ray flagging and removal, but only if sufficient blank sky is available to perform an accurate determination. Great care must be taken when choosing to implement sky subtraction, because if blank sky is not available, sky subtraction will produce erroneous results. Two important parameters to consider are the upper and lower values for data that will be used to estimate the sky value. These should be set to include most pixels in the sky (so substantially more than the FWHM of the sky

distribution) but not so large as to include a substantial amount of power from objects or cosmic rays. For more information, refer to Section 4.6 of the ACS Data Handbook.

During pipeline processing, the header keyword MDRIZSKY is estimated using the default MultiDrizzle parameters and is then written to the image header in the science extensions [1] and [4] of the '*_flt.fits' image. The sky is not subtracted from the science array pixel values. If the user wishes instead to compute the sky using an alternate method, the parameter 'skyuser' can be set to point to a keyword in the image header which gives the user's sky value. MultiDrizzle will assume that this sky value has already been removed from the '*_flt.fits' images prior to processing. The value set by the parameter 'skyuser' is then copied to MDRIZSKY for computing statistics, but it is not actually subtracted from the drizzled image since it assumes this has already been done.

The sky background is calculated independently for each of the two chips, and the lowest value is taken to represent the true sky value for both chips. Since the main body of NGC4449 is centered on the lower chip, the sky value from the upper chip has been selected by MultiDrizzle for populating the header keyword MDRIZSKY in the '*_flt.fits' image header. MultiDrizzle will subtract the sky from the single drizzled prior to the creating the median. Note that the pixel values in the '*_flt.fits' images themselves are not modified when MultiDrizzle completes.

When 'median=yes', MultiDrizzle creates a median image from the separate drizzled input images, allowing a variety of combination and rejection schemes. If 'combine_type' is set to 'median' or 'average', then the routine carries out a similar calculation to the standard IRAF task 'imcombine', with equivalent behavior for the parameters 'combine_nlow' and 'combine_nhigh' (the number of low and high pixels to reject), and 'combine_grow' (the amount by which flagged pixels can grow). All imcombine parameters other than those specified above are reset to their default values.

If 'combine_type=minmed', a more sophisticated algorithm is used to combine images. The basic concept is that each pixel in the output combined image will be either the median or the minimum of the input pixel values, depending on whether the median is above the minimum by more than n times sigma. An estimate of the "true" counts is obtained from the median image (after rejecting the highest-valued pixel), while the minimum is actually the minimum unmasked ("good") pixel. This algorithm is designed to perform optimally in the case of combining only a few images (3 or 4), where triple-incidence cosmic rays often pose a serious problem for more simplified median combination strategies. The 'minmed' algorithm performs the following steps:

a.) Create median image, rejecting the highest pixel and applying masks.

b.) Use this median to estimate the true counts, and thus derive an rms.

c.) If the median is above the lowest pixel value by less than the first value in 'combine_nsigma', then use the median value, otherwise use the lowest value.

d.) If 'combine_grow' > 0, repeat the above 3 steps for all pixels around those that have already been chosen as the minimum, this time using a lower significance threshold specified as the second value in 'combine_nsigma'.

The last step is very successful at flagging the lower signal-to-noise "halos" around bright cosmic rays which were flagged in the first pass.

If ‘median_newmasks=yes’, then the singly drizzled weight maps (‘*_single_wht.fits’) are used to create pixel masks for each image, based on the pixel flag information originally present in the DQ arrays. These masks will then be used when combining images, in order to prevent bad pixels from adversely affecting the median. If ‘median_newmasks=no’, MultiDrizzle will use masks which have been created offline by the user and which are specified via the ‘BPM’ header keyword in each image.

The median image has been generated by running MultiDrizzle steps 1-4 in succession on the ‘*_flt.fits’ images. The eight separately drizzled images are combined using the bad pixel masks and the rejection parameters specified below to create a single clean median image named ‘f555w_med.fits’. Note that by rerunning step 3, new versions of the single drizzled images have been created, and these will now be correctly aligned. These separate images will be combined to create the median image.

```
multidrizzle *_flt.fits output=f555w shiftfile='shift.dat' \
     static+  skysub+  driz_separate+ driz_sep_fillval=99999 \
     driz_sep_bits=4192 median+ combine_type=minmed \
     combine_nhigh=0 combine_hthresh=99990
```

Selecting the best parameters for the median step can be an iterative process and should always involve examination of the clean, combined product to verify that the majority of cosmic rays and other artifacts are successfully removed. The rejection algorithm which is ultimately chosen depends largely on the number of datasets being combined and the amount of overlap between dithered images. In this example, the parameter ’driz_sep_fillval’ has been set to a very large number (99999) and the median parameter ’combine_hthresh’ has been chosen so that pixels with no valid input are rejected prior to median combination. The parameters ’combine_type=minmed’ and ’combine_nhigh=0’ were selected because the left and right halves of the mosaic have only 4 input images contributing, and because the area containing the WFC chip gap has only 2 input images. When a large number of input images are available, the parameter ’combine_type’ may be set to ’median’, and this will save the user excess computation time.

When examining the median image, the width and shape of the PSF should be inspected over the entire field of view to confirm the quality of the image registration. A PSF which is ’round’ and ’narrow’ is a sign that the alignment was successful. To confirm that no residual offsets or rotations remain, the position of stars and the geometric transformations should again be examined for the new separately drizzled images (using the steps outlined in Example 2). The median combination may require some ’tweaking’ of the rejection parameters to produce a clean result. To avoid overwriting the median image when ’fine-tuning’ the parameters, the user should copy the image to a unique filename before rerunning MultiDrizzle. The median image created from each test run should be carefully inspected and compared to verify which parameters work best at removing cosmic rays, hot pixels, and other artifacts.

```
imcopy f555w_med.fits f555w_med_test1.fits
```

6.2.4.4 Computing the Cosmic Ray Masks

Turning on steps 5 and 6 (blot and driz_cr), MultiDrizzle will transform the median image back to the distorted reference frame, and it will compare the original and the blotted median images in order to identify cosmic rays. For optimal flagging, sky subtraction should be performed when the background is more than a few electrons. Images obtained from the archive have already been run through MultiDrizzle using a default set of parameters, and these pixels will be flagged with a value of 4096 in the data quality (DQ) array of the '*_flt.fits' images. Because the default values may not be optimal for every dataset, the user is encouraged to fine-tune the cosmic ray rejection parameters 'driz_cr_snr' and 'driz_cr_scale', especially if the images were not obtained using a pre-defined dither pattern.

When 'blot=yes', the median image is transformed back ('reverse drizzled') to the reference frame of each original input image. This is done by backing out the shifts/rotations which were applied in step 3 and by applying an inverse distortion correction. The median image is resampled to the pixel scale of the original images and is trimmed to match the dimensions of each input image. The blotted frames are created for each chip and are named '*_sci?_blt.fits'. If desired, the user may wish to display the input images and blink them with their 'blotted' counterparts. The 'blotted' images should align perfectly with their respective input images and should be similar in appearance, except for being cleaned of cosmic rays and other defects.

When 'driz_cr=yes', MultiDrizzle uses the original input images, the blotted median image, and the derivative of the blotted image (created with the 'deriv' task) to create a cosmic ray mask for each input image. The 'deriv' task 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 saved to a temporary image, '*_sci?_bl_deriv.fits', which represents an effective gradient or spatial derivative. These derivative images are used by 'driz_cr' 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*deriv_image + SNR*noise
```

where 'scale' is defined as the multiplicative factor applied to the derivative which 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.

MultiDrizzle is not designed to be restarted in the middle of processing, so when computing improved cosmic ray masks, the user must turn on all steps prior to 'driz_cr'. If this is not done, the header keyword MDRIZSKY will be set to zero, and the software will flag any reasonably bright objects in the image. The result is a cosmic ray mask that looks like a map of the image, where all pixels containing sources have been flagged. The following command illustrates how to reproduce the cosmic ray rejection performed in the pipeline.

```
multidrizzle *_flt.fits output=f555w shiftfile='shift.dat'\
   static+ skysub+ driz_separate+ driz_sep_fillval=99999\
   driz_sep_bits=4192 median+ combine_type=minmed \
   combine_nhigh=0 combine_hthresh=99990 blot+ driz_cr+ \
   driz_cr_corr+ driz_cr_snr='3.5 3.0' driz_cr_scale='1.2 0.7' \
   crbit=0
```

When 'driz_cr_corr'=yes, the task will create both a cosmic ray mask image ('*_sci?_crmask.fits') and a clean version of the original input images ('*_sci?_cor.fits') where flagged pixels are replaced with pixels from the blotted median. The cosmic ray masks are multiplied by the bad pixel masks from step 1 (which are a combination of the image DQ array and the static masks) to create a final mask for each image. The optional parameter 'crbit' tells MultiDrizzle which flag value to assign to cosmic rays, and this flag will be written to the DQ array of each input image. If this parameter is left blank during reprocessing, the default value of 4096 is used. While experimenting with the 'driz_cr' parameters, the user can tell MultiDrizzle not to update the DQ arrays by setting this parameter equal to zero. Then, once the user is happy with the cosmic ray masks, the 'crbit' may be assigned a unique value, for example, 8192.

The quality of the cosmic ray masks should be verified by blinking the original input image with the cosmic ray mask. For comparison, the user may also blink the cosmic rays flagged by the pipeline. This may be done by displaying only pixels in the DQ array with a value greater than 4096. When testing various parameter values, the user may find that the centers of stars are being unnecessarily masked, and this indicates that the 'driz_cr_scale' parameter should be increased. If not enough cosmic rays are being detected, this parameter should be decreased.

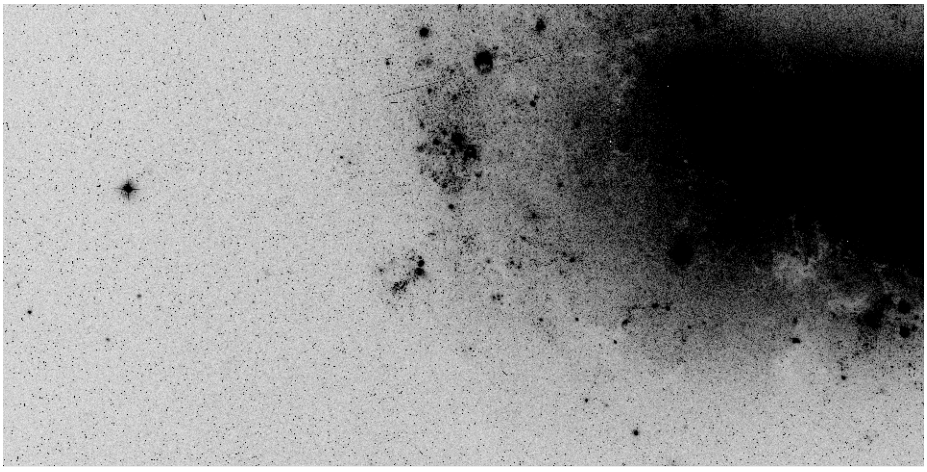
```
displ f555w_posa_01_flt.fits[1] 1 \
      zs- zr- z1=0 z2=1000 ztran=log

displ f555w_posa_01_sci1_crmask.fits 2 \
      zs- zr- z1=0 z2=1

displ f555w_posa_01_flt.fits[3] 3\
      zs- zr- z1=4000 z2=5000
```

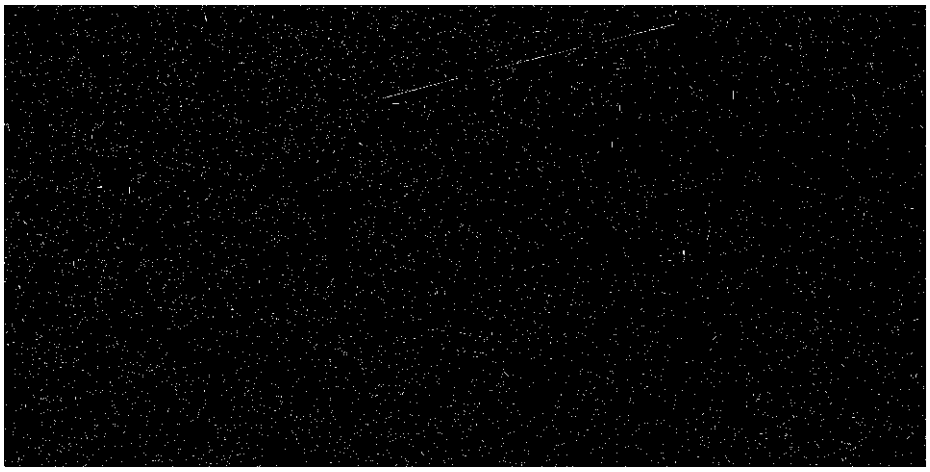
In the figures below, chip 2 of the original input image j8cw54orq_flt.fits[sci,1] (Figure 6.8) is presented with its corresponding cosmic ray mask(Figure 6.9). Note that a mask value of 1 (shown in black) corresponds to good pixels, while a mask value of 0 (white) corresponds to pixels flagged as cosmic rays.

Figure 6.8: Single Chip from ACS FLT Image



The FLT image.

Figure 6.9: Cosmic Ray Mask from Single ACS FLT Image



The cosmic ray mask.

For these observations, the default 'driz_cr_snr' values "3.5 3.0" are too stringent and cause numerous single pixels to be flagged where the background noise is high. True cosmic ray impacts are usually more spatially extended, so the 'driz_cr_snr' parameter has been increased to "5.0 4.0" to create more conservative masks. This approach was chosen because only 4 images go into making each half of the final mosaic. Thus if too many input pixels are flagged, the final image may be degraded. Large rms deviations in the background noise will be 'beat down' when the images are combined in the final drizzle step.

Once the optimal cosmic ray rejection parameters have been determined, the user should rerun the MultiDrizzle with the 'crbit' set to a unique value, 8192, for example. In this way, the DQ arrays will be updated with the user-defined masks. Then, during the final drizzle step, the 'driz_final_bits' parameter can be set so that MultiDrizzle will ignore the erroneous 4096 DQ values flagged during pipeline processing.

```
multidrizzle input="*_flt.fits" output=f555w \
   shiftfile='shift.dat' static+ skysub+ driz_sep+ \
   driz_sep_fillval=99999 driz_sep_bits=4192 \
   combine_type=minmed median+ driz_comb+ blot+ driz_cr+
   driz_cr_corr+ driz_cr_snr='5.0 4.0' combine_nhigh=0\
   combine_hthresh=99999 driz_cr_scale='1.2 0.7' crbit=8192
```

6.2.4.5 Final Drizzle

Once the residual shifts and improved cosmic ray masks have been computed, the final drizzled combined mosaic may be created by running step 7 only and turning off all intermediate processing steps. The sky background is NOT removed in this example, following the recommendation from the DAOPHOT Reference Guide by Lindsey Davis: "The sky background should NOT be subtracted from imaging prior to photometry. DAOPHOT fitting routines use an optimal weighting scheme which depends on the readnoise, gain, and true counts in each pixel. If the mean sky has been subtracted, the computed weights will be incorrect". Depending on the user's science needs, the sky background can be left in or removed by setting the 'skysub' parameter appropriately.

```
multidrizzle output='f555w'  final_bits=4192 \
   shiftfile='shift.dat'  static- skysub- driz_sep- \
   median- blot-  driz_cr- driz_comb+ final_rot=INDEF
```

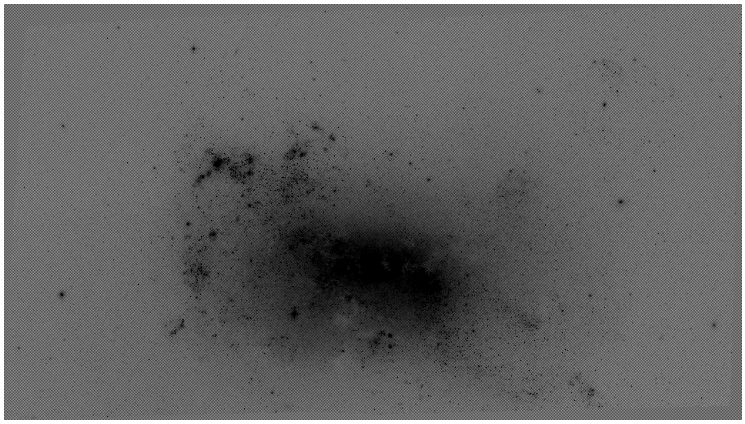
When 'driz_combine=yes', this step takes the original input images, together with the final masks, and drizzles them onto a single output image. The standard drizzle parameters 'final_kernel', 'final_scale', 'final_pixfrac', and 'final_rot' can be specified by the user, if desired. By default the scale of the output WFC image is 0.05"/pixel, but the user is encouraged to experiment with other options (e.g. shrinking the 'final_scale' and 'final_pixfrac' to yield a sharper output PSF.)

The 'bits' parameter is defined as the integer sum of all bit values from the input image DQ array that should be considered 'good' when building the weight mask. Because MultiDrizzle was designed for use with multiple instruments, the default value for the 'bits' parameter and is set to zero both in step 3 ('driz_sep_bits') and in step 7 ('final_bits'). For ACS data, the pipeline uses the value 96 which is specified in the MDRIZTAB reference file and which tells MultiDrizzle to ignore data quality flags 32 and 64. These flags were set by CALACS and correspond to CTE tails of hot pixels in superdark DQ arrays and warm pixels in superdark DQ arrays. For more information on selecting the appropriate bits for your data, refer to Section 5.5.7. Information from the DQ array for each chip is used in combination with the 'bits' parameter to create temporary mask files for each chip, where pixels which were flagged in the DQ array and which were not specified as good are assigned a value of zero in the mask. The 'final_bits' value in this example is 4192, the sum of 4096, 32, and 64. This tells MultiDrizzle to ignore pixels which were flagged with a value of 4096 during pipeline processing and to treat them as 'good' pixels.

The first extension of the drizzled mosaic 'f555w_drz.fits[1]' contains the science (SCI) image which has been corrected for distortion which represents the combination of all eight dithered images. All pixels cover an equal area on the sky and have an

equal photometric normalization across the field of view, giving an image which is photometrically and astrometrically accurate for both point and extended sources. The dimensions of the output image are computed on-the-fly by MultiDrizzle and the default output plate scale is read from the 'scale' column in the IDCTAB. These parameters, however, may be chosen by the user to best suit the actual data. In this example, the size of the output image is ~7700x4340 pixels. The SCI image is presented in Figure 6.10 and is in units of 'electrons/sec'. Changing the 'final_units' parameter from the default value 'cps' (counts per second) to 'counts' will produce a drizzled image in units of electrons.

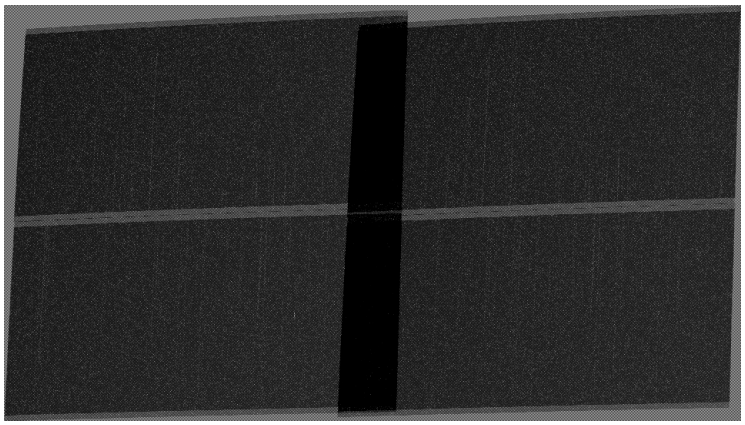
Figure 6.10: The Final ACS Science Image



The final science image.

The second extension of the output image contains the weight (WHT) image. When the 'final_wht_type' is set to 'EXP', the weight image can be considered an effective exposure time map. The weight image from our example is shown in Figure 6.11, where darker areas have higher weight. The chip gaps are clearly visible, as are column defects and cosmic ray features. The center of the image is "black" and corresponds to the overlap of all eight images. The weight in this region is ~4700 which is approximately equal to the sum of the exposure times of all eight images. The majority of the weight image is "dark gray" and corresponds to the overlap of the four input images at each position (A and B). In this region the weight is ~2400. Note that there is a smooth variation of approximately 10% across both the left and right halves mosaic which is due to the variation of the pixel area on the sky caused by the distortion. This variation cannot be seen in the figure, but is clearly present when performing statistics on different regions of the weight image. Finally, the "light grey" regions correspond to a weight of ~1200 where only two images contribute to the final product, for example in the chip gap Figure 6.11.

Figure 6.11: The Final ACS Weight Image



The final weight image.

The output weight image can also be specified to be in units of inverse variance by setting 'final_wht_type = ERR'. In this case, the weight is calculated using the error arrays in the second and fifth extensions of the '*_flt.fits' files. The error arrays include all sources of error, including read-noise, dark current, as well as the sky background and all the source.

Finally, if 'final_wht_type = IVM', MultiDrizzle will look for inverse variance files provided by the user (for example, to include all sources of noise except those related to photons from objects in the field). These would be specified by giving all the input '*_flt.fits' files to MultiDrizzle in an ASCII file list, with each line containing just two filenames, namely the name of the '*_flt.fits' file and its corresponding IVM file. The IVM file should have a similar structure as the '*_flt.fits' file, except that the inverse variance values are stored in an extension that is called [IVM].

6.2.4.6 Optimizing Scale and Pixfrac for Subsampled Data

To determine the optimal drizzle parameters for the final mosaic, the user is advised to experiment with different combinations of the parameters 'final_scale' (the size of the output pixels) and 'final_pixfrac' (the linear size of the 'drop' in input pixels). One must choose a pixfrac value that is small enough to avoid degrading the final image, but large enough that when all images are dropped onto the final frame, the flux coverage of the output frame is fairly uniform. As suggested in the HST Dither Handbook, statistics performed in different regions of the drizzled weight image should yield an rms value which is less than 20% of the median value. In general, the 'pixfrac' should be slightly larger than the scale to allow some spill over to adjacent pixels.

For bookkeeping purposes, it can be helpful to rename the final drizzled product to reflect the final drizzle parameters. Then, different versions of the drizzled science and weight images may be compared directly for the purpose of selecting the best parameters. Following the guidelines described above, a final scale equal to 0.7 times the default pixel scale (0.05"/pixel) and a final pixfrac of 0.8 provides the best sampling for these dithered data. (For more discussion, refer to ACS Example 1.) The

plate scale of the resulting mosaic is 0.035"/pixel and the FWHM of the PSF (in arcseconds) is narrower than in the drizzled pipeline product.

```
multidrizzle output='f555w'  final_bits=4192 \
   shiftfile='shift.dat' static- skysub- driz_sep- median+ \
   blot- driz_cr- driz_comb+ final_scale=0.035 \
   final_pixfrac=0.8 final_rot=INDEF

imrename f555w_drz.fits f555_drz_sc07_px08.fits
```

6.3 NICMOS

6.3.1 Introduction

This example describes the combination of NICMOS data from program 7115 (PI: Dean Hines), a series of NIC2 observations of the CRL 2688, the Egg Nebula. In this example, we will generate drizzled images of the Egg Nebula as seen using the F165M, F190N, and F212N filters.

Prior to attempting this example, we assume that users have worked through the examples in the previous sections and are familiar with using [association tables] and [shift files]. NICMOS observations may be taken independently or specified as an associated dataset - in which case an ASN table will be delivered with the data from the archive. When requesting the data for proposal 7115, you will automatically obtain a set of ASN tables associating several datasets together. In this particular example however, these ASN tables contain no information about offsets and rotations between observations. We will describe how to determine these in the following section. The table below replicates the default information which is available in the ASN table n3uv01020_asn.fits. It is also possible to create your own association table for datasets you would like to combine, or merely specify a unique rootname to MultiDrizzle and the software will create the table.

Table 6.5: Default ASN table of the F190N images used in this examples

MEMNAME	MEMTYPE	MEMPRSNT
N3UV01ALQ	EXP-DTH	yes
N3UV01AMQ	EXP-DTH	yes
N3UV01ANQ	EXP-DTH	yes
N3UV01AOQ	EXP-DTH	yes
N3UV01020	PROD-TARG	yes

6.3.2 NICMOS Specific issues

There are several properties of NICMOS data which can affect the proper working of MultiDrizzle. In particular, NICMOS images are subject to a several additive signals present in raw NICMOS data, such as the Pedestal effect, which is dealt by the NICMOS calibration pipeline. These corrections however can be less than optimal and result in a significant amount of residuals in pipeline calibrated NICMOS data. Running MultiDrizzle on NICMOS data where these effects have not been totally removed will result in an improper background determination, cosmic ray removal for example. Users are encouraged to individually check the background levels of their pipeline calibrated NICMOS images to ensure that effects such as Pedestal have been properly removed before proceeding with MultiDrizzle.

6.3.3 Initial Setup

In this example, it is assumed that the user has requested and obtained all files and calibration products from the HST archive. While the geometrical distortions in NICMOS are small, the best results will only be achieved when these are taken into account by MultiDrizzle. The NICMOS IDCTAB files can be obtained from the NICMOS instrument Web pages or the HST data archive, or can be downloaded together with the calibrated data from the HST Archive. The following is a list of all the available CAL files for program 7115 in all filters (all are taken using the NIC2 detector):

```
n3uv01alr_cal.fits
n3uv01amr_cal.fits
n3uv01anr_cal.fits
n3uv01aor_cal.fits
n3uv01arr_cal.fits
n3uv01asr_cal.fits
n3uv01atr_cal.fits
n3uv01avr_cal.fits
n3uv01awr_cal.fits
n3uv01axr_cal.fits
n3uv01ayr_cal.fits
```

```
n3uv01b1r_cal.fits
n3uv01b2r_cal.fits
n3uv01b3r_cal.fits
n3uv01b4r_cal.fits
n3uv01b6r_cal.fits
n3uv01b7r_cal.fits
n3uv01b8r_cal.fits
n3uv01b9r_cal.fits
n3uv01bbr_cal.fits
n3uv01bcr_cal.fits
n3uv01bdr_cal.fits
n3uv01ben_cal.fits
n3uv01bgm_cal.fits
n3uv01bhm_cal.fits
n3uv01bim_cal.fits
n3uv01bkr_cal.fits
n3uv01blr_cal.fits
```

The appropriate calibrated science files along with the associated calibration reference files should be placed in the user's local disk area. These should minimally include:

- the geometric distortion reference table (IDCTAB)
- any additional static bad pixel masks the user might want to use. This will be addressed later in this document.
- if you don't have a local copy of the HST CDBS, then a copy of the calibration reference files used for the images

Next, make sure the 'nref' directory is defined in your OS environment shell. Start PyRAF, and load the dither package by typing "stsdas" and then "dither". To run MultiDrizzle, the parameters may be edited in the standard way using the "epar" facility. We recommend that beginners use the 'epar facility' to become familiar with all steps and parameters before running from the command line. MultiDrizzle software has an extensive set of parameters, but the default values should allow the task to process nearly any set of images for an initial review. The parameters are separated according to the processing step they control, making it easier to interpret them.

Next, we need to identify and create list of datasets taken with the same filters. This can be done in Pyraf using, for example:

```
hselect *cal.fits[0] $I,TARGNAME,CAMERA,FILTER yes > all.lst
```

The content of the file all.lst can then be used to generate lists of datasets in each filter. For example using (the ! is necessary if this is done within iraf or Pyraf):

```
!awk '$2=="CRL2688" && $3==2 && $4=="F190N" \
    {print substr($1,0,18)}' all.lst > F190N_NIC2.lst
!awk '$2=="CRL2688" && $3==2 && $4=="F165M" \
    {print substr($1,0,18)}' all.lst > F165M_NIC2.lst
!awk '$2=="CRL2688" && $3==2 && $4=="F212N" \
    {print substr($1,0,18)}' all.lst > F212N_NIC2.lst
```

In this example, we describe the parameters for each step in succession, though in practice, the user would set all relevant parameters at once. While the majority of relevant parameters are discussed here, a help document describing all parameters and tasks can be accessed by typing 'help MultiDrizzle' from within PyRAF. Note that running MultiDrizzle on a modern computer with NICMOS data is not very time consuming. While we sometimes will need to only examine intermediate MultiDrizzle products, such as when generating the offset corrections, or creating a cosmic ray free median image, we sometimes just run the entire MultiDrizzle process. This limits the number of times we have to reset the various parts of MultiDrizzle.

6.3.4 The Default Parameters

The default MultiDrizzle parameters usually produce satisfactory results. It is recommended to first run MultiDrizzle on NICMOS data with default parameters and to them slowly modify these to improve the qualify of the end product. The main limiting factors in NICMOS data is that these data sometime suffer from a significant amount of amp glow near the corners of an exposure and/or of the `Pedestal` effect. This causes the sky background of NICMOS data to be quadrant dependent and non uniform and can lead so some problems when combining these images together (such as inconsistent sky across the image). It is recommended that in these cases, users pay particular attention to manually correcting for the Pedestal effect and amp glow. More information about these and how to correct them can be found in the NICMOS Data Handbook. Sky estimation can also be thrown off by crowded fields or bright objects. Datasets such as these should be closely inspected to ensure the correct sky correction has been made for each dataset before the drizzling process has started.

The only required parameter is the rootname for the 'output' drizzled product. By default, MultiDrizzle will look for all files in the working directory with the FLT extension. However, this is specific for ACS processed data. The first calibrated output from the NICMOS calibration pipeline are the *_CAL.FITS files. At a minimum the user should start with these. If additional processing, such as pedestal removal, South Atlantic Anomaly (SAA) correction has occurred, then the appropriate extension for the final calibrated images should be supplied. The user may modify the 'suffix' parameter to include some other extension or may specify a subset of images via the 'filelist' parameter. Using the images specified, MultiDrizzle calls PyDrizzle which uses the buildAsn task to create an association table named 'output_asn.fits'. This association will be used to define the dataset. The header WCS information from the entire set is used to define a common WCS output frame, and MultiDrizzle sets the drizzle parameters appropriately. If user defined shifts are available, these may be specified in the 'shiftfile' parameter, and the association table will be updated accordingly. However, a shift file is not usually available until after step 3, separately drizzling the images onto a common WCS, has been performed and objects matched.

A reference image which has the desired output WCS may be specified, and the input images will be centered to match the WCS of this image. Alternately, the central RA and Dec (ra, dec) of the reference pixel and the dimensions of the output frame (outnx, outny) may be specified, if desired, though reasonable values will be

automatically determined from the images' WCS if these parameters are left blank. While the central RA and Dec are specified in the initial setup parameters listed above, the output image dimensions are specified in both the 'driz_separate' and 'driz_combine' parameters in steps 3 and 7, respectively.

The 'crbits' parameter is defined as the integer sum of all bit values from the input images' DQ array that should be considered 'good' when building the weighting mask. Because MultiDrizzle was designed for use with multiple instruments, the default value is set to zero. For NICMOS data, the recommended default value is SATURA. Information from the DQ array for each chip is used in combination with the 'bits' parameter to create temporary mask files called '*_mask1.fits'. Pixels which were flagged in the DQ array and which were not specified as good via the bits parameter are assigned the value 0. All other pixels are set to 1 in the mask. The following table summarizes the values in the DQ array for NICMOS:

Table 6.6: NICMOS Data Quality (DQ) Flags

Name	Value	Description
REED_SOL	1	Reed-Solomon decoding error
BAD_LIN	2	Poor linearity correction
BAD_DARK	4	Poor dark correction
BAD_FLAT	8	Poor flat field correction
GROT	16	Pixel affected by "grot"
DEFECTIVE	32	Hot or cold pixel
SATURATED	64	Saturated pixel
MISSING	128	Missing data (telemetry dropout)
BADPIX	256	Bad pixel set during calibration
CR_HIT	512	cosmic ray hit
SOURCE	1024	Pixel contains source
ZEROSIG	2048	Zero read signal correction
USER1	4096	User flag value 1
USER2	8192	User flag value 2
HIGH_CURVATURE	16384	High curvature in ramp
RESERVED2	32768	Reserved flag value 2

The distortion reference file is read from the image header via the IDCTAB keyword which specifies the name and location of the appropriate file. The distortion coefficients for each chip are written to temporary ascii files named '*_coeffs?.dat'. If the user wishes to retain these files, the parameter 'clean' should be set to 'no'.

In general, the default parameters will work well for most data. When setting up this example, we have specified only the following non-default parameters:

- output=F190N
- crbit=8578
- driz_cr_snr=4.0 3.0
- input=@F190N.NIC2.lst
- output=F190N

The choice of these last two parameters is explained in the sections that follow. In default mode, MultiDrizzle performs each of its 7 steps in order. In this example, however, we perform some of the steps and examine the intermediate products before final drizzle combination is performed. Approximately 1GB of free disk space is required for this example when intermediate products are not removed. An outline of the entire process is described below:

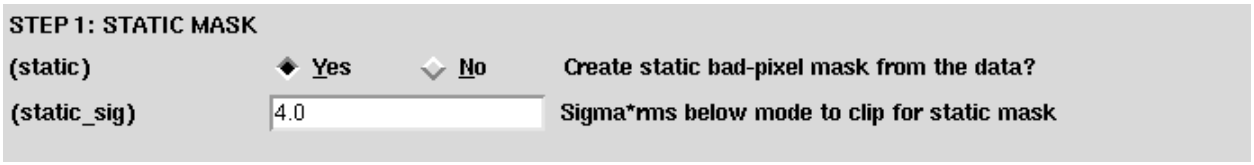
1. Run MultiDrizzle to create sky-subtracted, separately drizzled images (*single_sci.fits) which are based on a common WCS. (MultiDrizzle Steps 1 through 3)
2. Measure the positions of stars, or cross correlate images generated above (* single_sci.fits), to derive a residual delta shift file (shiftfile) which defines the residual offsets.
3. Run Mutlidrizzle again, now using the shiftfile created above, to create new separately drizzled images as well as a well-aligned median image of the field. MultiDrizzle Step 5 through 7 can be turned off at this stage to save processing time.
4. Examine the median image to ensure that images were properly combined. The median image should contain as little artifacts as possible. Also, if the alignment of the individual images has been properly corrected, stellar PSFs should be ‘round’ and ‘narrow’.
5. If the quality of the median image is satisfactory, and if Step 5 through 7 were turned off above, run MultiDrizzle with steps 5 through 7 turned on. This will transform the median image back to the reference frame of each of the original input images and derive cosmic ray masks. Using these new masks, perform the final drizzle combination and produce the final drizzled image.

The steps outlined above should be followed in an iterative manner, tweaking the necessary parameters until a given intermediate product reaches the expected quality (e.g. median image). The process can be safely be run several times over the same input files. Be warned however, than the bad pixel maps of the input CAL files is modified and that pixels flagged as CR by MultiDrizzle are updated in the DQ extension of the input CAL files. Users might hence want to always run MultiDrizzle on fresh, unmodified, copies of the input CAL files.

While above we refer to running only a subset of the MultiDrizzle steps (such as Step 1 through Step 3), the user can very well run the step every single time. While it is possible to turn off subsequent steps manually in MultiDrizzle, the processing of NICMOS data is not time consuming on a modern computer and not much time will be wasted if, while following this tutorial, you choose to run all MultiDrizzle steps every time you run MultiDrizzle. However, keep in mind that intermediate, or final products will be produced and that these might be useless, have artifacts etc. and will only become acceptable as every tweak of the parameters, shift between individual files, editing of bad pixel masks, etc. are done.

6.3.5 Step 1: Static Mask

Figure 6.12: Image of EPAR Frame for the First Drizzle Step

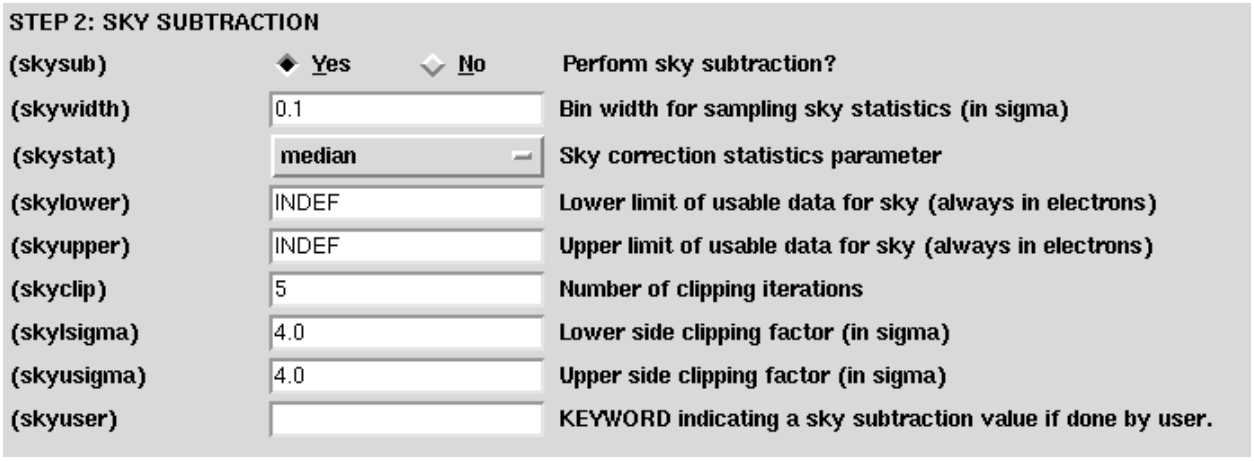


STEP1 in the process

When ‘static=yes’, this step goes through each of the input images, calculates the rms value for each chip, and identifies pixels that are below the median value by more than ‘static_sig’ times the rms. It is aimed at identifying pixels that may have high values in the dark frame, which is subtracted during calibration, but may not necessarily have high values in the images, and thus subtraction gives them strongly negative values. Such pixels are not always flagged in the DQ file, and this step allows them to be identified. Sometimes such pixels fall on bright objects so instead of being negative, they would be positive but lower than surrounding pixels. If the images are dithered, then they should land on blank sky at least some of the time, in which case they will appear negative and will be flagged.

6.3.6 Step 2: Sky Subtraction

Figure 6.13: Image of EPAR Frame for the Second Drizzle Step



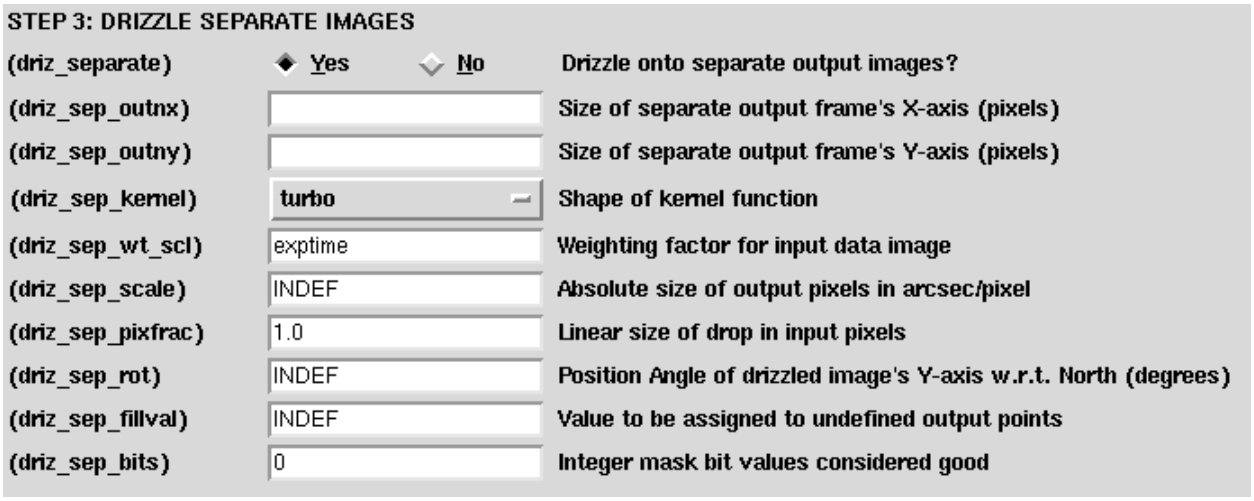
STEP 2 in the process.

When 'skysub=yes', this task will subtract the sky from each input exposure. Two important parameters to consider are the upper and lower values for data that will be used to estimate the sky value. These should be set to include most pixels in the sky (so substantially more than the FWHM of the sky distribution) but not so large as to include a substantial amount of power from objects or cosmic rays.

The 'SkySub' task will update the header keyword defined by the parameter 'skyname' with the derived sky value for each chip and will subtract the sky from the original exposures prior to the final drizzle combination. Note that the CAL images themselves are not modified when MultiDrizzle completes, although the sky value is stored in the images headers as the MDRIZSKY keyword. Sky subtraction is recommended for effective cosmic ray flagging and removal, but only if sufficient blank sky is available to perform an accurate determination. Great care must be taken when choosing to implement sky subtraction, because if sufficiently blank sky is not available, sky subtraction will produce erroneous results. In the case of the Egg Nebula images, adequate blank sky is available to allow an accurate sky determination.

6.3.7 Step 3: Drizzle separate images and Refining Offsets

Figure 6.14: Image of EPAR Frame for the Third Drizzle Step



STEP 3 in the process.

MultiDrizzle uses the world coordinate system (WCS) information in the image headers to align the images. While dithers applied to a target within a single visit of HST are usually accurately reflected by the WCS, this is not the case for multiple visits which normally require guide star re-acquisitions and may utilize different guide stars. Even for nominally back-to-back exposures, offsets large enough to degrade final combinations do sometimes occur. As a result, it is essential to accurately determine image-to-image shifts (delta shifts), and possibly rotations, before running MultiDrizzle. These (delta) shifts may be determined by separately drizzling each image onto a common WCS frame.

This is accomplished with the help of the products output by the 'driz_separate' task in step 3. Object lists derived for each drizzled image may then be matched and fit to derive a single shift for each image. Once the shifts have been determined, MultiDrizzle must be rerun from the beginning with the shift file specified in the 'shiftfile' parameter. MultiDrizzle will update the association table to reflect these 'delta' offsets and will now use both the header WCS information plus the shift file to appropriately align images.

In this example, we choose to compute these offsets using a cross correlation. Different approaches are possible to determine residual offsets between images however, such as manually measuring the position of sources in individual images.

```
crossdriz n3uv01anr_single_sci.fits \
      n3uv01anr_single_sci.fits cross1x1
crossdriz n3uv01amr_single_sci.fits \
      n3uv01anr_single_sci.fits cross1x2
crossdriz n3uv01alr_single_sci.fits \
      n3uv01anr_single_sci.fits cross1x3
crossdriz n3uv01aor_single_sci.fits \
      n3uv01anr_single_sci.fits cross1x4
```

Shifts are then computed from the cross* files generated above using the shiftfind program from the IRAF fitting package. Again, after loading the fitting package, this can be done with the following commands:

```
shiftfind cross1x* shifts.F190N.txt
```

The result if a simple text file, shifts.F190N.txt. The content of this file is shown below:

```
 n3uv01anr_single_sci.fits  0  0.0001 0.0490  -0.0003 0.0386  0.00
287.85
 n3uv01amr_single_sci.fits  0  0.1354 0.0497   0.1049 0.0385  0.00
287.85
 n3uv01alr_single_sci.fits  0  0.3194 0.0497  -0.0148 0.0381  0.00
287.85
 n3uv01aor_single_sci.fits  0  0.1395 0.0502  -0.1362 0.0402  0.00
287.85
```

The file generated by the shiftfind routine can be edited into a properly formatted MultiDrizzle input shift file format, which we name shifts.F190N in this example:

```
# units: pixels
# frame: input
# form: delta
n3uv01anr 0.0001   -0.0003    0.00
n3uv01amr 0.1354    0.1049    0.00
n3uv01alr 0.3194   -0.0148    0.00
n3uv01aor 0.1395   -0.1362    0.00
```

In the example above, each line shows the dataset name, followed by the x and y delta offsets, in pixels, followed by the optional rotation in degree and which in this case we simply set to zero.

While we use the crossdriz and the shiftfind program to compute offsets between these exposures, users are free to use whatever means at their disposal to determine the relative offsets between singly drizzle images. In some cases, a stellar field for example, it might be more appropriate to carefully measure the position of individual stars in each frame for example. No matter what method is used, the end result, if the relative offsets are measured between singly drizzle images produce by MultiDrizzle, should be entered in a file similar to the file shifts.F190N shown above.

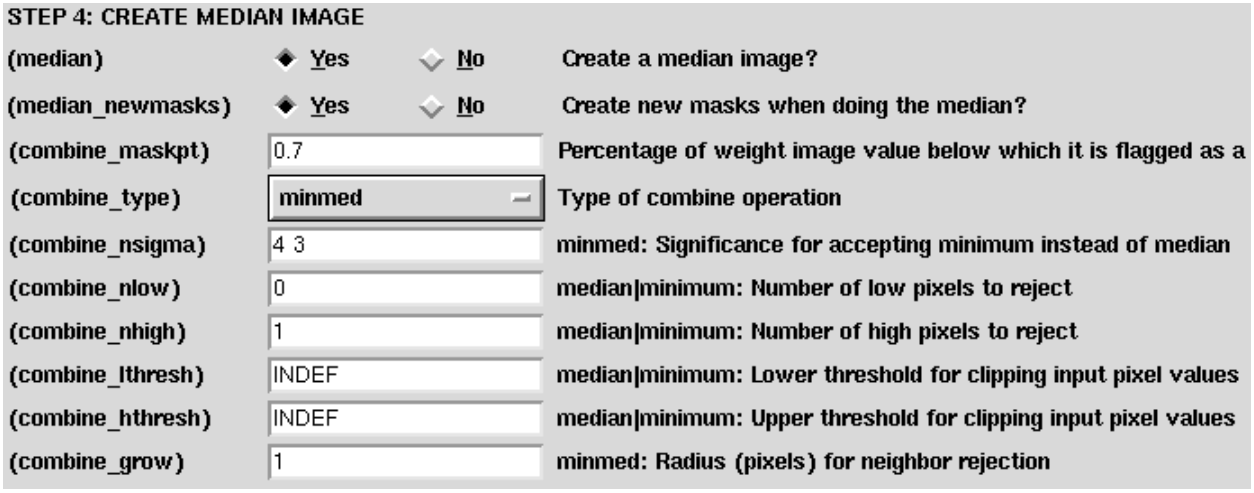
When the ‘driz_separate’ step is run for the second time, but with a ‘shiftfile’ specified, the association table for the data set will be automatically updated. To confirm that the new separately drizzled images are appropriately registered, cross correlation using crossdriz could again be examined. It is also useful to create a median image and examine the width and shape of the PSF (if stars are present in the image) over the entire FOV to look for any effects of mis-registration. Median combination is performed in Step 4.The content of the ASN table after running MultiDrizzle with the shiftfile shown above is:

Table 6.7: Default ASN table of the F190N NICMOS images used in example

MEM-NAME	MEM-TYPE	MEM-PRSNT	X OFFSET	Y OFFSET	X DELTA	Y DELTA	ROTA-TION	SCALE
N3UV01ALQ	EXP-DTH	yes	0.	0.	0.3194	-0.0148	0.	0.
N3UV01AMQ	EXP-DTH	yes	0.	0.	0.1354	0.1049	0.	0.
N3UV01ANQ	EXP-DTH	yes	0.	0.	1.0E-4	-3.0E-4	0.	0.
N3UV01AOQ	EXP-DTH	yes	0.	0.	0.1395	-0.1362	0.	0.
N3UV01020	PROD-TARG	yes	0.	0.	0.	0.	0.	0.

6.3.8 Step 4: Create Median Image

Figure 6.15: Image of the EPAR Window for the Fourth Drizzle Step



STEP 4

When ‘median=yes’, this step creates a median image from the separate drizzled input images, allowing a variety of combination and rejection schemes. If ‘combine_type’ is set to ‘median’, then the routine carries out a similar calculation to the standard IRAF task imcombine, with equivalent behavior for the parameters ‘combine_nlow’ and ‘combine_nhigh’ (the number of low and high pixels to reject), and ‘combine_grow’ (the amount by which flagged pixels can grow). All imcombine parameters other than those specified above are reset to their default values.

If ‘combine_type=minmed’, a more sophisticated algorithm is used to combine images. The basic concept is that each pixel in the output combined image will be either the median or the minimum of the input pixel values, depending on whether the median is above the minimum by more than n times sigma. An estimate of the “true” counts is obtained from the median image (after rejecting the highest-valued pixel), while the minimum is actually the minimum unmasked (“good”) pixel. This algorithm

is designed to perform optimally in the case of combining only a few images (3 or 4), where triple-incidence cosmic rays often pose a serious problem for more simplified median combination strategies. The 'minmed' algorithm performs the following steps:

- Create median image, rejecting the highest pixel and applying masks.
- Use this median to estimate the true counts, and thus derive an rms.
- If the median is above the lowest pixel value by less than the first value in 'combine_nsigma', then use the median value, otherwise use the lowest value.
- If 'combine_grow' > 0, repeat the above 3 steps for all pixels around those that have already been chosen as the minimum, this time using a lower significance threshold specified as the second value in 'combine_nsigma'. \

The last step is very successful at flagging the lower signal-to-noise "halos" around bright cosmic rays which were flagged in the first pass.

If 'median_newmasks=yes', then the singly drizzled weight maps ('*_single_wht.fits') are used to create pixel masks for each image (with values 0 and 1) which are named '*_single_wht_maskhead.pl'. The IRAF task mask_head prepares the singly drizzled images by populating the header bad pixel mask keyword 'BPM' for each image. These masks will be used by imcombine when combining images, where the assumed mask parameters are 'masktype=goodvalue' and 'maskvalue=1', indicating that pixels assigned a value of 1 are considered good.

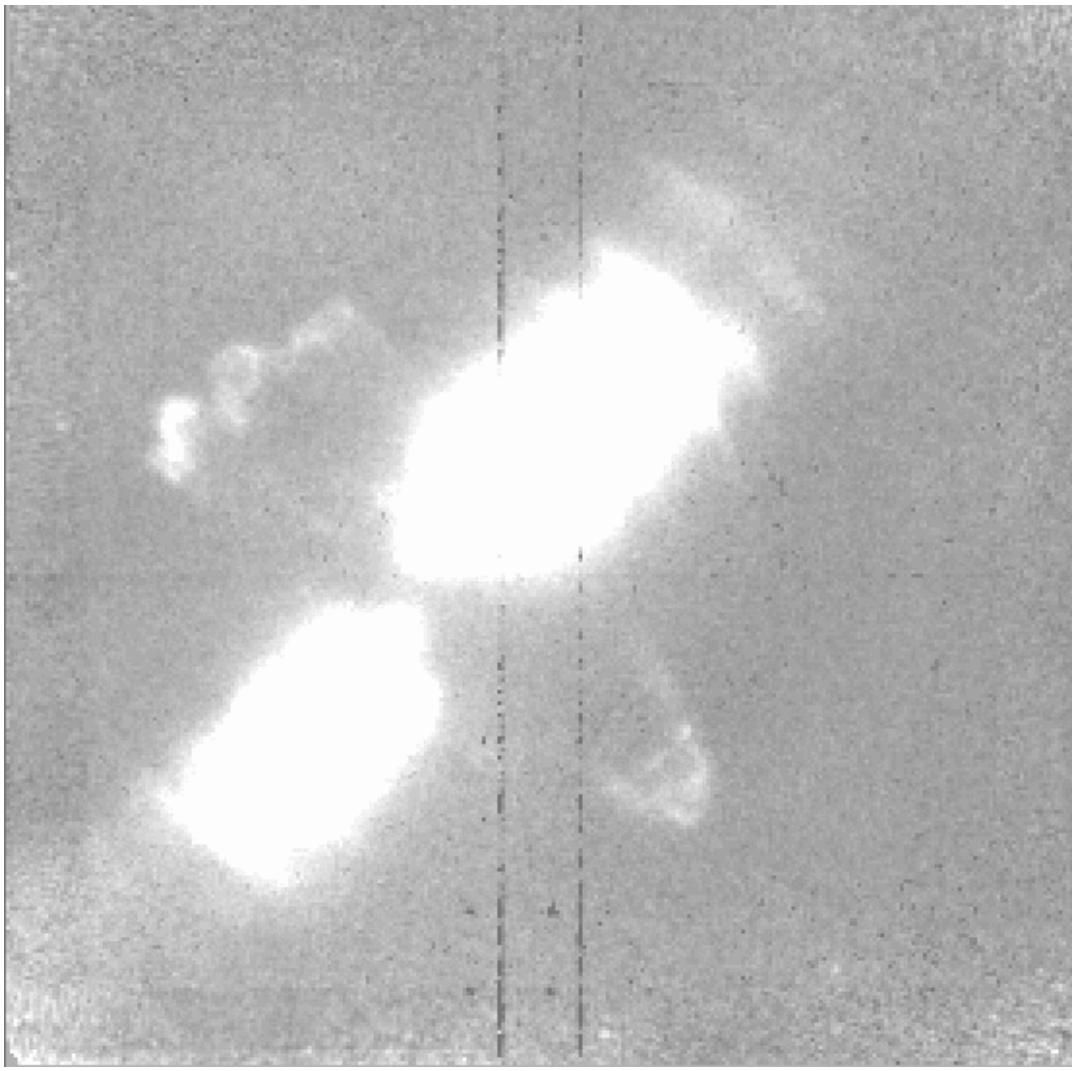
If 'median_newmasks=no', this task will use whatever masks are specified by the user (and which are created offline) in the 'BPM' header keyword of each image. In general, however, it is recommended that the pixel masks which are generated by default are used instead.

Selecting the best parameters for the median step can be an iterative process and should always involve examination of the clean, combined product to verify that the majority of cosmic rays and other artifacts are successfully removed. The rejection algorithm which is ultimately chosen depends largely on the number of datasets being combined and the amount of overlap between dithered images.

In this example, we have chosen the default parameters 'combine_type=minmed', 'combine_nlow=0', and 'combine_nhigh=1'.

This median image is shown (Figure 6.16):

Figure 6.16: The Median Combined NICMOS Image



The median combined image .

The vertical lines seen in the median image above are caused by the NICMOS quadrant boundaries. Since our dataset comprises several dithered observations, we can further improve this by using a static bad pixel map. This file must be a multi-extension file containing one extension named MASK. The simplest way to create such a file is to copy one of the original CAL files, rename the DQ extension and set all pixels to 1.0 except for the column near the middle of the detector. MultiDrizzle simply multiplies the input data by the content of the static bad pixel map. The latter should hence label good pixels with values of 1.0 and bad ones with values of 0.0.

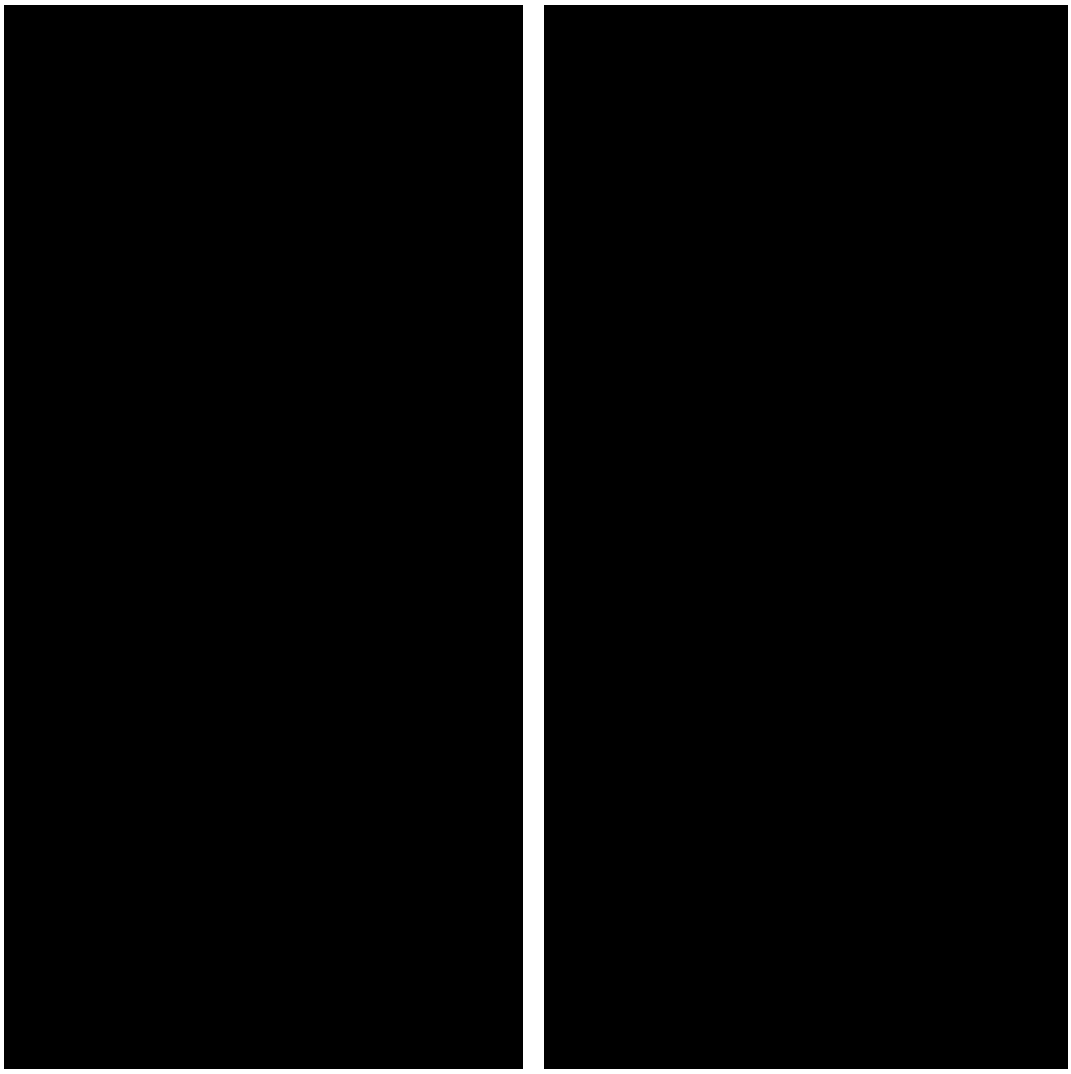
One way within Pyraf to do this is:

```
!cp n3uv01alr_cal.fits static.fits
hedit static.fits[DQ] extname MASK update+ verify-
import pyfits
```

```
fin = pyfits.open("static.fits",mode="update")
fin['MASK'].data[:,:]= 1.
fin['MASK'].data[:,125]=0.
fin['MASK'].data[:,126]=0.
fin['MASK'].data[:,127]=0.
fin['MASK'].data[:,128]=0.
fin['MASK'].data[:,129]=0.
fin.close()
```

The DQ extension of the static.fits file should look like this. This is the content of the 3rd extension in the static.fits file. The content of the other extensions, which are nevertheless required, is ignored:

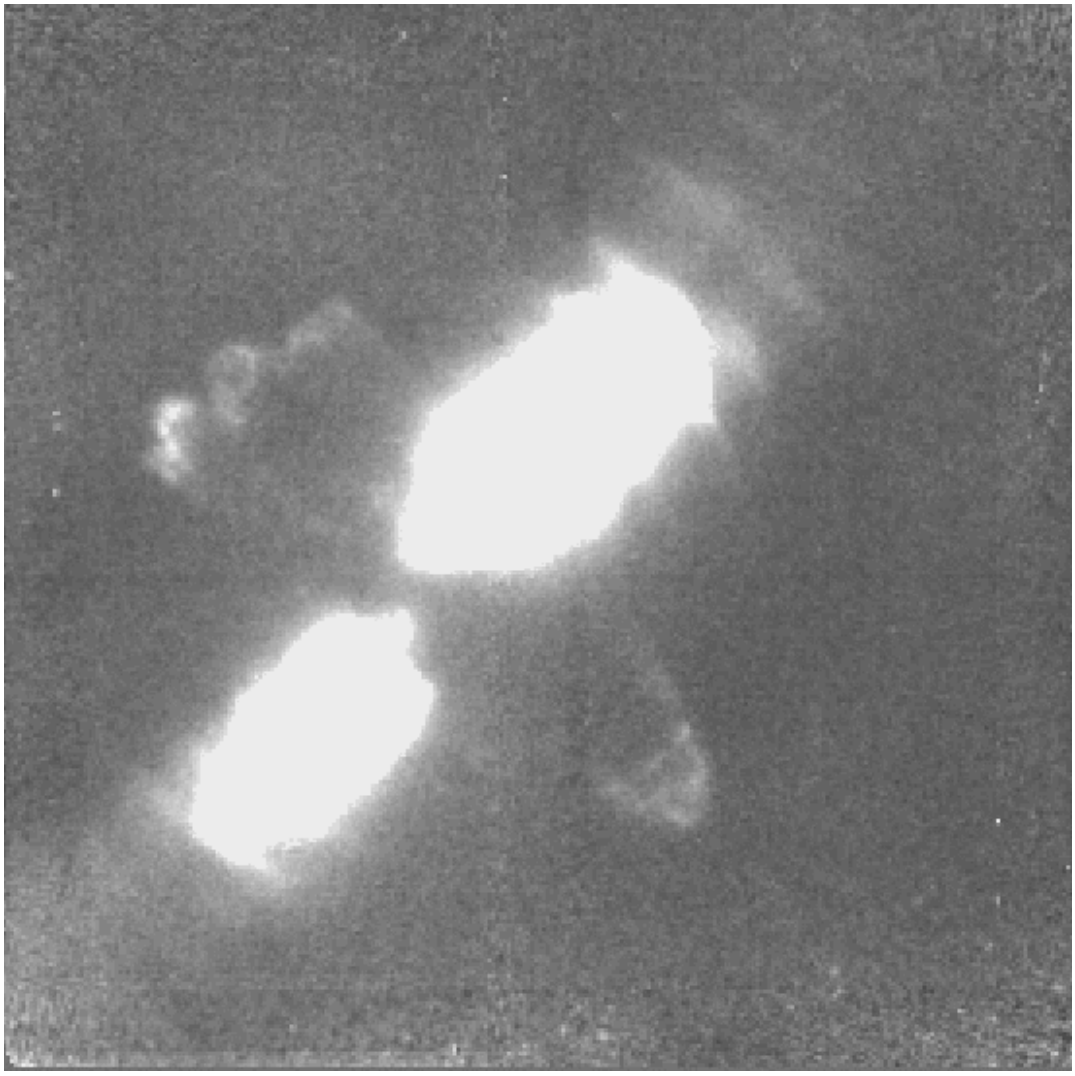
Figure 6.17: The NICMOS Static Mask Image



The static mask image

Running MultiDrizzle after setting ‘static.fits’ file, with parameters ‘combine_type=minmed’, ‘combine_nlow=0’, and ‘combine_nhigh=0’, results in the following improved median file (Figure 6.18):

Figure 6.18: The Improved NICMOS Median Image



The improved median image using the static mask file.

6.3.9 Step 5: Blot back the median image

Figure 6.19: Image of the EPAR Window for the Fifth Drizzle Step

STEP 5: BLOT BACK THE MEDIAN IMAGE

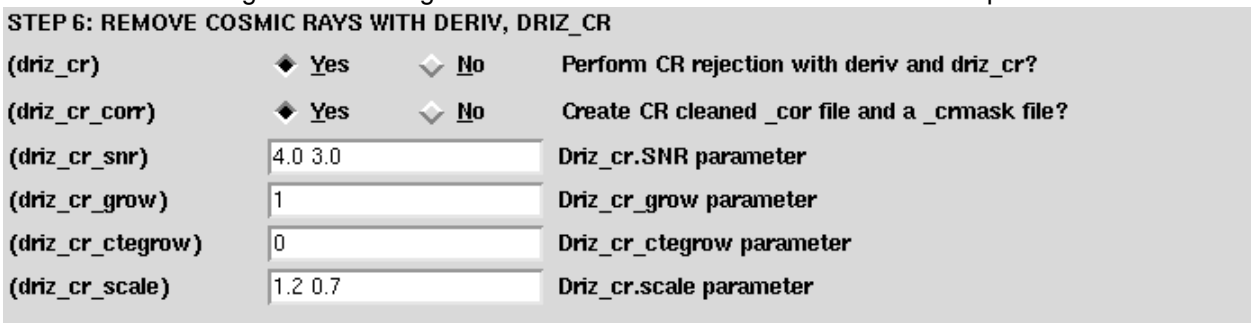
(blot)	◆ Yes ◇ No	Blot the median back to the input frame?
(blot_interp)	poly5	Interpolant (nearest,linear,poly3,poly5,sinc)
(blot_sinscl)	1.0	Scale for sinc interpolation kernel

STEP 5

When ‘blot=yes’, this task takes the median image and uses the **dither** package **blot** task to apply the geometric distortion and to transform (‘reverse drizzle’) it back to the reference frame of each of the original individual input images. This involves reversing the shifts and reapplying the geometric distortion that had been removed in step 3. In addition, the median image is resampled to the pixel scale of the original images and is trimmed to match the dimensions of each input image. This step is done in preparation for subsequent cosmic ray rejection in step 6. The blotted frames are named ‘*_sci?_blt.fits’. If desired, the user may wish to display the input images and blink them with their ‘blotted’ counterparts. The ‘blotted’ images should align perfectly with their respective input images and should be reasonably similar in appearance, except for the fact that they should be cleaned of cosmic rays and other defects.

6.3.10 Step 6: Remove Cosmic Rays with DERIV or DRIZ_CR?

Figure 6.20: Image of the EPAR Window for the Sixth Drizzle Step



STEP 6 in the process

First, the deriv task uses the blotted median images (‘*_sci?_blt.fits’) from step 5 to calculate the absolute value of the difference between each pixel and its four surrounding neighbors.

These derivative images are used by the task driz_cr when comparing the original and blotted images. First, the original CAL images are compared with the corresponding blotted median image ‘*_sci?_blt.fits’ and its absolute derivative to create a mask of cosmic rays (and other blemishes, like satellite trails). Where the difference is larger than can be explained by noise statistics, or the flattening effect of taking the median, or perhaps 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:

$$|data\_image - blotted\_image| > scale*deriv\_image + SNR*noise$$

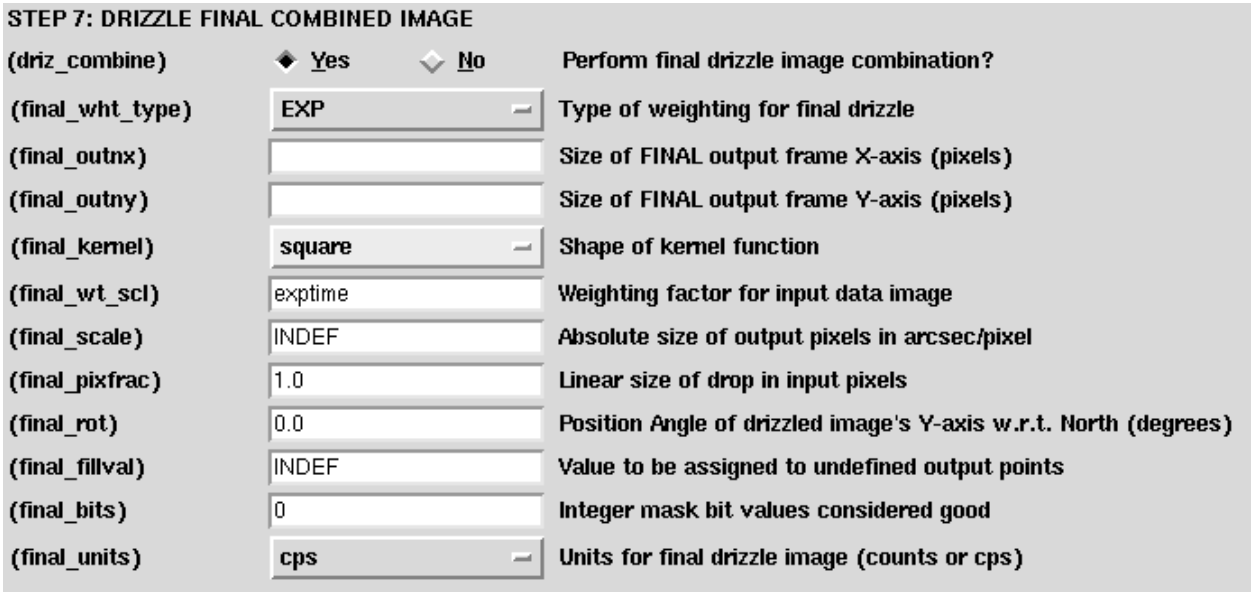
where ‘scale’ is the user supplied **driz_cr** parameter listed above and is defined as the multiplication factor applied to the derivative before determining if the difference between the data image and the blotted image is sufficiently great to require masking. ‘Noise’ is calculated using a combination of the detector read noise and the poisson noise of the blotted median image plus the sky background.

The user must specify a cut-off signal-to-noise (SNR) value for determining whether a pixel should be masked. Actually, two cut-off signal-to-noise ratios are needed, one for detecting the primary cosmic ray, and a second for masking lower-level bad pixels adjacent to those found in the first pass. After the first pass through the image, the procedure is thus repeated on pixels that are adjacent to previously masked pixels using a lower SNR threshold, since cosmic rays often extend across several pixels.

The final output is a cosmic ray mask file named ‘*_mask1.fits’. One of the resulting masks for chip 1 is shown and should be blinked with the original image (or the equivalent science file) to visually ascertain that all cosmic rays were flagged. If it appears that the central pixels of some stars were unnecessarily masked, the ‘driz_cr_scale’ parameter should be increased. If not enough cosmic rays were masked out, this parameter should be decreased. In this example, the default ‘driz_cr_snr’ values “3.0 2.5” were too stringent and resulted in flagging the centers of stars and the core of the Egg Nebula. Instead, we have increased the default SNR values to “4.0 3.0” to create ideal cosmic ray masks for this data set. If the ‘driz_cr_corr’ option is set to ‘yes’, the driz_cr task also creates a ‘*_cor.fits’ image, where flagged pixels are replaced with pixels from the blotted median image. The cosmic ray mask files are then multiplied by the bad pixel masks (which are a combination of the image DQ array and the static masks) to create a final mask file for each input image, ‘*_mask1.fits’, which will be used during final drizzle combination.

6.3.11 Step 7: Drizzle final Combined Image

Figure 6.21: Image of the EPAR Window for the Seventh Drizzle Step



STEP 7

When the following initial setup parameters are set: ‘build=yes’ (default) and ‘context=yes’ (non-default), the final output image will be a single multi-extension FITS file named ‘final_drz.fits’. This file contains the science image in extension 1, the weight image in extension 2, and the context image in extension 3. When ‘build=no’, these files will be written to separate output files. When the default value ‘context=no’ is used, no context image is created.

The first extension of the drizzled product contains the science (SCI) image which is corrected for distortion and which is dither-combined (or mosaiced), if applicable. All pixels have equal area on the sky and equal photometric normalization across the field of view, giving an image which is both photometrically and astrometrically accurate for both point and extended sources. The dimensions of the output image are computed on-the-fly and the default output plate scale is read from the ‘scale’ column in the IDCTAB. These parameters, however, may be chosen by the user to best suit the actual data.

The second extension of the output image contains the weight (WHT) image. This image gives the relative weight of the output pixels and, in standard processing using the defaults, it can be considered an effective exposure time map. The weight image from the example is shown, where darker areas have lower weight. The chip edges are clearly visible, as are column defects and cosmic ray features. The bulk of the image is “white” corresponding to the overlap of all six inputs. In this area the weight value is ~1160, equal to the sum of the exposure times of the six images which contribute.

If the option ‘context’ was set to ‘yes’, the third extension of the output image contains the context (CTX) image which encodes information about which input image contributes to a specific output pixel. This is done using a bitmask for each

output pixel, where 'bit set' means that the image, in the order it was combined, contributed with non-zero weight to that output pixel. The context image starts as a single 32-bit integer image but is extended as a cube with additional 32-bit deep planes as required to handle all the input images. As there are four input images, this image has 4 bit planes which may be set. The darkest area shown corresponds to the area with all four inputs and hence has all the following even bits set: 1+2+4+8=15. cosmic ray hits or other defective pixels contribute to the appropriate bit plane with zero weight and hence appear as lighter spots.

6.3.12 Other Filters

The tutorial above can be repeated using the following shift file for the F165M filter data:

```
# units: pixels
# frame: input
# form: delta
n3uv01b3r 0.0006   -0.0000    0.00
n3uv01b2r 0.0       0.0000    0.00
n3uv01b6r 0.1955   -0.3277    0.00
n3uv01b4r 0.0554   -0.2480    0.00
```

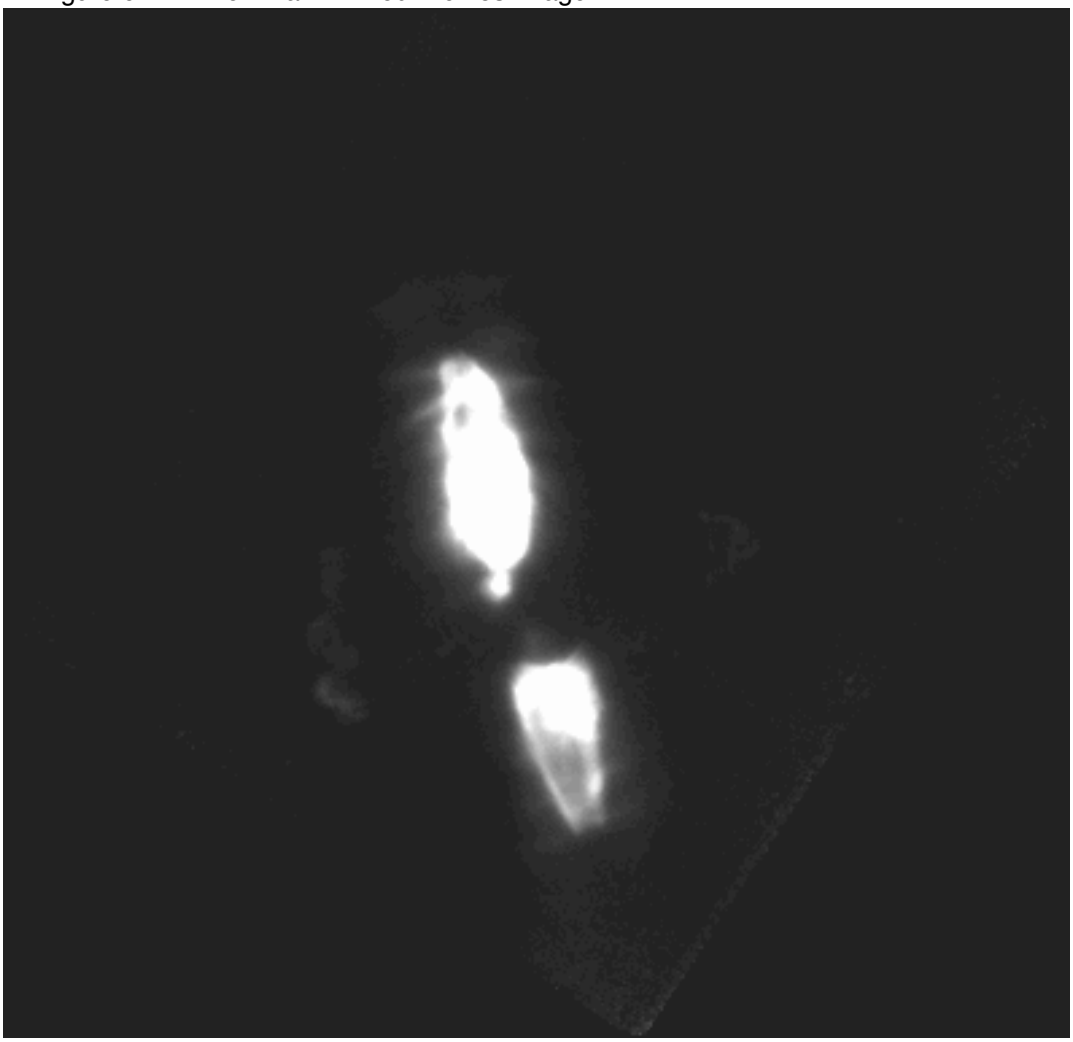
and the following shift file for the F212N filter data:

```
# units: pixels
# frame: input
# form: delta
n3uv01b3r 0.0006   -0.0000    0.00
n3uv01b2r 0.0       0.0000    0.00
n3uv01b6r 0.1955   -0.3277    0.00
n3uv01b4r 0.0554   -0.2480    0.00
```

6.3.13 Examining the Results

The following shows the content of the F190N MultiDrizzle result (Figure 6.22):

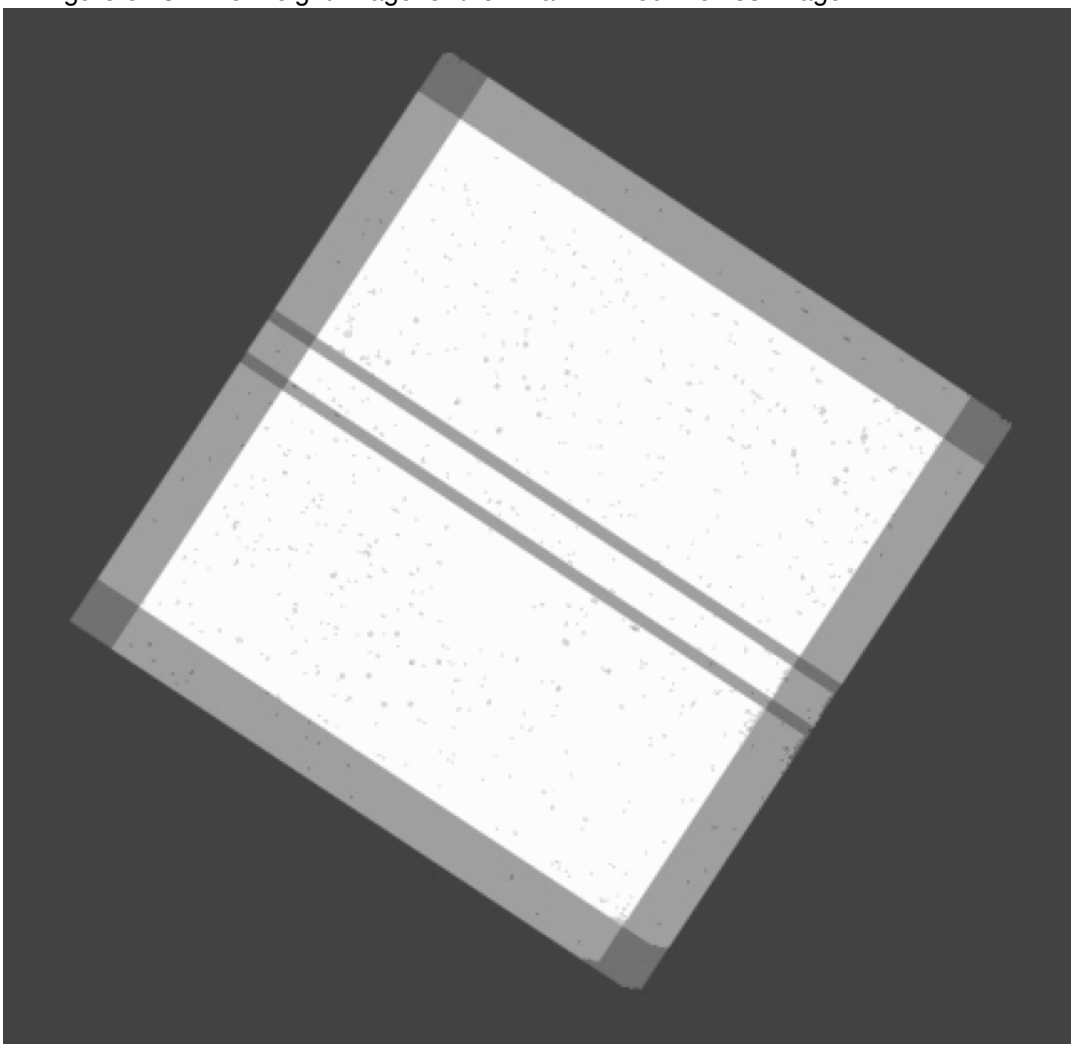
Figure 6.22: The Final Drizzled Nicmos Image



The F190N MultiDrizzle result.

The second extension of the example_drz.fits file generated by MultiDrizzle contains the weight map (Figure 6.23):

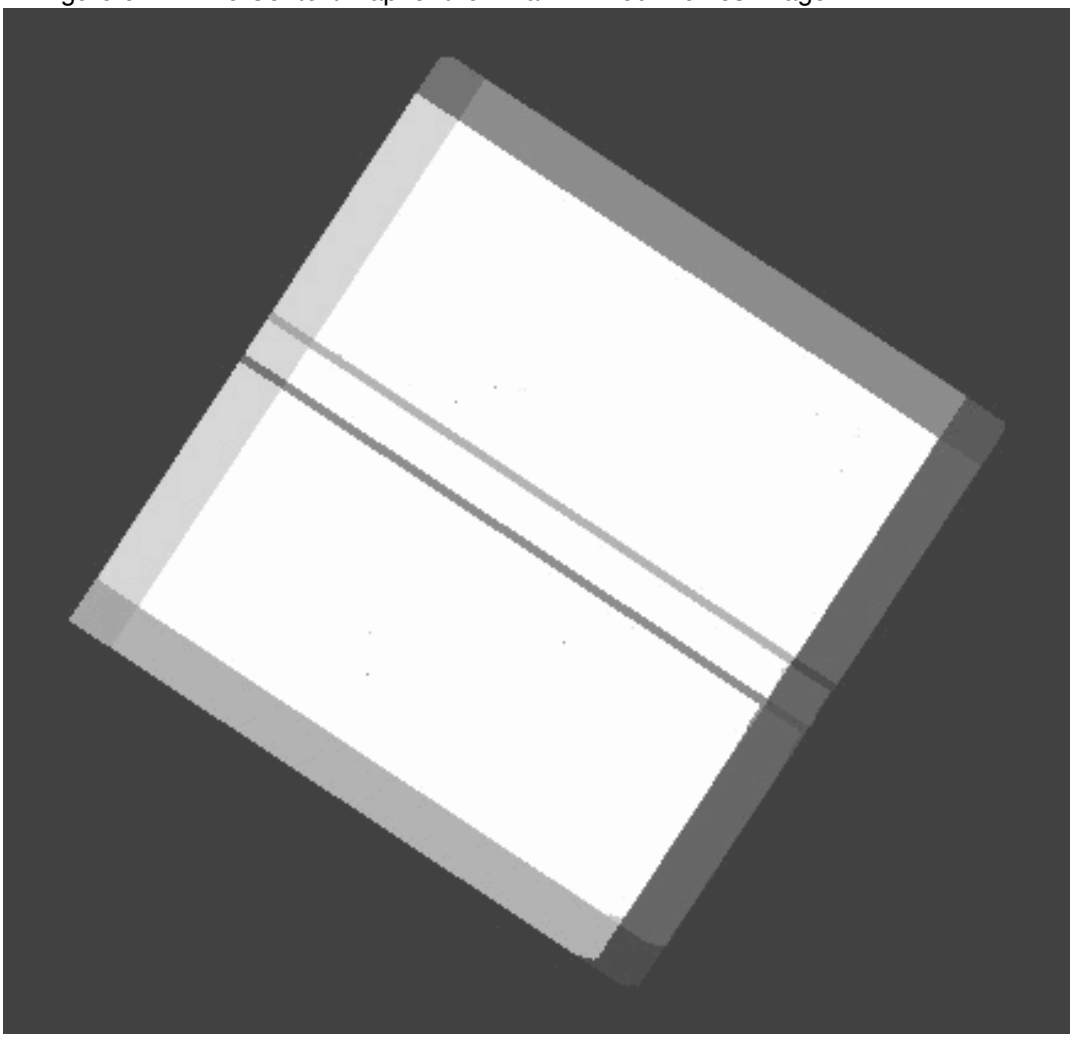
Figure 6.23: The Weight Image for the Final Drizzled Nicmos Image



The weight map that has been generated by MultiDrizzle.

If a context map is generated, it is available in the third extension of the MultiDrizzle output (Figure 6.24):

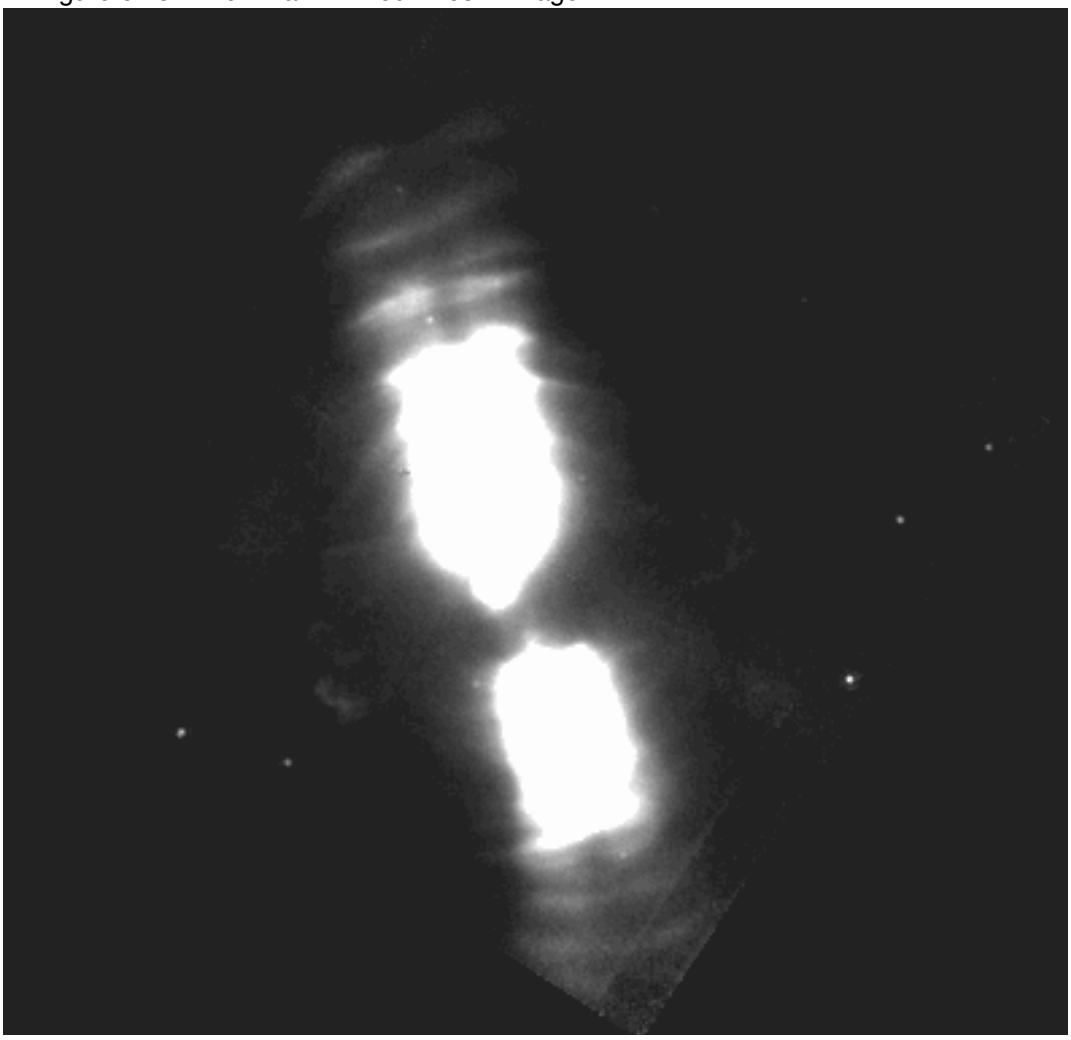
Figure 6.24: The Context Map for the Final Drizzled Nicmos Image



The context map that has been generated by MultiDrizzle.

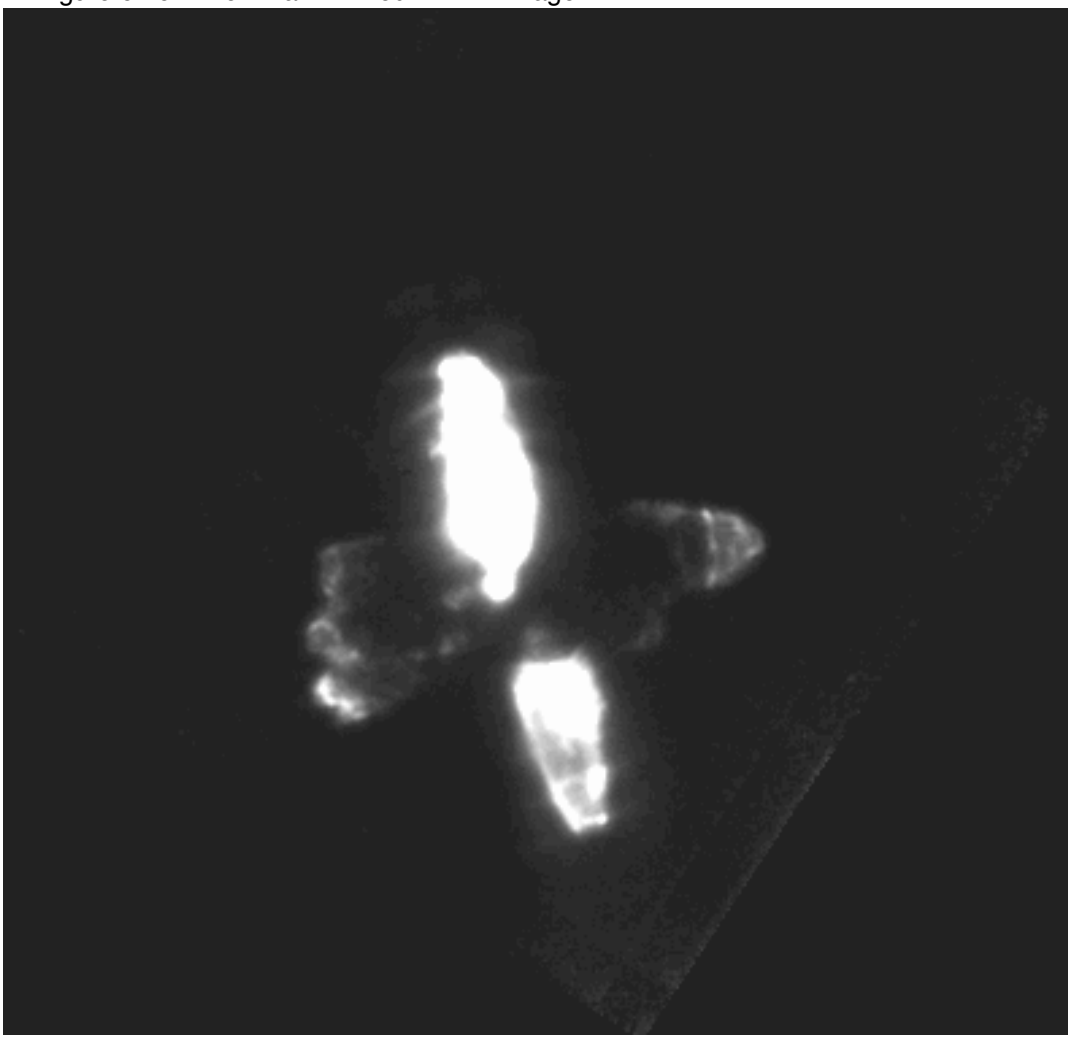
The other images, assembled using the F165M and F212N data, are shown below:

Figure 6.25: The Final Drizzled F165M Image



The F165M MultiDrizzle result.

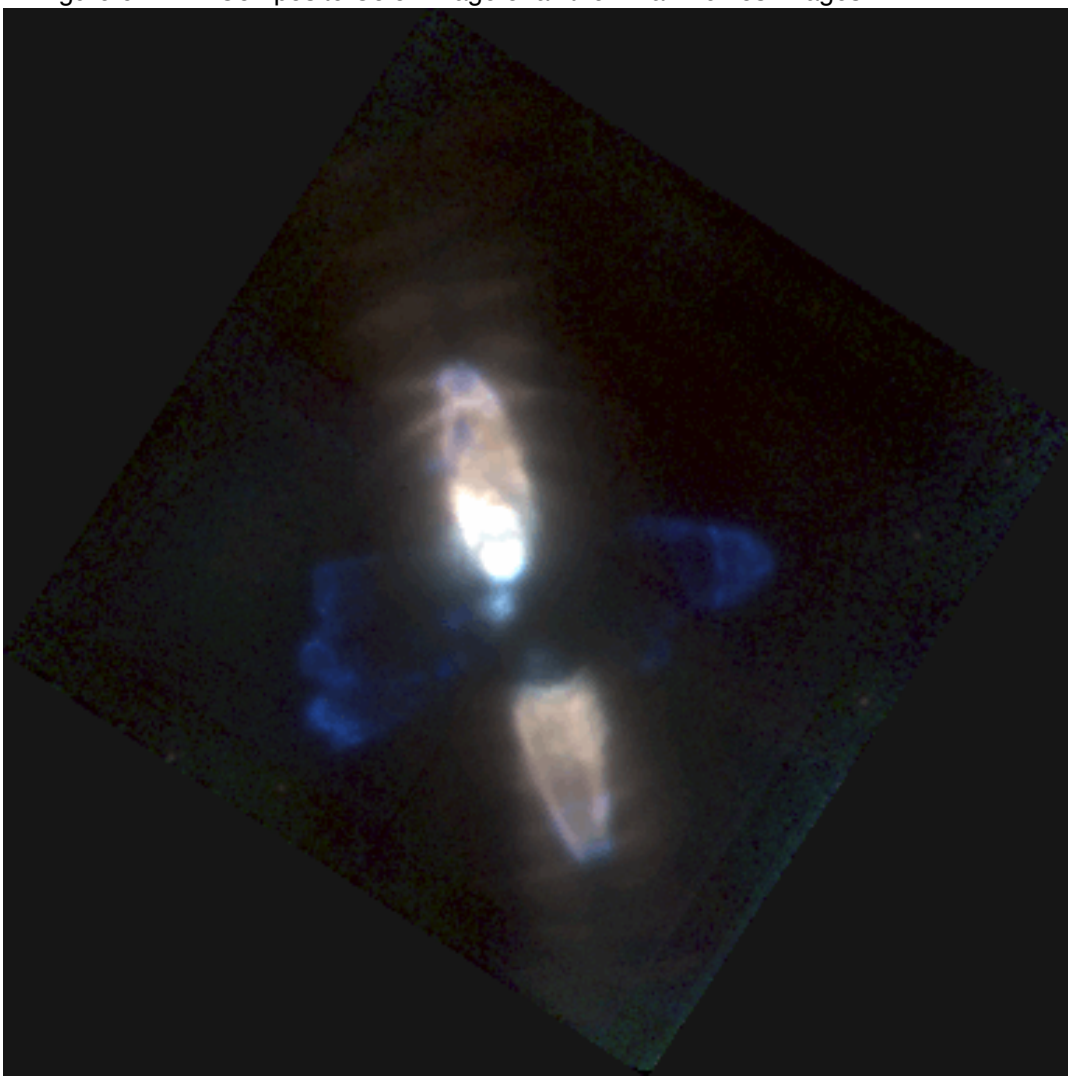
Figure 6.26: The Final Drizzled F212N Image



The F212N MultiDrizzle result.

Finally, a color composite using all of the available data can easily be created using a program such as Stiff:

Figure 6.27: A Composite Color Image of all the Final Nicmos Images



A color composite using the F165M, F190N and F212N images created above.

6.4 WFPC2

6.4.1 Introduction

The HST calibration pipeline does not automatically drizzle (combine and clean) associated WFPC2 datasets, i.e. datasets employing a dither or mosaic pointing pattern (or a pattern defined with POS TARGs). In addition to the following example, the online WFPC2 drizzling cookbooks:

http://www.stsci.edu/hst/wfpc2/analysis/WFPC2_drizzle.html)

provide more details and examples of drizzling WFPC2 data in the standalone environment. The online cookbooks provide reasonable first-pass parameters for

quickly drizzling various types of WFPC2 datasets using PyDrizzle and MultiDrizzle. Select the cookbook that best represents your dataset, with regards to target placement and pointing strategy. For any given dataset, a few trial-and-error iterations are typically necessary to produce optimal results, so some guidance on inspecting your output and experimenting with parameters is included in each cookbook. Below some cookbooks are links to well-documented sample datasets, which illustrate the processing in greater detail (see their README files).

6.4.2 General tips

Specific applications of these tips appear in many of the cookbooks and sample datasets above.

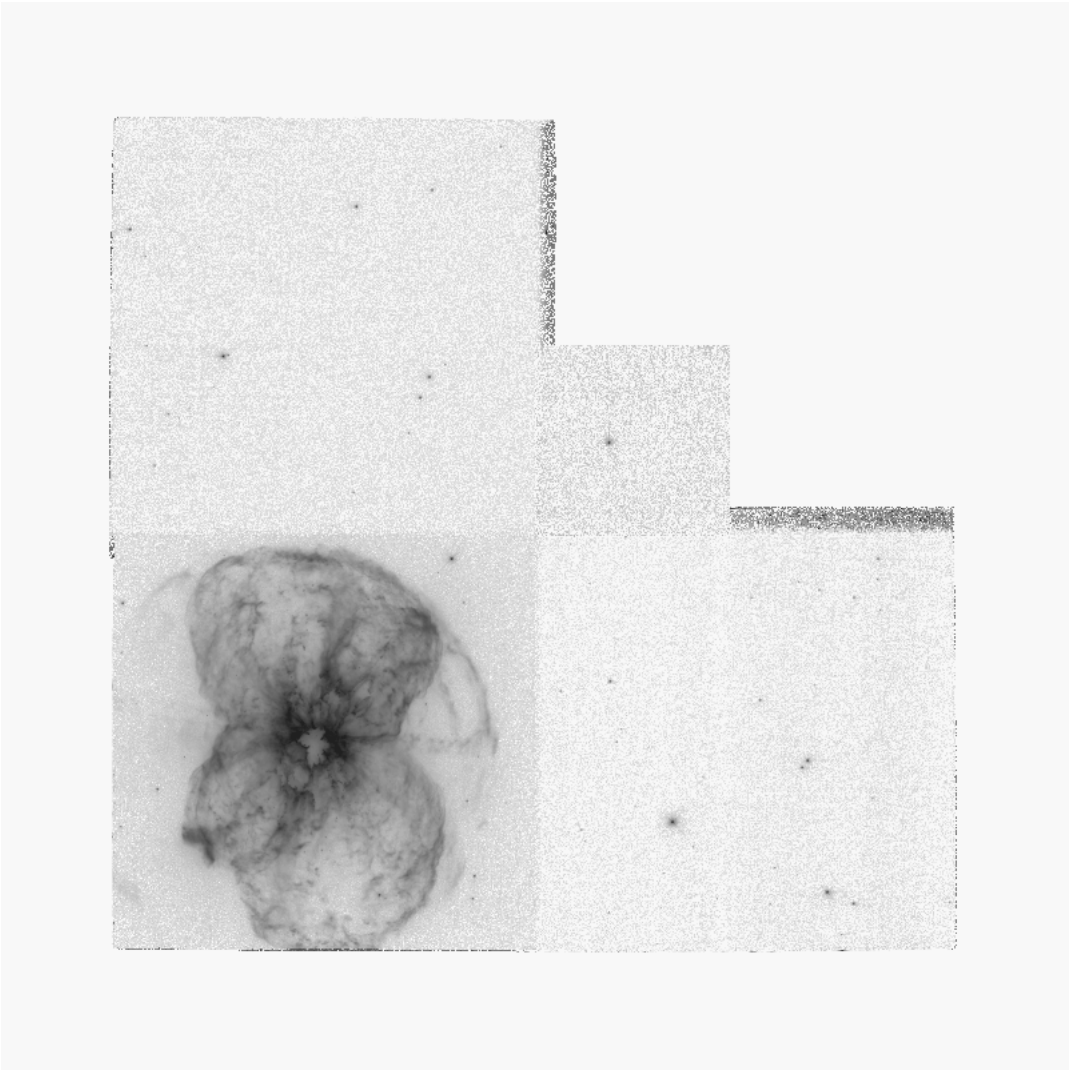
1. Drizzle input files: request only the *c0f.fits (science) and corresponding *c1f.fits (data quality) files from the archive, and their associated best reference files. Note that the best bias and dark corrections are generally not available until several weeks after your observations. So either wait to retrieve your data, or be prepared to re-retrieve it (via on-the-fly-reprocessing or "OTFR") and re-drizzle it. You will need to convert these FITS files to GEIS format.
2. Create a uref directory and put your reference files in it. The drizzle software requires only the distortion correction files indicated in your image headers (keywords IDCTAB and OFFTAB), which can also be downloaded individually from the STScI uref directory.
3. If your dataset involves large mosaic shifts (greater than 100 arcsec), and/or data from different epochs, orientations, or observing programs (using different guide stars), then you will need to measure and apply delta shifts to refine the image registration before combining your data.
4. Output files: set build=no to generate separate files for science image (drz_sci.fits) and exposure weight map (drz_weight.fits). Set context=no to not generate the context image (drz_ctx.fits). Setting clean=yes leaves fewer intermediate files in your working directory, but you may need them for diagnostic purposes (see next bullet).
5. Intermediate files: set clean=no to keep the intermediate files that are helpful while verifying your output and iterating. In addition to inspecting your output files, the median image (*_med.fits) should look almost as good as your final drizzled output, or else it will not help reject cosmic rays and artifacts very well. The single-drizzled images (*_single_sci.fits) may also help verify good cleaning, and may also be needed for image registration.
6. Many types of detector artifacts are flagged in the calibration pipeline, and populate the data quality files (c1h). By default, the drizzle bits are set to zero, meaning any/all flagged pixels will be excluded from the processing. If you wish to include some of these pixels, set bits to include them (and sum the bits for multiple types). For example, to include saturated pixels and warm pixels, set bits = 8+1024 = 1032.

7. If the target is on both the PC and WF chips, the data should be drizzled as if it is all WF data. Or the PC data could be drizzled separately (specify group=1) following instructions for PC data.
8. If any part of your target falls on the WF4 chip, and the data was obtained in 2004 or later, you might need to apply the correction for the WF4 bias anomaly
9. To minimize CPU time and disk space usage (especially during early experimental iterations), you can specify the center (ra,dec) and dimensions (outnx, outny) of your output image to center your target in the output (e.g. give NED coordinates), and limit the output to the minimal region of interest (e.g. only the area essential for measuring shifts). This can greatly improve your efficiency when working with large datasets, especially mosaics. Further, if your target falls on only one chip, you can process that chip (group) alone.
10. Due to declining charge transfer efficiency (CTE), bright objects and artifacts (e.g. stars and cosmic rays) may have prominent comet-like tails of deferred charge in the anti-readout direction. You can use the driz_cr_ctegrow parameter to grow the cosmic ray rejections preferentially in the direction of these CTE tails.

6.4.3 NGC 2440 example

The Hubble Heritage Team obtained WFPC2 observations of the planetary nebula NGC 2440 on 6 Feb. 2007, as part of HST program 11090. The target is placed on the WF3 chip, and a sub-sampling dither box pattern was employed, so the resulting data can be drizzle-combined to a finer output scale, as described below.

Figure 6.28: Drizzled WFPC2 Image



Drizzled WFPC2 H-alpha (F658N) image of NGC 2440, showing the target primarily on the WF3 chip (North not up here).

6.4.4 Data calibration and image registration

We began with the standard pipeline-calibrated WFPC2 archival files (*c0f.fits and *c1f.fits). The entire dataset was taken within one visit, using the same guide stars. So we expect the data to already be well-registered. Nonetheless, small guiding excursions can still occur, and drizzling sub-sampled data to a finer output scale is very sensitive to any misregistration among the input frames. Two independent methods were used on the F675W frames (only), to refine the image registration. Using the centroided location of stars in each frame (tweakshifts task), and also a cross-correlation of nebular structure (crosscor task), small delta-shifts were measured and applied during image combination (below) via this shift file (h_n2440_f657w_shifts.txt):

```
# frame: output
# refimage: u9ws040jm_c0h_wcs.fits
# form: delta
# units: pixels
u9ws040jm.c0h    0.0000   0.0000
u9ws040mm.c0h    0.0136   0.0088
u9ws040pm.c0h    0.0304   0.0205
u9ws040sm.c0h    0.0245   0.0211
```

Since we are combining 4 exposures for each filter, the rejection of typical detector artifacts (bad columns and hot pixels) and cosmic rays, should be excellent. But there has been no manual masking of large artifacts such as satellite trails, scattered light (e.g. “dragon’s breath from a bright star near CCD edges), or optical filter ghosts. So if they are present in this dataset, there could be residual artifacts of their incomplete rejection in the final output.

6.4.5 Image combination and cleaning

These instructions are intended for datasets where an optimally sub-sampling dither box pattern was used, and the target is primarily on the WF chip/s. De-archive the science and data quality FITS files to your working directory. Make a uref directory and download the distortion reference files (IDCTAB and OFFTAB in your image headers) into it, and define your uref directory (set uref). Convert the files to GEIS format (with strfits), and make a list of input images (list_c0h):

```
> set uref = "/data/mymachine/uref/"
> strfits *c0f.fits "" ""
> strfits *c1f.fits "" ""
> ls u*c0h > list_c0h  (no blank lines!)
```

After unpacking the FITS files to GEIS format (using strfits) and making an input list, the following MultiDrizzle parameters were used to combine and clean the four frames for each filter.

```
multidrizzle.static = yes
multidrizzle.static_sig = 4.0
multidrizzle.skysub = no

multidrizzle.driz_separate = yes
multidrizzle.driz_sep_outnx = 1600
multidrizzle.driz_sep_outny = 1600
multidrizzle.driz_sep_kernel = 'turbo'
multidrizzle.driz_sep_wt_scl = 'exptime'
multidrizzle.driz_sep_scale = 0.1  # the WF pixel scale
multidrizzle.driz_sep_pixfrac = 1.0
multidrizzle.driz_sep_rot = INDEF
multidrizzle.driz_sep_fillval = -9.9   # arbitrary low value
multidrizzle.driz_sep_bits = 0  # exclude all flagged pixels

multidrizzle.median = yes
multidrizzle.median_newmasks = yes
```

```
multidrizzle.combine_type = 'median'
multidrizzle.combine_nsigma = '4 3'
multidrizzle.combine_nlow = 0
multidrizzle.combine_nhigh = 1
multidrizzle.combine_lthresh = '-8.8' #exclude empty pixels
multidrizzle.combine_hthresh = 'INDEF'
multidrizzle.combine_grow = 1

multidrizzle.blot = yes
multidrizzle.blot_interp = 'poly5'
multidrizzle.blot_sinscl = 1.0

multidrizzle.driz_cr = yes
multidrizzle.driz_cr_corr = no
multidrizzle.driz_cr_snr = '4.0 3.5'
multidrizzle.driz_cr_grow = 1
multidrizzle.driz_cr_ctegrow = 0
multidrizzle.driz_cr_scale = '1.2 0.7'

multidrizzle.driz_combine = yes
multidrizzle.final_wht_type = 'EXP'
multidrizzle.final_outnx = 1600
multidrizzle.final_outny = 1600
multidrizzle.final_kernel = 'square'
multidrizzle.final_wt_scl = 'exptime'
multidrizzle.final_scale = 0.06  # output pixels 60% of input pixels
multidrizzle.final_pixfrac = 0.8 # shrink the drop size
multidrizzle.final_rot = INDEF  # 0.0 to rotate North up
multidrizzle.final_fillval = INDEF
multidrizzle.final_bits = 0  # exclude flagged pixels
multidrizzle.crbit = 0
```

Drizzling target region to sub-pixel scale (repeat for other filters):

```
multidrizzle input='@list_c0h' output='h_n2440_f675w_s06n'
```

6.4.6 Output and data quality

The output pixel scale (0.06 arcsec/pixel) is 60% of the input detector scale (0.10 arcsec/pixel for the WF chips, since the target is on WF3 chip). A pixfrac of 0.8 was used. Optimal values for scale and pixfrac may be slightly different, and could be found with further experimentation. A good rule-of-thumb is to keep the rms of any region of the exposure weight map under 30% of the mean. The drizzled FITS files described here are available as High-Level Science Products (HLSP) via the Multimission Archive at STScI (MAST):

http://archive.stsci.edu/prepds/heritage/ngc2440/).

So as an exercise, the steps above could be repeated and compared to the HLSP images. Preview GIF images of these files are also available on the MAST Web site, and the resulting color composite image and related information are available from the associated STScI press release:

http://hubblesite.org/newscenter/archive/releases/2007/09/).

6.5 STIS

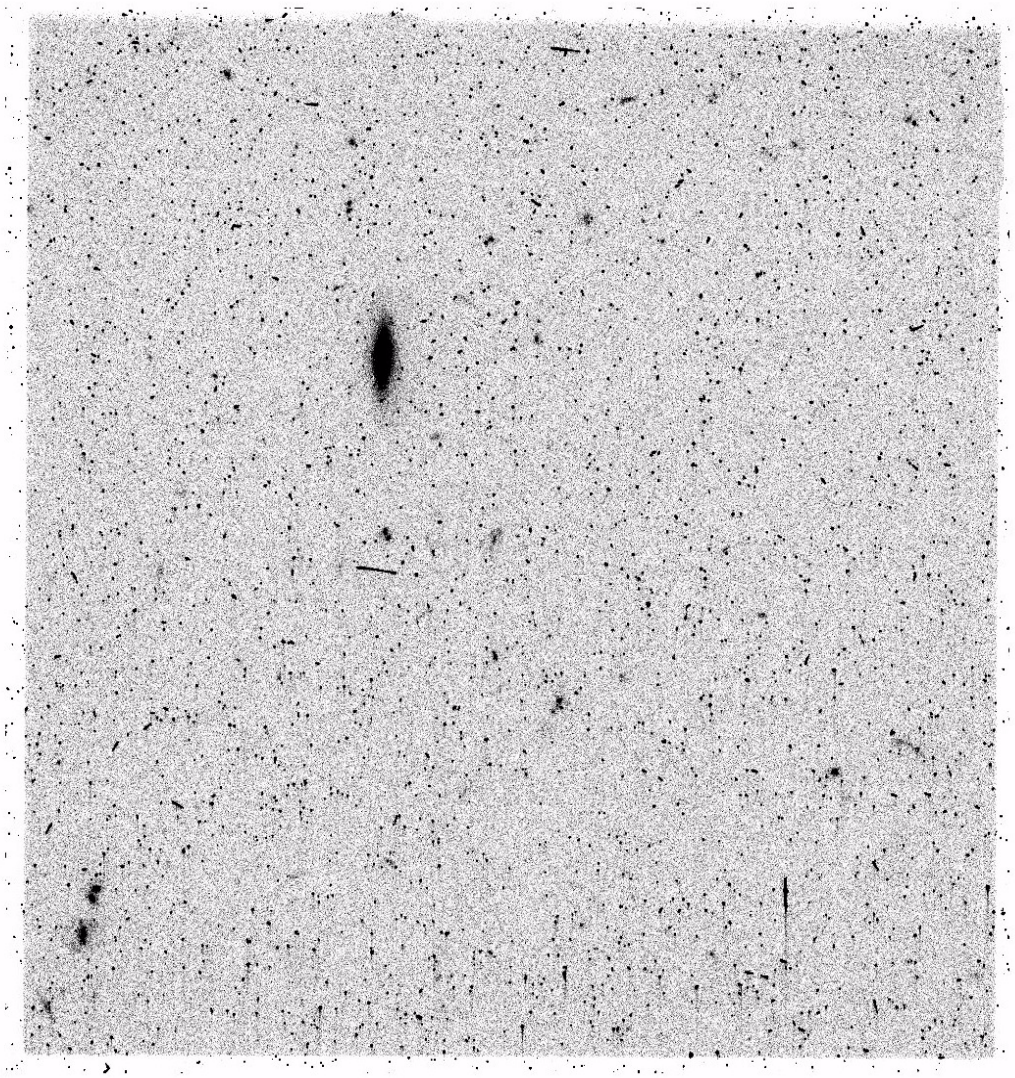
6.5.1 Drizzling STIS Data

By comparison to ACS or WFPC2, planning dithered STIS observations, and drizzling STIS data is relatively simple. There is only one chip to worry about, and the differential distortion across that chip is small, so dithers are quite uniform across the entire chip. The examples provided in the ACS section will give the user a good introduction to questions concerning output pixel size and cosmic ray parameters, for drizzling as the pixel sizes and intrinsic PSFs of ACS and STIS are nearly identical.

One of the authors (Andy Fruchter) has used STIS extensively and has found for his imaging data, the default MultiDrizzle parameters work very well with a few exceptions discussed in the ACS example section: the resolution of STIS is sufficiently good such that when several dithered images are available it makes sense to use both a pixfrac and scale smaller than the native resolution. A pixfrac in the range of 0.8 and a scale of about 0.0333" work well for the final image (and the latter pixel scale also equates three pixel to 0.1"). When doing the earlier single image drizzle for cosmic ray rejection, the reduced output pixel scale should be used, but it is best to leave that pixfrac = 1.

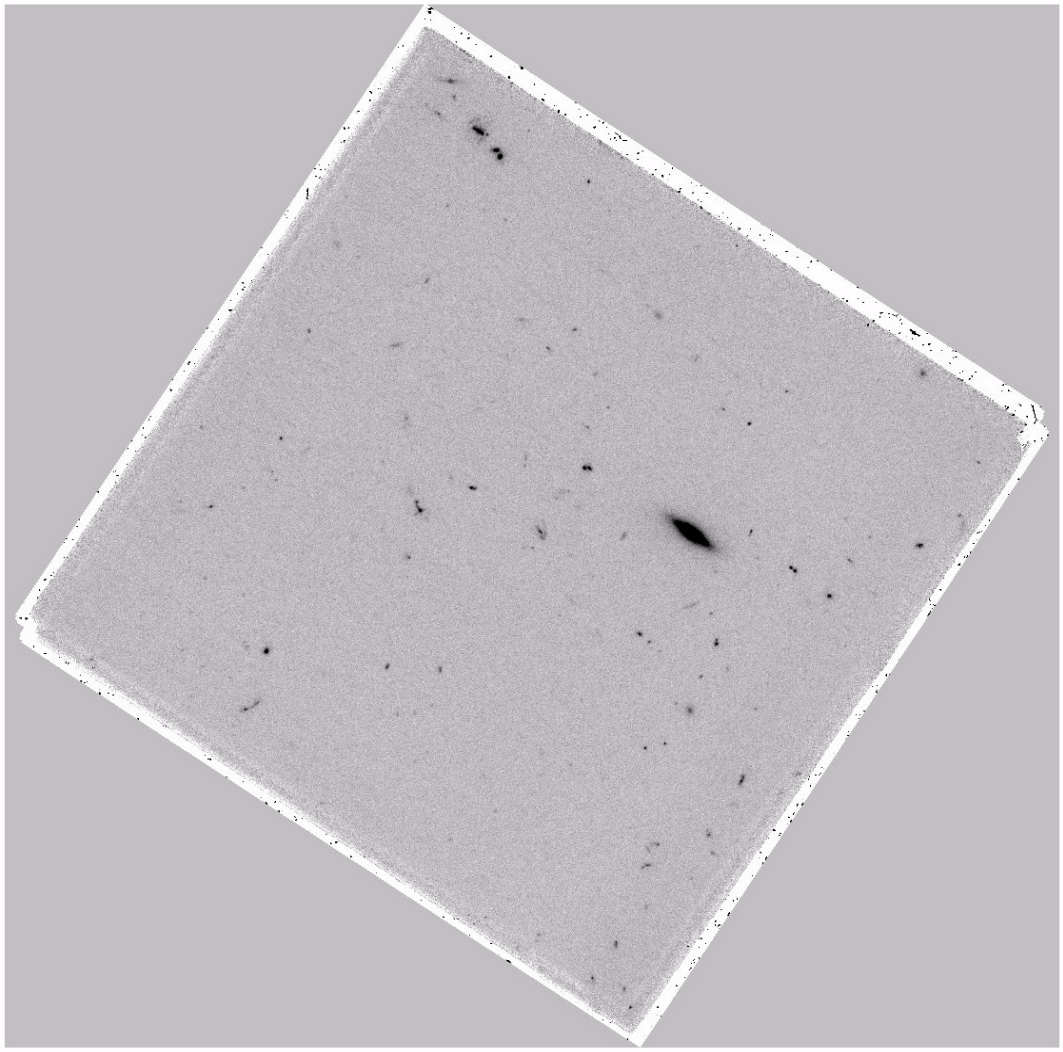
An excellent example of dithered STIS data can be found in the HST archive under data sets o59275*. These are eight dithered STIS images of the field of the host of GRB 990123. When one has more than a few images, such as here, the user may wish to also change from the default "minmed" cosmic ray rejection (which handles small numbers of images well) to the "median" with "combine_nhigh=1" This takes the median of images after first rejecting the highest value in the various images for a given pixel and this procedure is somewhat better adapted to a large number of images than is the standard "minmed". One of the single STIS images, and the combined drizzled image using the parameters discussed above are shown in Figure 6.29 and Figure 6.30.

Figure 6.29: Single STIS Image Ready for input to MultiDrizzle



Single STIS image before MultiDrizzle has been run.

Figure 6.30: Final Drizzled STIS Image



The combined drizzled image using the parameters described above.

6.6 WFC3

WFC3 has a CCD detector, “UVIS”, and an infrared detector, “IR”. The UVIS detector has a plate scale ~0.04 arcsec/pixel, and the point spread function (PSF) has FWHM 1.6 to 2.3 pixels, depending on wavelength. The IR detector provides coarser sampling of the PSF, with a plate scale of 0.13 arcsec/pixel and FWHM of 1.0 to 1.2 pixels, so IR imaging will benefit even more than UVIS imaging from combining dithered exposures. Both detectors have substantial non-linear distortion, so a dither step will have different sizes in pixels at different locations on the detectors.

WFC3/UVIS images will be similar to ACS/WFC images, as can be seen in the table of SIAF file information in “Summary of Detector Plate Scales” under “Detector Plate Scales and Geometric Distortion”. These detectors comprise two rectangular

CCD chips which together form a square. They have roughly comparable fields of view and plate scales. Both have strong linear and non-linear distortion, as explained in “Summary of Detector Plate Scales”. The linear distortion is characterized by an angle of 86 degrees between the projected x and y axes of the WFC3/UVIS detector on the sky, and 85 degrees between the projected x and y axes of the ACS/WFC detector. Note that the on-axis location of WFC3 in the telescope does not ensure that either the linear or non-linear distortion will be small, since the optical path within the instrument is the determining factor.

The WFC3/IR detector is unique among the HST imaging instruments in having a rectangular projection on the sky to first order. At 1100 nm, it has a percentage of PSF flux in the peak pixel comparable to that of the NICMOS NIC3 detector during optimal focus runs, but has a much larger field of view and more uniform sensitivity across pixels. WFC3/IR data is taken in specified timing sequences of non-destructive reads, and the data reduction pipeline performs cosmic ray rejection using an up-the-ramp fitting process over the frames in the sequence. cosmic rays identified in the pipeline reductions for IR sequences and cr-split UVIS exposures will be flagged with DQI value 8192, while those identified by MultiDrizzle will be flagged with 4096.

Examples of using MultiDrizzle to process WFC3 images will be made available after WFC3 data have been obtained in orbit following Servicing Mission 4.

6.7 Scripting

6.7.1 Introduction

So, you have a bunch of data sitting around on disk and you would like to drizzle it all in a common and consistent manner. If all the images have similar content and this is appropriate for your data, recording your method in a nice reusable script is a great way to go! Scripting is a great way not only to record what you’ve done to process a particular dataset, but it allows you to re-process the data later on if you decided to make changes. It also gives you a nice framework to use for processing other datasets.

This is a simple example showing how to write a script that will keep track of your data, run it through MultiDrizzle and produce some nice information about the results. This is designed for more advanced users who were comfortable with Python/Pyraf programming and looking for a start on automating their data reduction pipeline. If you are not familiar with the Python environment, you should still be able to use this script as a starting point and edit it to suit your needs. More advanced users might want to add extra features.

The actual data is unimportant for this example, you can add your own image names, or leave the scripts general so they are applicable to many datasets.

6.7.2 The Quick and Easy Way

If you hate scripting, or are just uncomfortable with it, there is an easy way to record what you have done that allows you to repeat your process. MultiDrizzle already has an option to save the output script commands to the file specified through the "runfile" parameter. This file saves the commands necessary to create the final drizzled file. You can also simply copy the commanded parameters you used to run MultiDrizzle into a file.

These command files can then be edited down to the important commands which need to be run together. When the PyRAF system is started using the "pyraf" command as described previously, the user's commands are actually being passed to an enhanced interpreter environment that allows use of IRAF CL emulation and provides other capabilities beyond those provided by the standard Python interpreter. In fact, when "pyraf" is typed, a special interpreter is run which is a front end to the Python interpreter. This front-end interpreter handles the translation of CL syntax to Python, command logging, filename completion, shell escapes and the like which are not available in the default Python interpreter.

It is also possible to use PyRAF from a standard Python session, which is typically started by simply typing "python" at the Unix shell prompt. In that case the simple CL syntax for calling tasks is not available and tab-completion, logging, etc., are not active, unless you are in iPython. For interactive use, the conveniences that come with the PyRAF interpreter are valuable and we expect that most users will use PyRAF in this mode.

One important thing to understand is that the alternate syntax supported by the PyRAF front end interpreter is provided purely for interactive convenience. When such input is logged, it is logged in its translated, Python form. Scripts should always use the normal Python form of the syntax. The advantage of this requirement is that such scripts need no preprocessing to be executed by Python, and so they can be freely mixed with any other Python programs. In summary, if one runs PyRAF in its default mode, the short-cut syntax can be used; but when PyRAF is being used from scripts or from the standard Python interpreter, one must use standard Python syntax (not CL-like syntax) to run IRAF tasks.

Even in Python mode, task and parameter names can be abbreviated and, for the most part, the minimum matching used by IRAF still applies. As described above, when an IRAF task name is identical to a reserved keyword in Python, it is necessary to prepend a 'PY' to the IRAF task name (i.e., use iraf.PYlambda, not iraf.lambda). In Python mode, when task parameters conflict with keywords, they must be similarly modified. The statement iraf.imcalc(in="filename") will generate a syntax error and must be changed either to iraf.imcalc(PYin="filename") or to iraf.imcalc(input="filename"). This keyword/parameter conflict is handled automatically in CL emulation mode.

- Start up PyRAF:

```
my_computer> pyraf
```

- Load the stsdas and dither packages

```
--> stsdas
--> dither
```

- Edit the MultiDrizzle parameters and set them to your favorite values and run MultiDrizzle

```
--> epar multidrizzle
```

- now, save the parameters to a local parameter file by using the "SaveAs" button in the EPAR GUI. You should try to pick a name that describes what data configuration this set of parameters applies to when running MultiDrizzle, such as 'multidrizzle_4ptdither.par'.

Now, open a new command file in your favorite editor and add a few lines to it so that it can be executed from the shell, your newly edited file should look something like this:

```
#!/usr/bin/env python

#Load up the necessary software modules
import pyraf
from pyraf import iraf
from iraf import stsdas,dither

#copy the saved version of your multidrizzle parameter
#file to the uparm directory
#the full path might look something like
#/Users/you/iraf/uparm/ on a MAC
#or /home/you/iraf/ on a linux or solaris machine.

dither.multidrizzle(ParList='multidrizzle_4ptdither.par')

#at this point, multidrizzle will run and verify
#the input filenames and the output filename,
#these should be defaulted to the values you
#saved as part of your parameter editing session.
#you can also edit the command above to include
#the input and output names
```

- if you would like to save everything into one file, you can also copy the default MultiDrizzle parameters into your command script. If you are only changing a few things, the easiest way to do this is add the line:

```
iraf.unlearn("multidrizzle")
```

```
dither.multidrizzle(input="j08*flt.fits",output="j08_driz.fits",skys
ub="no")
```

```
my_computer> chmod u+x name_of_file.py
```

```
my_computer> name_of_file.py
```

These are some additional ways in which a simple file like this can be very useful:

- If you have a science program that's taking observations over several months. If you have the data automatically delivered to a local directory whenever it's available, you can set up a crontab job that executes your MultiDrizzle script every so often, giving you the most recent drizzled version of the full dataset.
- You can distribute copies of your batch file to colleagues and students for their use, this means you need to transfer fewer files around, only the original calibrated science files need to be archived, saving everyone disk space.
- You can make multiple script files that cover the different science cases or field morphology that you commonly encounter

CHAPTER 7:

Troubleshooting

In This Chapter...

7.1 Interpreting your results

MultiDrizzle ties together a fairly substantial set of algorithms, each designed to accomplish a different task, and as such has a large parameter set. While the functions have been set to default parameters which should produce good results for a wide range of data, there are times when they must be fine-tuned to achieve the best results. Along the way it is possible to run through the entire process without software error but end up with a product which is not of the optimum science quality. The following section attempts to outline possible outcomes and their probable solutions.

7.1.1 General Issues

- When I display the weight map generated for my final drizzled image (or any of the single drizzled images as well), there seem to be representations of the objects in my science image. What's going on?

There are several issues which you should examine:

1. Most likely you need to refine the parameters that are set during the driz_cr step which detects cosmic rays in each of your science frames. This step uses the science image, a model of that image (the blotted median image), and the absolute derivative of the model image to locate cosmic rays. It uses the scaling, readnoise and gain specified by the user as well as the background level specified in the header of the image as part of it's noise comparison. Often times the cores of saturated or very bright objects will be flagged as cosmic rays if the scaling is improper.

2. You should double check the alignment of your images. You can do this by visually blinking between your science image and your blotted image. If the images are not well aligned when the median is created then the objects in the blotted image will not fall in the same place as the original science image - leading the incorrectly flagged pixels.

- I've calculated refined offsets between my science images but when I supply MultiDrizzle with the delta shifts the final image is not aligned properly.

1. Make sure that you have calculated your shifts and rotations using the same frame of reference that the MultiDrizzle software is expecting. The most important thing is that all your shifts are stated relative to the reference image that you chose. You can specify the reference image that was used to calculate the shifts inside the delta shift file.

- When I look at my output drizzled image, there seem to be obvious seams between each pointing.

1. This could be a sky subtraction problem. Sky in this case can be somewhat of a misnomer, what it means is that the background levels in all your input images have not been normalized to each other. It is important to have a good background subtraction before drizzling your images so that proper cosmic ray rejection can be performed and so that you are not adding unneccesary noise to your combined output image. You might also be drizzling images with vastly different S/N ratios which could have the same affect.

2. This could also be a problem with the shifts between your images. If your shifts are close, but not precise, you may see ringing around your objects, edges between observations and distorted or smeared star profiles. These will vary depending on the morphology of your field.

7.1.2 Instrument Specific Information

These are questions you might encounter as you drizzle your data and are related to specific instrumental characteristics.

7.1.2.1 ACS

- I just retrieved ACS data from the archive, but the images are very noisy and there are stripes all over the fields, which is not common in the ACS data.

The correlated noise "stripes" you see are most prominent when low S/N narrowband exposures are drizzled. When the significant ACS distortion correction is applied to such data, it can impose this Moire-like beat pattern in the noise. The best way to mitigate against this is to employ a dither pattern or otherwise combine more frames from overlapping observations, so that the correlated noise will be out of phase and cancel each other in the combined output. You can run MultiDrizzle yourself to combine any number of frames. Note that in combining data from different visits (which employed different guide stars), you will likely need to manually register the frames (measure and apply shifts).

If the data is undithered, and more frames are not available for combination, you might want to try using the lanczos3 kernel for the final drizzle step in MultiDrizzle. It can help suppress the correlated noise, but note that it doesn't perform well in the presence of artifacts such as hot pixels and cosmic rays. So if you are not combining enough frames to reject all such artifacts, you will notice some "ringing" (a halo of negative pixels) around the sharp edges of any residual artifacts.

CHAPTER 8:

References

Anderson, J., King, I. R., 2000, "Toward High-Precision Astrometry with WFPC2. I. Deriving an Accurate Point-Spread Function", PASP 112, 1360

Anderson, J., 2007, "Variation of the Distortion Solution of the WFC", ACS-ISR 07-08

Arendt, R. G., Fixsen, D. J., Moseley, S. H., 2000, "Dithering Strategies for Efficient Self-Calibration of Imaging Arrays", ApJ 536, 500

Beckwith, S., et al, 2006, "The Hubble Ultra Deep Field", AJ, V132, Issue 5, pp. 1729-1755

Biretta, J., Dithering: Relationship Between POS TARGs and CCD Rows/Columns

Biretta, J., Wiggs, M., General Advice on Dithering HST Observations

Biretta, J., Wiggs, M., Questions About Dithering WFPC2 Observations

Biretta, J., et al., 2001, "WFPC2 Instrument Handbook, Version 6.1", (Baltimore: STScI)

Bowers, C., Baum, S., 1998, "Plate Scales, Anamorphic Magnification and Dispersion: CCD Modes", STIS-ISR-98-23

Brown, T., Hartig, G., Baggett, S., 2008, "WFC3 TV3 Testing: UVIS Window Contamination", WFC3-ISR-2008-10

Casertano, S., Wiggs, M., 2001, "An Improved Geometric Solution for WFPC2", WFPC2-ISR-2001-10

Casertano, S., de Mello, D., Dickinson, M., Ferguson, H. C., Fruchter, A. S., Gonzalez-Lopezlira, R., Heyer, Inge; Hook, R. N., Levay, Z.; Lucas, R. A., Mack, J., Makidon, R. B.; Mutchler, M., Smith, T. E., Stiavelli, M., Wiggs, M. S., Williams, R. E., 2000, "WFPC2 Observations of the Hubble Deep Field South", AJ 120, 2747

Cox, C., 1994, "The WFPC2 Scales and Alignments" , SOB-ISR-94-10-21

Cox, C., Ritchie, C., Bergeron, E., Mackenty, J., Noll, K., 1997,"NICMOS Distortion Correction", OSG-ISR-CAL-97-07

Fruchter, A. S.,"Andy's Dither Page"

Fruchter, A. S., "The Hubble Deep Field - Drizzling"

Fruchter, A. S., "The Hubble Deep Field - Image Registration and Combination",

Fruchter, A. S., Hook, R. N., "Linear Reconstruction of the Hubble Deep Field",

Fruchter, A. S., Hook, R. N., 1997, "A Method for the Linear Reconstruction of Undersampled Images" PASP; astro-ph/9808087

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, "A Package for the Reduction of Dithered Undersampled Images",1997 HST Calibration Workshop, ed. S. Casertano et al.

Fruchter, A. S., Mutchler, M., "Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration"

Gardner, J. P., Baum, S. A., Brown, T. M., Carollo, C. M., Christensen, J., Dashevsky, I., Dickinson, M. E.; Espey, B. R., Ferguson, H. C., Fruchter, A. S., Gonnella, A. M.,

Gonzalez-Lopezlira, R. A., Hook, R. N., Kaiser, M. E., Martin, C. L., Sahu, K. C., Savaglio, S., Smith, T. E, Teplitz, H. I.; Williams, R. E., Wilson, J., 2000, "The Hubble Deep Field South: STIS Imaging", *AJ 119, 486*

Gilliland, R. L.,"Guiding Errors in 3-Gyro: Experience from WF/PC, WFPC2, STIS, NICMOS, and ACS", ISR-TEL-2005-02

P. D., Frandsen, S., Howell, J. H., Lin, D. N. C., Marcy, G. W., Mayor, M.; Naef, D., Sigurdsson, S., Stagg, C. R., Vandenberg, D. A., Vogt, S. S., Williams, M. D., 2000, "A Lack of Planets in 47 Tucanae from a Hubble Space Telescope Search", *ApJ 545, L47*

Gilmozzi, R., Ewald, S., Kinney, E., 1995, "The Geometric Distortion Correction for the WFPC Cameras", WFPC2-ISR-95-02

Gonzaga, S., Biretta, J., Wiggs, M., et al., 1998, "The Drizzling Cookbook" WFPC2-ISR-98-04

Gonzaga, S., Wiggs, M., "Datasets and IRAF Scripts for use with the Drizzle Cookbook"

Greenfield, R., White, R. 2001, "The PyRAF Home Page",

HDF-S Team, "The Hubble Deep Field South - Data Reduction / Technical Information",

Hack, W., Cox, C., 2001, "Revised IDCTAB Definition: Application to HST Data", ACS-2001-008

Hartig, G., Kinney, E., Hodge, P., Lallo, M., Downes, R., 1999, STIS Coordinate System Orientation and Transformations,TIR STIS 99-02

Heyer, I., 2001, "The WFPC2 Photometric CTE Monitor", WFPC2-ISR-2001-09

Holtzman, J., et al., 1995, The Performance and Calibration of WFPC2 on the Hubble Space Telescope, PASP 107, 156

Koekemer, A. M, et al. 2002, "HST Dither Handbook", Version 2.0, Baltimore:STScI

Koekemoer, A. M., Grogin, N. A., Schreier, E. J., Giacconi, R., Gilli, R., Kewley, L., Norman, C., Zirm, A., Bergeron, J., Rosati, P., Hasinger, G., Tozzi, P., Marconi, A., 2001, "HST Imaging in the Chandra Deep Field South: II. WFPC2 Observations of an X-Ray Flux-Limited Sample from the 1 Msec Chandra Catalog", *ApJ (in press)*; astro-ph/0110385

Koekemoer, A. M., Fruchter, A. S., Hook, R. N., Hack, Wl, 2002, "MultiDrizzle: An Integrated Pyraf Script for Registering, Cleaning and Combining Images", 2002 HST Calibration Workshop (eds. S. Arribas, A. M. Koekemoer, & B. Whitmore; Baltimore: STScI), 337

Koekemoer, A. M., Wiggs, M., "The WFPC2 Drizzle Page"

Kozhurina-Platais, V.,J. Anderson, A. M. Koekemoer, 2003, "Toward a Multi-Wavelength Geometric Distortion Solution for WFPC2", WFPC2-ISR-2003-02

Lallo, M., Cox, C., Lupie, O., 1998, A Few Words on Pointing and Jitter File Accuracies, OSG MEMO 1998-09-29

Lauer, T. 1999, "Combining Undersampled Dithered Images", PASP 111, 227

Lauer, T. 1999, "The Photometry of Undersampled Point-Spread Functions", PASP 111, 1434

Malumuth, E. M., Bowers, C. W., 1997, Determination of Geometric Distortion in STIS Images, in *1997 HST Calibration Workshop*, ed. S. Casertano et al.

McLaughlin & Wiklind, et al., 2007,"NICMOS Data Handbook, Version 7.0", (Baltimore: STScI)

Mutchler, M., Cox, C., 2001, "ACS Dither and Mosaic Pointing Patterns", ACS-ISR-2001-07

Mutchler, M., Fruchter, A. S., 1997, "Drizzling Dithered WFPC2 Images - A Demonstration", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.

Sahu, K. C., Casertano, S., Livio, M., Gilliland, R. L., Panagia, N., Albrow, M. D., Potter, M., 2001, "Gravitational Microlensing by Low-Mass Objects in the Globular Cluster M22", *Nature* 411, 1022

Schreier, E. J., Koekemoer, A. M., Grogin, N. A., Giacconi, R., Gilli, R., Kewley, L., Norman, C., Hasinger, G., Rosati, P., Marconi, A., Salvati, M., Tozzi, P., 2001, "Hubble Space Telescope Imaging in the Chandra Deep Field-South. I. Multiple Active Galactic Nucleus Populations", ApJ 560, 127

Shupe, D.L., et al., 2005, ADASS XIV, ASP Conference Series, Vol. 347, p. 491.

Sosey,M., Wheeler, T., Sivaramakrishnan, A., 2003, "Analysis of HST Thermal Background as Seen by NICMOS+NCS", NICMOS-ISR-2003-007

Sparks, W. B., Hack, W., Clampin, M., 2001, "ACS Software Tool Development", ACS-ISR-2001-10,

Sparks, W. B., Hack, W., Hook, R. N., 2001,"Initial Implementation Strategy for Drizzle with ACS", ACS-ISR-2001-04

Trauger, J. T., Vaughan, A. H., Evans, R. W., Moody, D. C., 1995, ÄúGeometry of the WFPC2 Focal Plane Äù, in *Calibrating HST: Post Servicing Mission*, eds. A. Koratkar & C. Leitherer

Walsh, J. R., Goudfrooij, P., Malumuth, E. M., 2001, "TIS Geometric Distortion - SMOV3A Tests for CCD, NUV-MAMA and FUV-MAMA", ISR STIS-2001-02

Whitmore, B., Heyer, I., 1997, "New Results on CTE and Constraints on Flat-Field Accuracy", WFPC2-ISR-97-08

Index

U

V

W