Version 1.0
November 2000

HST Dither Handbook

SPACE TELESCOPE SCIENCE INSTITUTE

Hubble Division
3700 San Martin Drive
Baltimore, Maryland 21218
help@stsci.edu

User Support

For answers to questions, please contact the STScI Help Desk.

- **E-mail:** help@stsci.edu
- **Phone:** (410) 338-1082

World Wide Web

Information, software tools, and other resources are available on the STScI World Wide Web Dither Page:

- **URL:** http://www.stsci.edu/instruments/wfpc2/dither.html

Revision History

Version	Date	Editor
1.0	December 2000	Anton M. Koekemoer

Authors and Contributors

Anton Koekemoer, Shireen Gonzaga, Andy Fruchter, John Biretta, Stefano Casertano, J.-C. Hsu. Matt Lallo, Max Mutchler (STScI), with additional input from Richard Hook (ST-ECF), and the following STScI groups: WFPC2, STIS, NICMOS, ACS, OSG

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

Table of Contents

List of Figures

List of Tables

x ■

CHAPTER 1:

Introduction to Dithering: Why Dither?

In This Chapter...

This chapter begins with a general overview of the organization of this Handbook and its relationship to other STScI documents. This is followed by a description of dithering, together with the benefits and drawbacks of employing dithering techniques in an observing program.

Overview

The HST Dither Handbook is intended as the current, up-to-date source of information required to successfully plan, obtain, and reduce/analyze dithered observations with the primary Science Instruments onboard the Hubble Space Telescope (HST). The information in this Handbook has been drawn from the various individual Instrument Handbooks, as well as a considerable number of other separate documents available on the STScI website. A bibliography of the relevant material is presented at the end of each chapter. The information in many of these older documents is out of date and has been updated as necessary when included in this Handbook.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.

This Handbook is organized in approximately the same order as that in which an observing program would be carried out. Chapter 1 deals with general benefits and drawbacks of dithering, while Chapter 2: and Chapter 3: present information about the HST and instrument characteristics, useful for designing the observations. Chapter 4: presents a discussion of the fundamental design of the **drizzle** code that is used to combine separate dithered images into a single image. Chapter 5: is a general tutorial on using the IRAF/STSDAS **dither** package, including the **drizzle** code, and gives general advice on choosing parameters applicable to different kinds of datasets. Finally, Chapter 6: presents a selection of specific examples of using the software to analyze data drawn from real HST observing programs.

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 integer 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 a fundamentally different set of observational considerations that are beyond the scope of this document.

Benefits of Dithering

Dithering of HST observations is hardly new; indeed 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 flat field 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 reason is particularly important in the case of HST imaging, since the Wide Field Cameras (WF) of WFPC2, and Camera 3 of NICMOS (NIC3) are unable to take full advantage of the resolving power of the optics. The width of a WF pixel, at about 0.1", 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 NIC3 similarly undersamples the image over much of its spectral range. Even the smaller-scale cameras, such as the WFPC2/PC, STIS/CCD and ACS/WFC with their scales of ~0.05"/pixel, do not provide optimal sampling of the PSF over much of the spectral regime. Likewise, the STIS MAMAs do not fully sample the PSF in the UV. While the ACS/HRC will 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). Thus dithering is becoming an increasingly important technique in designing HST observations.

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 in Chapter 3: but are summarized here:

- Elimination of cosmic rays may be compromised, especially if using only a few sub-pixel offsets with just one image at each location.
- 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 whether this is significant by actually running tests of various scenarios using RPS2.
- 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 because of intra-pixel sensitivity variations.

- Too much dithering in an orbit can impose additional loads on the system both in terms of command volume needed to execute the observations and in the volume of resulting data that must be processed through the pipeline. In extreme cases these extra loads can result in lowered overall efficiency of HST observations.

Although there are a number of potential drawbacks to dithering, these are in general outweighed by the scientific benefits. However, in specific instances it might be possible that the above drawbacks are deemed too severe, for example in programs with very few available orbits, and in such cases it may be advisable not to dither.

Bibliography

Biretta, J., Wiggs, M., “General Advice on Dithering HST Observations”,

`http://www.stsci.edu/instruments/wfpc2/wfpc2_dither_all.html`

Biretta, J., Wiggs, M., “Questions About Dithering WFPC2 Observations”,

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_faq/wfpc2_dith_faq.html`

Fruchter, A. S., “Andy’s Dither Page”,

`http://www.stsci.edu/~fruchter/dither/`

Fruchter, A. S., Hook, R. N., 1997, “A Method for the Linear Reconstruction of Undersampled Images”, *PASP (submitted)*; astro-ph/9808087,

`http://xxx.lanl.gov/abs/astro-ph/9808087`

Fruchter, A. S., Hook, R. N., “Linear Reconstruction of the Hubble Deep Field”,

`http://www.stsci.edu/~fruchter/dither/drizzle.html`

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, “A Package for the Reduction of Dithered Undersampled Images”, in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/fruchtera.ps`

Koekemoer, A. M., Wiggs, M., “The WFPC2 Drizzle Page”,

`http://www.stsci.edu/instruments/wfpc2/drizzle.html`

CHAPTER 2:

HST and Instrument Characteristics

In This Chapter...

A knowledge of the capabilities and limitations of HST and its complement of imaging 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. This chapter presents a discussion of the pointing and offsetting accuracy of HST, together with its stability during each orbit and also across periods spanning multiple orbits. Also summarized in this chapter are the plate scales and geometric distortion models of the various detectors, specifically in the context of how these effects should be taken into account when designing dither strategies.

HST Pointing Accuracy and Stability

A primary 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 taken into account 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:

1. 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)
2. 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
3. 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 HDF-N and S, and monitoring campaigns of the globular clusters M22 and 47 Tuc (Programs 7615 and 8267 respectively), as well as numerous smaller GO programs. Together, these are providing a growing 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.

HST Tracking Stability at a Single Location

During each orbit, thermal variations in the telescope cause "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 2-3 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. This will manifest itself primarily as a translation of the science instrument (with some slight rotation 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 pickles, 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, and also how to extract the relevant information from these files.

Precision of Commanded Offsets

It is recommended that typical dither offsets usually be less than one arcsecond. 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 ~3 - 5 mas during a single visit with good lock, ranging up to ~10 - 15 mas from visit to visit over many days. Occasionally, however, the actual offsets can differ substantially from the commanded offsets by ~0.1 - 0.2 arcsec 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. For example, this behavior was observed in two out of nine pointings during the HDF-N campaign.

Dither patterns with offsets larger than about an arcsecond in size are generally discouraged, not only because of the chance of increased discrepancies between commanded and actual offsets, but also because the camera distortions introduce errors toward the corners, which scale in direct proportion to the size of the offset. Much larger dithers (e.g. more than several arcseconds) may also cause the telescope to acquire a different set of guide stars, in which case the pointing repeatability is essentially lost as a result of the relative positional uncertainties in the guide star catalog (~0.2 - 0.5 arcsec).

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 - 10 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 arcsec. 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, if at all possible, to provide improved relative registration between the different images.

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 generally constrained by the scheduling software 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, due to slight differences induced by thermal “breathing” variations and related effects. 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 subdominant guide star and tracks it in fine lock. The Pointing Control System 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 subdominant 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 guidestars and same roll, to determine quite accurately the actual roll change that was incurred between visits.

In Table 2.1 on page 9 we summarize the HST pointing and stability accuracy for various commonly-used observing scenarios. Once again we emphasize that these are the **typical expected values**, and that substantially larger errors can always occur as a result of uncertainties in the known positions or proper motions of guide stars, achieving fine lock on only one guide star, or unexpected interferometric peculiarities.

Table 2.1: Typical HST pointing and stability characteristics

Observing Scenario (with full lock on two guide stars)	Type of Program	Typical RMS Accuracy
Single pointing	Small programs (no dithering)	< 2 - 3 mas
Offsets within an orbit (recommend < 1 arcsec)	Small programs (with dithering)	~ 3 - 5 mas
Re-acquisition for contiguous orbits in the same visit	Medium-sized programs (e.g., < 5 orbits per target)	5 - 10 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

Detector Plate Scales and Geometric Distortion

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 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).

Summary of Detector Plate Scales

The detector pixels themselves are in general not precisely square, and this can further impact the relative shifts along the x and y axes for large dithers. The current best-determined values of the pixel dimensions are stored in the Science Instrument Aperture File (SIAF), specifically defined for a single pixel at a “reference” location on the chip. The geometric distortion model appropriate to a given detector then allows the calculation of pixel sizes at other locations on the chip.

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 Table 2.2 on page 11, and have been drawn from the current (Cycle 10) version of the SIAF. 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 are nominally orthogonal to one another; however, analysis of on-orbit astrometric measurements is currently underway to verify the extent to which this is true for the various instruments. The last column in the table lists the approximate maximum distortion, i.e. the change in linear dimensions of pixels near the corners of the detector relative to the central pixel. 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 Apertures and Pointing Page:

`http://www.stsci.edu/instruments/observatory/taps.html`

Table 2.2: Pixel scales of the primary HST instruments (from the current SIAF file).

Instrument	Camera/ Aperture	x-scale ("/pixel)	y-scale ("/pixel)	beta-x (deg)	beta-y (deg)	Maximum Distortion
WFPC2	PC1 (800^2)	0.045528	0.045507	134.908	224.908	1%
	WF2 (800^2)	0.099500	0.099596	224.388	314.388	2%
	WF3 (800^2)	0.099573	0.099480	314.698	44.698	2%
	WF4 (800^2)	0.099539	0.099635	45.258	135.258	2%
NICMOS	NIC1 (256^2)	0.043200	0.043026	225.312	315.312	1%
	NIC2 (256^2)	0.076002	0.075320	224.507	314.507	0.2%
	NIC3 (256^2)	0.203758	0.203012	224.851	314.851	0.5%
STIS	CCD (1024^2)	0.050763	0.050777	45.056	135.056	0.3%
	NUV (1024^2)	0.024750	0.024750	45	135	1%
	FUV (1024^2)	0.024742	0.024742	45	135	0.7%
ACS	WFC (4096^2)	0.049	0.049	270	0	8%
	HRC (1024^2)	0.028	0.025	270	0	11%
	SBC (1024^2)	0.033	0.030	270	0	11%

Please note that the values for ACS in this table are preliminary, being derived purely from the pre-flight optical models of the instrument, and are subject to change pending on-orbit measurements.

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 reimaging 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 k in x, in y, and including all their cross-terms:

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

Explicitly, these distortion polynomials expand out into:

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

The distortion coefficients characterizing each instrument are stored in the form of tables that are available from STScI, and are also incorporated into the IRAF/STSDAS **dither** software used to analyze the data.

WFPC2

The most widely used distortion models for WFPC2 are polynomials of the above form expanded up to third-order (Gilmozzi et al. 1995; Holtzman et al. 1995; Trauger et al. 1995), thus characterizing the spatial distortion in terms of 10 coefficients each for X and Y, calculated relative to a given reference pixel. The primary difference between these models is that the Gilmozzi and Holtzman models are determined only at one wavelength (555nm), the Holtzman model being defined in terms of a single "meta-frame" coordinate system centered at the vertex of the WFPC2 pyramid. The Trauger model, on the other hand, includes wavelength dependence resulting from the MgF_2 CCD window and is derived separately for each of the four chips, relative to its central pixel. The r.m.s. accuracy of these models is of the order ~0.1 pixel in the PC and also in the WF chips, although localized deviations up to 0.2-0.3 pixels are also present in some cases.

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 cameras NIC1 and NIC2 the distortion is less than 1 pixel at the chip corners relative to the center and appears to be best modeled by a quadratic. The NIC3 camera has more substantial distortion at its nominal out-of-focus position, and the quality of the 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*, and have been used to derive the current NIC3 drizzle geometric distortion coefficient files.

The NICMOS plate scale is also more variable than the other instruments, particularly during the early part of its lifetime, due to pathlength changes introduced by the NICMOS dewar anomaly. More detailed information on these and other geometric effects are all available from the NICMOS WWW pages:

`http://www.stsci.edu/instruments/nicmos/`

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 2000). 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.

ACS

The ACS/WFC is expected to be 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 values listed in Table 2.2 on page 11 are those derived from pre-flight ray-tracing and other optical modelling. Work is currently in progress to define the distortion in terms of polynomials similar to those used for the other instruments, although orders higher than cubic will likely be used.

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. The most effective way to ensure that satisfactory subsampling is achieved across the entire field of view with ACS will be to obtain images at several different dither pointings.

Bibliography

Biretta, J., “Dithering: Relationship Between POS TARGs and CCD Rows/Columns”,

http://www.stsci.edu/instruments/wfpc2/Wfpc2_memos/dithering.html

Biretta, J., et al., 2000, “WFPC2 Instrument Handbook, Version 5.0", (Baltimore: STScI),

http://www.stsci.edu/instruments/wfpc2/Wfpc2_hand/wfpc2_handbook.html

Biretta, J., Wiggs, M., “General Advice on Dithering HST Observations”,

http://www.stsci.edu/instruments/wfpc2/wfpc2_dither_all.html

Biretta, J., Wiggs, M., “Questions About Dithering WFPC2 Observations”,

http://www.stsci.edu/instruments/wfpc2/Wfpc2_faq/wfpc2_dith_faq.html

Böker, T., et al., 2000, “NICMOS Instrument Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/instruments/nicmos/documents/handbooks/instrument/v4/ihb.html

Bowers, C., Baum, S., 1998, “Plate Scales, Anamorphic Magnification and Dispersion: CCD Modes”, *ISR STIS 98-23*,

http://www.stsci.edu/instruments/stis/documents/isrs/9823/9823.pdf

Cox, C., 1994, “The WFPC2 Scales and Alignments”, *SOB-94-10-21*

http://www.stsci.edu/instruments/observatory/PDF/wfpc2fit.pdf

Cox, C., Ritchie, C., Bergeron, E., Mackenty, J., Noll, K., 1997, “NICMOS Distortion Correction”, *ISR OSG-CAL-97-07*,

http://www.stsci.edu/instruments/observatory/PDF/nicmos.distort.pdf

Dickinson, M., et al., 1999, “NICMOS Data Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/cgi-bin/NICMOS/si.pl?nav=documents:handbooks&sel=id:512

Gilmozzi, R., Ewald, S., Kinney, E., 1995, “The Geometric Distortion Correction for the WFPC Cameras”, *ISR WFPC2 95-02*,

http://www.stsci.edu/instruments/wfpc2/Wfpc2_isr/wfpc2_isr9502.html

Hartig, G., Kinney, E., Hodge, P., Lallo, M., Downes, R., 1999, “STIS Coordinate System Orientation and Transformations”, *TIR STIS 99-02*

Holtzman, J., et al., 1995, “The Performance and Calibration of WFPC2 on the Hubble Space Telescope”, *PASP 107, 156*

Lallo, M., Cox, C., Lupie, O., 1998, “A Few Words on Pointing and Jitter File Accuracies”, *OSG MEMO 1998-09-29*

Lallo, M., Lupie, O. Toth, B., and STScI/OSG, "Pointing Calculations and Sources of Error", in: "On-Line Observation Logs Documentation",

http://www.stsci.edu/instruments/observatory/obslog/OL_7.html

Leitherer, C., et al., 2000, "STIS Instrument Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/instruments/stis/documents/ihb/stis_cy10_ihb_4.1.pdf

Malumuth, E. M., Bowers, C. W., 1997, "Determination of Geometric Distortion in STIS Images", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

http://icarus.stsci.edu/~stefano/CAL97_PAPERS/malumuth.ps

Trauger, J. T., Vaughan, A. H., Evans, R. W., Moody, D. C., 1995, "Geometry of the WFPC2 Focal Plane", in *Calibrating HST: Post Servicing Mission*, eds. A. Koratkar & C. Leitherer

Walsh, J. R., Goudfrooij, P., Malumuth, E. M., 2000, "STIS Geometric Distortion - SMOV3A Tests for CCD, NUV-MAMA and FUV-MAMA", *STIS ISR*, in preparation

CHAPTER 3:

Observational Dither Strategies

In This Chapter...

This chapter discusses observational dithering strategies for WFPC2, NICMOS and STIS. It begins with a section aimed at helping observers decide whether or not to obtain dithered observations. This is followed by more in-depth discussions of various dither strategies that are common to all the HST instruments, followed by sections describing additional issues specific to each instrument. The information in this chapter is a generic summary of dither-related material in the Instrument Handbooks and Phase II documentation, and can therefore be consulted in conjunction with these if more detailed information is required.

To Dither or not to Dither, and if so, How

This is the question that must be addressed when designing observational imaging programs with HST. Here we provide general recommendations aimed at guiding observers toward different strategies, depending upon the type of science and the level of detail that will be extracted from the data.

Dithering for Parallel Images

It should be noted that it is in general extremely challenging to design optimal dithers simultaneously for primary and parallel instruments, due to their large angular separation and generally different pixel scales. Thus, an optimal dither pattern for the primary science instrument generally corresponds to non-optimal dithers for the parallel instrument.

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.
- *Crucial PSF Modelling:* Dithering may introduce additional complications due to slightly inexact spacecraft offsets, so some observers prefer to obtain all images of a target at a single pointing location.

Simple Dithering

For programs up to about one orbit per target, at least 2 - 3 exposures should be obtained to facilitate cosmic ray rejection.

- 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 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.

Full Dithering

If improved spatial sampling is desired on programs of two or more orbits per target/filter combination, then at least a "full" 4-point dither is recommended (e.g., providing 1/2 pixel sub-sampling along both detector axes), preferably with multiple exposures per dither location. For very deep programs (e.g., 5-10 orbits or more per target), more sophisticated dither patterns can be considered, such as the "random" pattern of the HDF-S.

PLEASE NOTE these are GUIDELINES only, in no way intended as solid rules. There will likely be science programs that do not fit exactly into any one of the above categories, or that have different requirements. In all cases we strongly recommend that the observational strategy be discussed thoroughly with the STScI Contact Scientist for each particular HST program.

More Detailed Considerations

Issues involved in Dithering

The primary considerations in designing a dithered observational program are cosmic rays, hot pixels, spatial undersampling, and trading signal-to-noise for the ability to recognize and deal with these issues. Careful consideration must also 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 overhead. The optimal strategy chosen will ultimately depend upon carefully weighing all these issues against one another, and also against the scientific questions involved: is the underlying structure totally unknown, is spatial resolution of paramount importance, is photometric accuracy the most crucial aspect, etc.?

1. *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.
2. *Hot Pixels:* There are three ways to deal with hot pixels: recalibrate using "dark frames" that bracket the date of the observation (presently obtained weekly), obtain a second image (or pair of images to best reject cosmic rays) shifted by a small amount spatially (e.g. about 5 pixels), or use an IRAF/STSDAS task such as **warmpix** to filter out the obvious hot pixels. Since some hot pixels are variable on very short timescales, the most robust strategy is to obtain multiple images.
3. *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 offsetting 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.

4. *Photometric Accuracy:* The instruments generally have variations in the sensitivity across each individual pixel. Since the PSF is under-sampled, this can complicate the photometric analysis of dithered images. Thus, programs requiring maximal accuracy in photometry may in some cases not benefit greatly from dithering.

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

If integer-pixel dithers are all that is required, i.e. to ameliorate the effect of hot pixels, then 2 or 3 different locations should be sufficient to guarantee that sources of interest are not completely unrecoverable as a result of falling on hot pixels.

The remainder of this discussion focusses 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 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+1/2) pixels where n is an arbitrary integer. This scenario is only really relevant to STIS longslit spectroscopy, if it is desirable to improve the subsampling along the (spatial) slit direction.
- The general description of a two-point dither in imaging data is to obtain one image at the original pixel position of (0,0), and a second image offset by half a pixel in both x and y, i.e. at (n+1/2, m+1/2), where n and m are arbitrary integers. This produces a substantial increase in information over non-dithered data. Setting n and m to be a few pixels (e.g., up to 5-10) will also allow hot pixels to be moved by sufficient amounts to reduce their impact on objects of interest.
- There are sometimes 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 somewhat controversial. This is essentially because there is no natural way to tile the plane using three placements of the rectangular CCD grid, and this is why we generally recommend a two or four point dither. A calculation performed by A. Fruchter suggests that if the observer wants to do a three point dither, the best sub-pixel placements are along the diagonal at (0,0), (1/3,1/3) and (2/3,2/3) pixel offsets. (The symmetric diagonal works just as well, of course.)

- The four-point dither is a more sophisticated option, yielding a total of 4 images that are offset from one another by half pixels in x, in y, and in both x and y - i.e. (0, 0), (0, 1/2), (1/2, 0), (1/2, 1/2) - including also possible additional integer shifts to ameliorate 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.

For most medium-sized programs (e.g., a few orbits per target/field combination), very little additional information is gained from using more than four positions, if the standard four point dither is used, and if the telescope has successfully executed the dither. Unfortunately, slight non-repeatable uncertainties in HST offsets can produce dithers that do not exactly correspond to those requested. For medium-sized programs, one option is simply to accept this and use the non-optimal offsets as inputs to the **drizzle** software when combining the images.

For large programs, another option involves substantially increasing the number of dither positions. For example, the HDF-S observations employed a set of "random" dither positions. This had the double advantage of being less susceptible to random offsetting errors, as well as providing better subsampling for the parallel observations. In practice, then, increasing the number of dither positions will better ensure that the sub-pixel phase space is well sampled by the final dataset.

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 form part of the data products when requested from the HST archive. The *HST Data Handbook* contains further details on the jitter files and other data products, and how to extract information from them. 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 data with shifts in rotation as well as position.

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 current version of the **drizzle** software that is incorporated within the **dither** package (STSDAS release of Summer 2000). 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, we still recommend obtaining at least 2 - 3 images at each dither location. Not only will this assure more robust cosmic ray removal, but multiple images per dither location also help reduce errors introduced by inexact spacecraft offsets.

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 (of 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 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 2x2 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 POS TARG axes run exactly along the CCD rows and columns on whichever aperture is specified for the observations. For example, if aperture WF3 is specified, the POS TARG axes will run *exactly* along the rows and columns on WF3, and will run only approximately along the rows and columns of the other CCDs. Note that if WFALL is specified, then the rotation for WF3 is used since WF3 is the reference chip for the WFALL aperture.

The CCD rotation misalignments lead to errors when attempting to dither by certain pixel amounts. For small dithers (<0.3 arcsec) 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. 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. A POS TARG = 1.993, 0.000 arcsecond dither in X on WF3 would cause spurious motion in Y of 0.17 pixel on WF4, due to the rotation.

STIS

The concept of dithering as applied to STIS observations is multifaceted, since STIS can be used to obtain either images or spectra, and since 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 hot pixels or uncertainties of 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 will require 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 have nearly the same spatial scale as those obtained with the WFPC2/PC camera, so that the gain 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. While it is possible to detect and reject cosmic rays when combining individually dithered CCD images, this is not the best strategy for STIS. The STIS/CCD has lower read noise and less readout time overhead than those in WFPC2, so for most observations the best and most reliable cosmic ray rejection strategy would involve obtaining multiple CR-SPLIT images at each dither position. This approach will simplify the data reduction.

Spectroscopic-Mode Dithering

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

- Stepping along the slit direction:
  - average over pixel-to-pixel flat-field uncertainties;
  - facilitate removal of hot and cold pixels (e.g., by using integer pixel steps);
  - subsample the spatial PSF along the slit (by sub-pixel steps along the slit);
- Stepping along the dispersion direction, perpendicular to the spatial axis of the slit:
  - subsample the spectral line spread function by stepping a fraction of a pixel along the dispersion direction;
  - map out a two-dimensional region of the sky by using larger step sizes, for example 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 stepping, or dithering, effectively 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 effectively cause the counts from any output pixel to have been sampled at many independent detector pixels in the dispersion direction (for exposures comparable to an orbit visibility period and targets well away from the orbital pole of HST).

In slitless or wide-slit mode, stepping along the dispersion would allow independent solutions for spectrum and flat-field, but at a cost of lower spectral resolution. This technique is not likely to be useful unless 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 available-but-unsupported contingent of ‘‘fpsplit" 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, too, is likely to work well only if the constituent spectra have a good S/N ratio. The performance of the ‘‘fpsplit" slits, the techniques for using them, and the ability to execute a shift in wavelength only, have yet to be evaluated. In the echelle modes Doppler smoothing will generally provide for increased S/N, thus lowering the need for fpsplits.

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. Note that in employing 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. Note that the use of ‘‘high-res" (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.

NICMOS

A set of pre-defined patterns has been created for NICMOS to allow an easy implementation of both integer-pixel and sub-pixel dithering. 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 persistent signal that can take up to a full 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 described in Chapters 5 and 6.

- *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 perhaps for the NIC3 camera, and, 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 and the shortest wavelengths of NIC1 (below 1.0 microns) and NIC2 (below 1.75 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, especially for Cameras 1 and 2, 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 4 dither positions are advisable. For NIC3, the telescope pointing uncertainty corresponds to 0.1 pixels shift only, and 4 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 the telescope (variable) thermal emission. 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 2 or 4 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, see NICMOS update) is that the target will remain on the chip for all observations, increasing the efficiency of the observation.

Dithering NICMOS observations offers many potential advantages as described above. However, there are a number of disadvantages linked to dithering that an observer should consider before choosing to dither NICMOS observations:

- 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 2 ACCUM mode exposures per position should be obtained to help cosmic ray removal BEFORE image reconstruction. Note that, 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 dither.
- Rapid dithering can impose additional load on the full system in terms of command volume needed to execute the observations, overheads for science data buffer dump management and in the volume of data that must be processed through the pipeline. In extreme cases 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 shortward of 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.

Bibliography

Biretta, J., "Dithering: Relationship Between POS TARGs and CCD Rows/Columns",

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_memos/dithering.html`

Biretta, J., et al., 2000, "WFPC2 Instrument Handbook, Version 5.0", (Baltimore: STScI), http://www.stsci.edu/instruments/wfpc2/Wfpc2_hand/wfpc2_handbook.html

Biretta, J., Wiggs, M., "General Advice on Dithering HST Observations",

http://www.stsci.edu/instruments/wfpc2/wfpc2_dither_all.html

Biretta, J., Wiggs, M., "Questions About Dithering WFPC2 Observations",

http://www.stsci.edu/instruments/wfpc2/Wfpc2_faq/wfpc2_dith_faq.html

Böker, T., et al., 2000, "NICMOS Instrument Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/instruments/nicmos/documents/handbooks/instrument/v4/ihb.html

Casertano, S., et al., 2000, "WFPC2 Observations of the Hubble Deep Field - South", *AJ (in press)*; astro-ph/0010245,

http://xxx.lanl.gov/abs/astro-ph/0010245

Dickinson, M., et al., 1999, "NICMOS Data Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/cgi-bin/NICMOS/si.pl?nav=documents:handbooks&sel=id:512

Leitherer, C., et al., 2000, "STIS Instrument Handbook, Version 4.0", (Baltimore: STScI),

http://www.stsci.edu/instruments/stis/documents/ihb/stis_cy10_ihb_4.1.pdf

CHAPTER 4:

How the "Drizzle" Algorithm Works

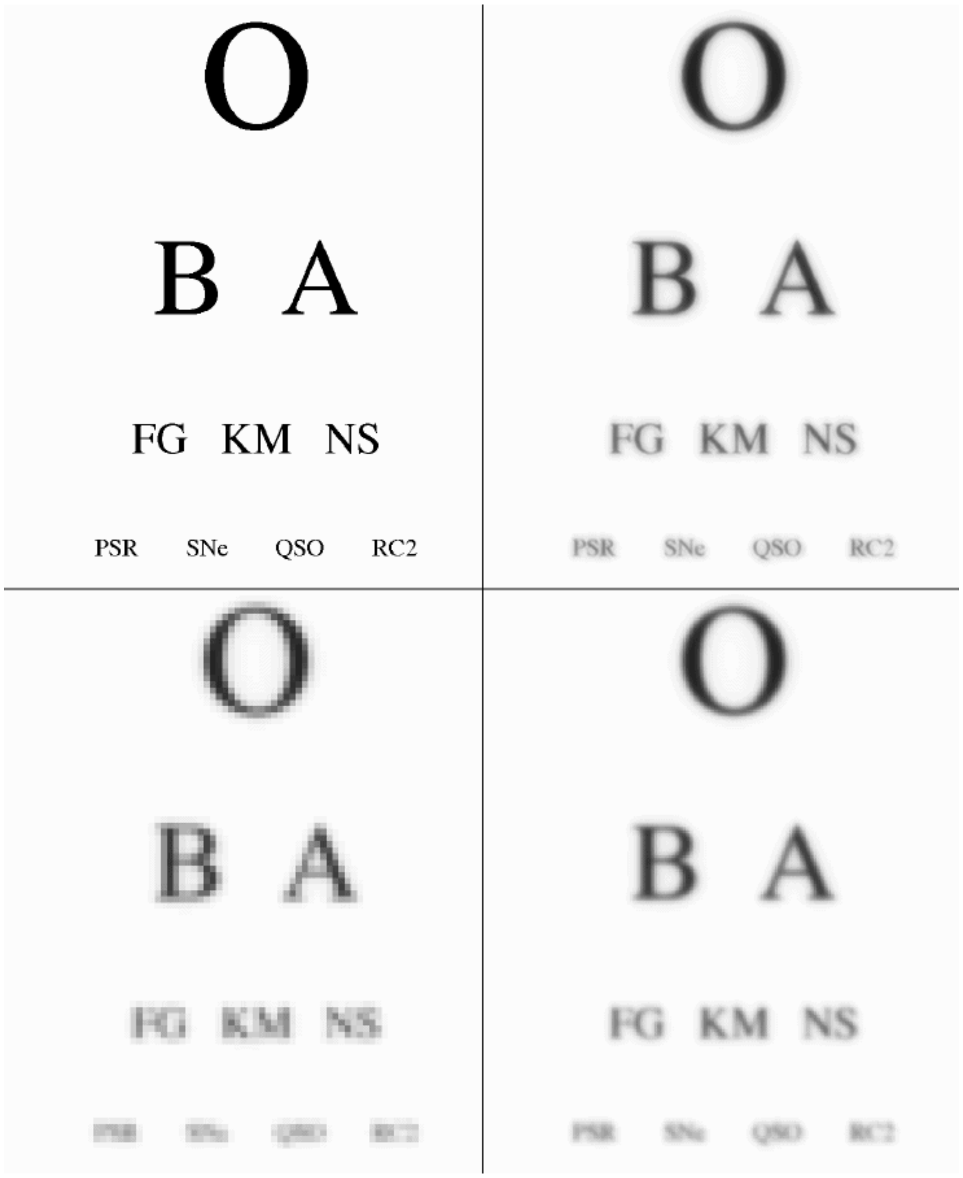
In This Chapter...

This chapter presents an overview of the general concepts involved in image reconstruction and restoration, together with a discussion on the **drizzle** algorithm and its relationship to other image reconstruction techniques. This includes a description of the ways in which **drizzle** can be used to improve image quality, as well as its limitations and its overall effect on the accuracy of scientific measurements such as photometry, astrometry and noise properties of the resulting images.

Introduction

While much high spatial frequency information in the image is permanently lost by smearing with the response of the detector pixels, the quality of the image can nevertheless 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 in Figure 4.1 on page 30. 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 4.1: Representation of the effects of image convolution and subsampling.



The upper left corner of this Figure (from Fruchter and Hook 1997) shows the "true image", i.e, the image one would see with an infinitely large telescope. The upper right shows the image after convolution with the optics of the Hubble Space Telescope and the WFPC2 camera. The lower left shows the image after sampling by the WFPC2 CCD, and the lower right shows a linear reconstruction of dithered CCD images.

Much of the information lost to undersampling can be recovered, as shown in the lower right of Figure 4.1 on page 30, displaying the image recovered using one of the family of techniques referred to as "linear reconstruction." However, the simple implementations of these techniques generally introduce additional blurring due to convolution with the pixel shape. This effect can be seen directly in the present example by comparing the upper and lower right-hand images in Figure 4.1 on page 30 - the deterioration in image quality between these two images is due entirely to convolution of the image with the WF pixel.

Image Restoration and Reconstruction Techniques

There are two conceptually different approaches to recovering spatial resolution in images, and these are broadly referred to as image "reconstruction" and "restoration" (the latter also known as "deconvolution"). Reconstruction refers to recreating the image after it has been convolved by the instrumental PSF (which includes the pixel response function). Restoration refers to removing the effects of the PSF on the ideal image, attempting to improve upon the image resolution by enhancing high frequency components which had been suppressed by the optics or the detector (see Hanisch and White 1994 for a review). Non-linear image restoration may already be familiar to observers, as it was a popular means of partially suppressing the effects of spherical aberration in the images obtained with WF/PC-1.

The primary aim of these techniques is to recover image resolution while preserving the signal-to-noise ratio. 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, in addition to being unable to handle large dithers, the present implementation of this technique 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 WFPC2 field of view. And 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.

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 4.1 on page 30 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.

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.

Drizzle

In response to the limitations of the two techniques just described, an improved method known formally as variable-pixel linear reconstruction, 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**, 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** 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 becomes important, and takes into account missing data resulting from cosmic rays and bad pixels. It can also accommodate non-uniform subsampling across the field, which is caused by changes in pixel size resulting from geometric distortion.

The Drizzle Method

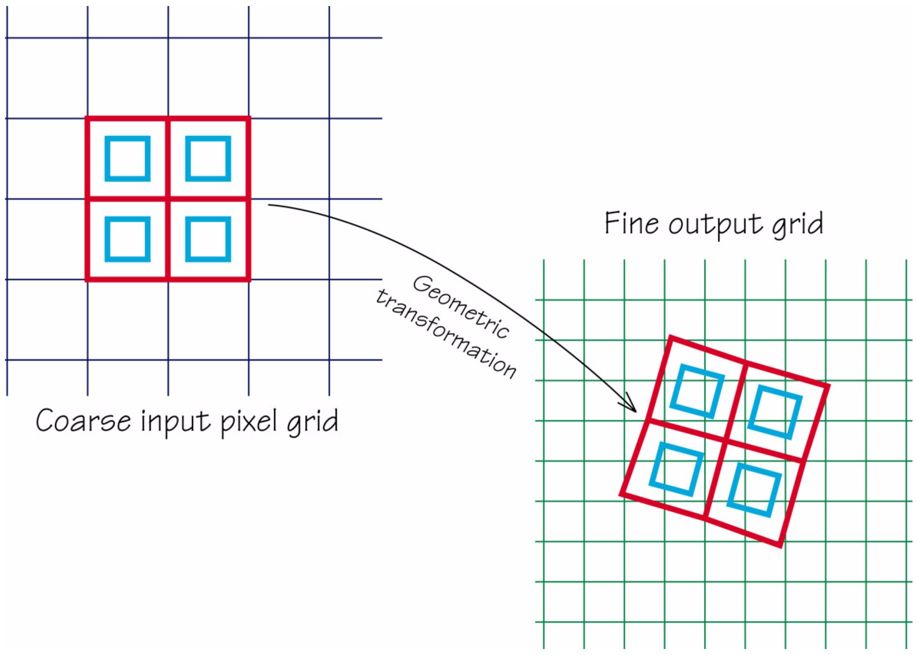
The **drizzle** algorithm is conceptually straightforward. Pixels in the original input images are mapped into pixels in the subsampled output image, taking into account shifts and rotations between images and the optical distortion of the camera. However, in order to avoid convolving the image with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

The new shrunken pixels, or "drops", rain down (or "drizzle") upon the subsampled output image, as shown in Figure 4.2 on page 34. A second parameter, **scale**, allows the user to specify the size of the output pixels relative to the input 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 (which were specified by setting `scale=0.4`).

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.

The drop size is controlled by the parameter **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). 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.

Figure 4.2: Schematic representation of how drizzle maps input pixels onto the output image.



A schematic representation of drizzling (from Fruchter and Hook 1997). The input pixel grid (shown on the left) is mapped onto a finer output grid (shown on the right), taking into account shift, rotation and geometric distortion. The user is allowed to "shrink" the input pixels to smaller pixels, which we refer to as "drops" (the inner squares). In this particular example, the central pixel in the output grid receives no flux from any pixel in the input image, hence the dropsize shown here would only be appropriate if more input images were to be drizzled onto the output.

Weight Maps and Correlated Noise

When images are combined with **drizzle**, a weight map can be specified for each input image (generally containing information on bad pixels in the image). When the **drizzle** task generates the final output image, it also creates an output weight map that combines the information from all the input weights using the following algorithm. When a drop with value i_{xy} and user-defined weight w_{xy} is added to an image with pixel value I_{xy}, weight W_{xy}, and fractional pixel overlap $0 < a_{xy} < 1$, the resulting value of the image I'_{xy} and weight W'_{xy} is:

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

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

This algorithm has a number of advantages over standard linear reconstruction methods. Since the area of the pixels scales with the Jacobian of the geometric distortion, **drizzle** preserves both surface and absolute photometry. Therefore flux can be measured using an aperture whose size is independent of position on the chip. As the method anticipates that a given output pixel may receive no information from a given input pixel, missing data (due for instance to cosmic rays or detector defects) do not cause a substantial problem, as long as there are enough dithered images to fill in the gaps caused by these zero-weight input pixels. Finally, the linear weighting scheme is statistically optimum when inverse variance maps are used as the input weights.

The output pixels in the final drizzled image are not independent, therefore the noise in adjacent pixels is correlated (although generally to a lesser extent than in pure shift-and-add). The correlated noise can in principle be fully described by creating a correlation image, but in general the implementation of such schemes is very complex when images are shifted at sub-pixel levels. 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 **scale** (expressed in terms of the ratio of the output pixel size to that of the input pixel) can be calculated in the following way (as described in more detail in Casertano et al. 2000):

$$Var = [(f(D+B)/g) + \sigma^2]/(f^2t^2)$$
$$W = 1/(Var \times scale^4)$$

where D and B are the counts per pixel (in DN) due to the dark current and background respectively, averaged over the entire image, while t is the exposure time (in seconds), g is the gain of the detector, and σ is the read-noise in DN/pixel. The quantity f represents the inverse flat field, corresponding to the way in which the WFPC2 "flat field" reference files are defined.

Observers are encouraged to examine the more detailed descriptions presented in Fruchter and Hook (1997, 2000) and Casertano et al. (2000). An alternative, more empirical, approach toward estimating the importance of the noise correlation in drizzled images is to block-average the final image with a number of different block sizes and examine the resulting sky noise as a function of block size. If the image is largely covered by astronomical objects, or a more detailed understanding of the noise is required, then the noise characteristics can be simulated in more detail by drizzling artificial noise images together.

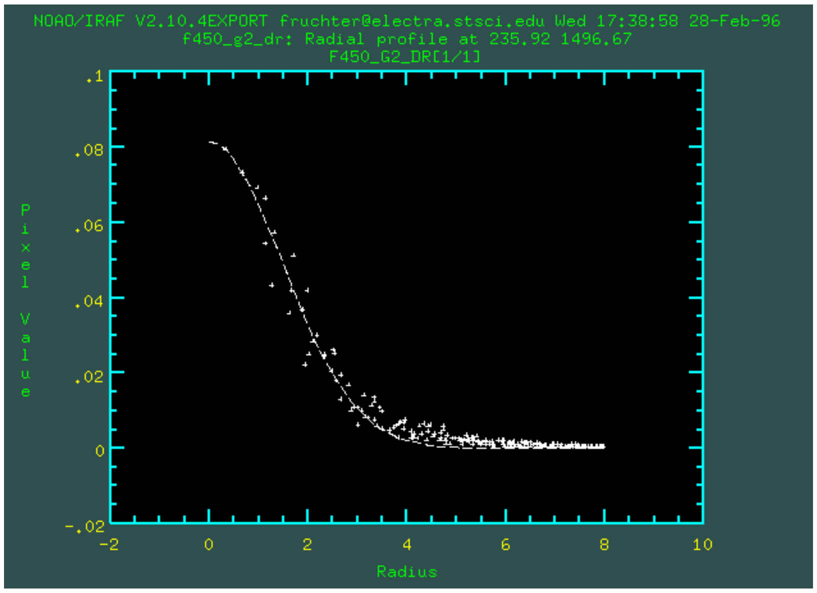
Image Fidelity

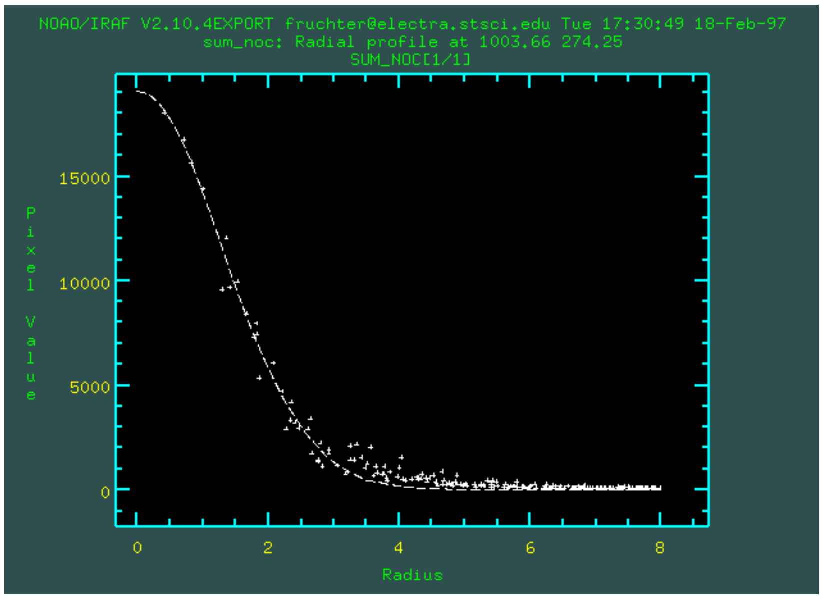
In the **drizzle** algorithm, the weight of an input pixel in the final output image is independent of its position on the chip. Therefore, if the dithered images do not uniformly sample the field, the "center of light" in an output pixel may be offset from the center of the pixel, and this offset may 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.

This effect is seen in the HDF-N images, where some pointings were not at the requested position or orientation. In Figure 4.3 on page 37 are shown two PSFs, which are 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).

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

Figure 4.3: The effect of drizzling upon the image PSF quality.





A comparison of two PSFs from the HDF-N (Fruchter and Hook 1997). The upper PSF is taken directly from the HDF-N F450W drizzled image. The PSF 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. The lower PSF 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.

Photometric Accuracy

The WFPC2 optics geometrically distort the images: pixels at the corner of each CCD subtend less area on the sky than those near the center. Flat fields 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 four times sub-sampled grid of 19×19 artificial stellar PSFs using the TinyTIM WFPC PSF modelling code. The stars were also 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 2×2 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.)

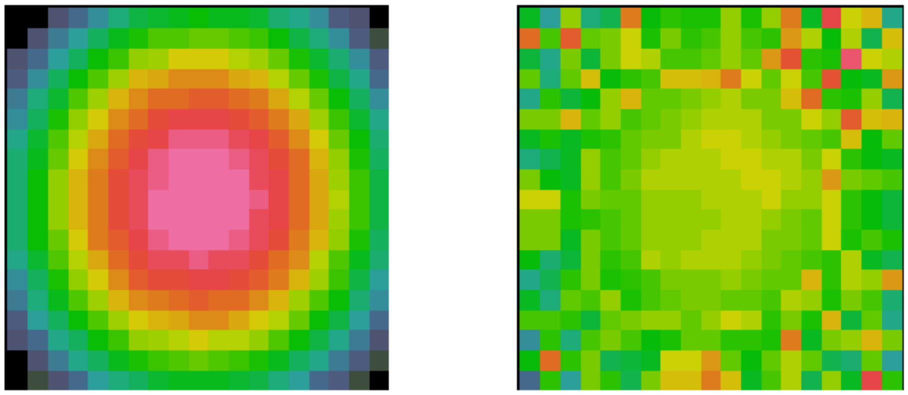
These four images were then combined using **drizzle** with typical parameters (output pixel `scale=0.5` of the original WF2 pixels, and a drop size with `pixfrac=0.6`). The geometric distortion of the chip was removed during drizzling using the polynomial model of Trauger et al.

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 4.4 on page 39, where the photometric measurements of the 19×19 stars are represented by a 19×19 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. The image on the right displays the results of aperture photometry on the 19×19 grid of stars after drizzling, showing that 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.

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 used in creating Figure 4.3 on page 37. 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 < 0.015 mags.

Figure 4.4: Improvement in photometry as a result of geometric distortion correction by drizzle.



The effect of geometric distortion on photometry (from Fruchter and Hook 1997). The pixels in the two images represent the photometric values of a 19×19 grid of stars on the WF chip (thus one pixel for each star). 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 flat field 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.

Astrometric Accuracy

Tests have also been carried out to characterize the astrometric accuracy of **drizzle** (Fruchter and Hook 1997). The stellar images described in the previous section were again drizzled using the shifts obtained in the HDF WF2 F814W images, setting `scale=0.5` and `pixfrac=0.6`. Both uniform weight files and cosmic ray masks were used. The positions of the drizzled stellar images were then determined with the IRAF **imexam** task, 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.

Bibliography

Biretta, J., Wiggs, M., "General Advice on Dithering HST Observations",

http://www.stsci.edu/instruments/wfpc2/wfpc2_dither_all.html

Biretta, J., Wiggs, M., "Questions About Dithering WFPC2 Observations",

http://www.stsci.edu/instruments/wfpc2/Wfpc2_faq/wfpc2_dith_faq.html

Casertano, S., et al., 2000, "WFPC2 Observations of the Hubble Deep Field - South", *AJ (in press)*; astro-ph/0010245,

http://xxx.lanl.gov/abs/astro-ph/0010245

Fruchter, A. S., "Andy's Dither Page",

http://www.stsci.edu/~fruchter/dither/

Fruchter, A. S., "The Hubble Deep Field - Drizzling"

http://www.stsci.edu/ftp/science/hdf/combination/drizzle.html

Fruchter, A. S., "The Hubble Deep Field - Image Registration and Combination",

http://www.stsci.edu/ftp/science/hdf/combination/combination.html

Fruchter, A. S., Hook, R. N., 1997, "A Method for the Linear Reconstruction of Undersampled Images", *PASP (submitted)*; astro-ph/9808087,

http://xxx.lanl.gov/abs/astro-ph/9808087

Fruchter, A. S., Hook, R. N., "Linear Reconstruction of the Hubble Deep Field",

http://www.stsci.edu/~fruchter/dither/drizzle.html

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, "A Package for the Reduction of Dithered Undersampled Images", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

http://icarus.stsci.edu/~stefano/CAL97_PAPERS/fruchtera.ps

Fruchter, A. S., Mutchler, M., "Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration",

`http://www.stsci.edu/~fruchter/dither/ditherII.ps`

HDF-S Team, "The Hubble Deep Field South - Data reduction / Technical information",

`http://www.stsci.edu/ftp/science/hdfsouth/hdfs.html`

Mutchler, M., Fruchter, A. S., 1997, "Drizzling Dithered WFPC2 Images - A Demonstration", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/mutchlerm.ps`

CHAPTER 5:

Using "Drizzle" - General Overview

In This Chapter...

This chapter presents a general tutorial on using the tasks in the IRAF/STSDAS **dither** package, including the **drizzle** program itself, together with related scripts that are intended to help with creating a final drizzled image from a set of dithered input images.

Introduction to the Dither Package

In order to effectively use the **drizzle** algorithm, one must know the offset and rotation between input images. Therefore the STSDAS **dither** package has been developed, to assist observers through the entire process of combining dithered images. The **dither** package currently contains the following tasks:

stsdas.analysis.dither:

- **avshift** - Determines the shifts between two sets of 4-group WFPC2 images by averaging the results obtained for each of the groups (after adjusting for the rotation angles between the four groups). **avshift** can also be used to estimate the rotation angle between two different WFPC2 images, when the rotation angle is a small fraction of a degree.

- **blot** - Samples images using interpolation (reverse "drizzling"). This task applies the appropriate geometric distortion to transform a drizzled image onto an output image that resembles the original input images. This is an essential component of the suite of tasks that remove cosmic rays from singly-dithered images.
- **crossdriz** - Cross-correlates two images, after some preprocessing which includes trimming, drizzling to remove geometric distortion or rotation, and coarse removal of cosmic rays. **crossdriz** will also solve for potential rotation angle difference between the images.
- **deriv** - Takes the absolute derivative of an image.
- **driz_cr** - Creates cosmic ray mask based on dithered data.
- **drizzle** - Performs linear image reconstruction using “drizzling.”
- **dunlearn** - Resets all task parameters to default values.
- **gprep** - Transforms shifts for one WFPC2 chip into parameters for mosaicing all four chips.
- **imextreme** - Locates the maximum and minimum pixels in an image.
- **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.
- **mask_head** - Attaches mask names to image headers.
- **offsets** - Cross-correlates all four images in a WFPC2 image, creating output cross-correlation images that can then be used by the **shiftfind** task to determine relative shifts between images. **offset** employs the task **crossdriz** to perform the cross-correlation.
- **precor** - Determines regions of the image containing astrophysical objects and nulls the remainder of the image, thereby substantially reducing the effect of cosmic rays and chip defects on the offset measurement. The output from **precor** is only used for offset determination and not final image creation.
- **rotfind** - Performs a fit for the rotation angle between two images, using a set of cross-correlation images. **rotfind** is called when **crossdriz** has been used to loop over a range of test rotation angles between two images.
- **shiftfind** - Locates the peak in a cross-correlation image and fits for sub-pixel shift information. The search region and details of the fitting can be adjusted by the user.

- **sky** - Processes sky with "**crrej**-like" algorithm.
- **cdriz** - Pset for the **offsets** task.
- **dq** - Pset for the **sky** task.
- **wfpc2_chips** - Pset for **avshift** and **gprep** tasks.

Although a detailed exposition of all these tasks is beyond the present review, here we describe a number of the **dither** tasks and typical results from their use.

The major engine for determining the offset between two images in the dither package is cross-correlation. The following equations explain the reason for the choice of this algorithm. Assume that one has a known template image, $T(x_{i,j})$ and an observed image, $I(x_{i,j})$, and one wishes to know the offset between the observed and template images. Then:

$$\sum_{i=1,j=1}^{N,N} (I(x_{i,j}) - T(x_{i,j}))^2 =$$

$$\underbrace{\sum_{i=1,j=1}^{N,N} I(x_{i,j})^2 + \sum_{i=1,j=1}^{N,N} T(x_{i,j})^2}_{\text{These terms are invariant under shifts and rotations.}} - 2\sum_{i=1,j=1}^{N,N} I(x_{i,j})T(x_{i,j})$$

Thus minimizing the difference of the sum of squares under shifts and rotations of $I(x_{i,j})$ is equivalent to maximizing the cross-correlation between the two images. If the errors in the images are Gaussian, then this is equivalent to minimizing χ^2. In practice, of course, images are a mixture of Gaussian and Poisson noise as well as, in many cases, very highly skewed noise additions such as cosmic rays and hot pixels. These can be largely removed by a task called **precor**. This task separates objects from hot pixels and cosmic rays, by determining whether the fraction of pixels above the sky in a given box is above a user-defined threshold. **precor** is not perfect, but is able to distinguish the vast majority of these defects from true objects. The regions of the sky-subtracted images containing objects are kept; other areas are set to zero. The image can also be converted from counts to signal-to-noise ratio, thus removing the distinction between Gaussian and Poisson noise in the minimization of χ^2 (in practice this step also appears to help reduce the errors introduced by the spatial undersampling of images).

After the cross-correlation image has been created, **shiftfind** determines the shift by locating the peak of the image. In the case of a single image (with no rotation) one is now done. However, in the case of WFPC, all four chips have been shifted (and rotated) by the same physical amount, and therefore one can obtain a better estimate of the shift. The script **avshift** performs this task, taking into account the (fixed) rotations between the four chips. Furthermore, in the case of a rotation between the template image and the observed image of significantly less than one degree, one can use **avshift** to estimate the rotation.

In cases where the rotation angle is large, or where one only has a single image, a different approach can be used. Test rotations are applied to the template and the images are cross-correlated after each test rotation. The best rotation angle is then determined by finding the maximum of the cross-correlation in this three-dimensional space of shifts and rotations. This is done by fitting for the rotation angle that produces the highest cross-correlation, and then interpolating for the precise shift using the shifts determined by the cross-correlation of the two best template rotations. The task **rotfind** is used to fit the rotation angle.

Comparison of single-group images typically produces a measurement which agrees with that in the header to ~0.01 degrees. This level of accuracy is typical for the comparison of two single-group WFPC2 images, and would result in an error of about 0.1 pixels at the very edge of the chip. Some of this error undoubtedly comes from the difficulty of attempting to interpolate an undersampled image. If many dithered images of the same field are available, then higher accuracy should be obtainable (though this is probably rarely required) by comparing the image with a first pass drizzled output of all the dithered images of the field.

Cosmic Ray Removal from Singly-Dithered Data

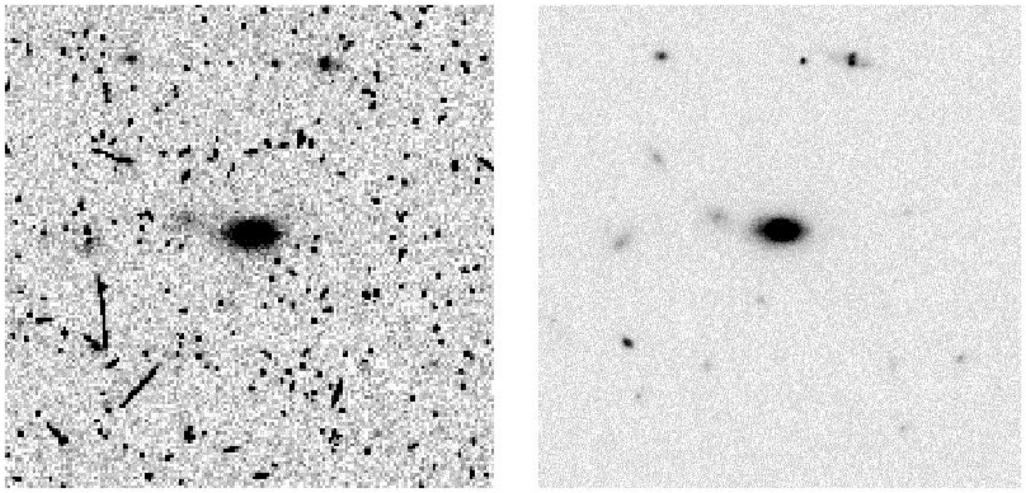
Few HST observing proposals have sufficient time to take a number of exposures at each of several dither positions. Therefore, if dithering is to be of wide-spread use, one must be able to remove cosmic rays from data where few, if any, images are taken at the same position on the sky. The dither package has therefore adapted to facilitate the removal of cosmic rays from singly-dithered data. Here we describe a general procedure, which has been found to be generally quite successful in removing cosmic rays from singly-dithered data:

1. Drizzle each image onto a separate sub-sampled output image, but setting `pixfrac=1.0` to preserve the initial pixel size (thus, for instance, an input pixel might cover four output sub-pixels, but because of the effects of shifting and geometric distortion, the sub-pixels in the output image are a weighted average of adjacent pixels, rather than just a block replication of the input).
2. Take the median of the resulting aligned output 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, applying the appropriate image shifts and geometric distortion. This is done by interpolating the values of the median image, using a program named **blot** ("blotting" is, in effect, the inverse of drizzling).
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 the value of the "blotted" estimate could have been distorted by errors in the computed image shift, or by the blurring effect of taking the median.
5. Compare the pixel values in each original image with those in the corresponding blotted image. 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 magnitudes of the latter two effects are estimated using the image derivative), 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 (typically now using a value of `pixfrac < 1`, to maximize spatial resolution).

Figure 5.1 on page 48 shows the result of applying this method to data originally taken by Cowie et al. (Cowie, Hu, and Songaila 1995), with the reduction done using IRAF scripts. In addition to demonstrating the degree to which cosmic rays can be removed from singly dithered data, this image also displays the degree to which linear reconstruction can improve the detail visible in an image. In the drizzled image, the gain in effective resolution is clearly evidenced by the "double" nucleus of the galaxy in the upper right (the two nuclei are separated by 0.2 arcseconds).

Figure 5.1: Improvement in image quality of a deep field that has been combined using drizzle.



The image on the left shows a region of one of twelve 2400s archival images taken with the F814W filter on the WF2 CCD. Numerous cosmic rays are visible. On the right is the drizzled combination of the twelve images, no two of which shared a dither position, and nearly all of which were dithered from the central position by approximately 50 pixels.

Overview of Processing Dithered Images

This section presents a more general description of the various steps involved in processing dithered images.

Remove the Sky from the Input Images (tasks: *sky*, *imhist*, *sgraph*)

If the background in the images are not identical, drizzling will create additional noise. Unequal background levels sometimes occur due to scattered Earth light, especially when observing in the continuous viewing zone. The **sky** task sets the background to zero, and stores the subtracted background value in a user-specified header keyword. Before running the **sky** task, the user should determine the approximate width of the histogram for sky counts using **imhistogram** - this value will be needed in the **sky** task.

In some cases, such as fields with extensive nebulosity, it is impossible to determine the sky value. In such cases, the user should at least make sure that the image statistics (i.e. average or median), scaled to a common exposure time and excluding the cosmic ray contribution, do not vary significantly from image to image. If it does, subtract a constant value from the images to make their overall counts similar.

In rare cases the background may be strongly non-uniform. For example, WFPC2 images may show diagonal bars or an "X" pattern with lower background. In these cases, one should mask off the anomalous region or not use the image.

Determine the Offsets between Images

The images are cross-correlated using the Fourier transform. This is performed on temporary working copies of each image where most cosmic rays have been removed and the image edges have been set to zero, in order to get a more accurate cross-correlation.

1. Perform a coarse cosmic ray removal on images to be cross-correlated.

(task: **precor**)

The **precor** task removes cosmic rays from the background areas of the image. The output of the task are cosmic ray-cleaned images having the image name with "_obj" appended to it.

The **precor** task is most useful for images of sparse deep fields, but can also be applied to most other kinds of images. It sets background areas in an image to zero while leaving astrophysical objects unchanged for cross-correlation. This is done by checking the entire image, section by section, to determine if the median in each section is significantly different from the sky. The dimension of the sections are set by the **precor.box_siz** parameter. Pixel values equal to, or greater than the **precor.min_val** parameter are counted; if the number of counted pixels is less than the **precor.min_pix** value, the software interprets that area as containing no astrophysical objects. The pixels in that region are set to zero in the output image, unless another overlapping box resets it to the original value. (Note: the sky should have been already removed from the image, either by using the **sky** task or by setting `precor.do_sky=yes`.)

2. Cross-correlate the cosmic ray-cleaned images with a reference image.

(task: **offsets** or **crossdriz**)

The task **offsets** uses the cosmic ray-cleaned images created in **preco***r* to cross-correlate a reference image with each input image. (Details about cross-correlation, and how it works can be found in the **offsets**, **crossdriz**, and **stsdas.analysis.fourier.crosscor** on-line help files in STSDAS.)

The first image in the input list is generally designated as the reference image; this is done to verify that the task ran properly (the cross-correlation of an image with itself should yield a zero shift), and to establish a numbering scheme for the output files that matches the number of images in the input list. For WFPC2 images with four groups (or CCDs) the task **offsets** performs the cross-correlation on all four groups. If only a single chip is being processed, the task **crossdriz** should be used instead. The

offsets and **crossdriz** tasks have the option to drizzle the images before cross-correlating them - this removes the geometric distortion effects.

3. Determine the shifts between the reference image and each input image.

(task: **shiftfind**)

If you examine the output from **offsets**, you'll see that the cross-correlation images have a well-defined peak near the center of the image. The center of a WFPC2 chip (dimensions 800x800) is at (401,401). The difference between the position of the cross-correlation image peak and the center of the image gives the shift between an input image and reference image. This is generally what the **shiftfind** task does for determining the image offsets.

4. Determine the average shift for each image based on **shiftfind** results

(task: **avshift**)

For a given image, **avshift** takes an average of the shifts from each chip to determine a better estimate of the overall image offset. When determining this average, the PC shift is generally excluded in the computation. This is because there is usually less image power in the PC compared to the other WF chips, yet the PC has the same amount of "noise" from cosmic rays. The output from **avshift** produces the average x- and y-shifts for each image, under the columns "best_xsh" and "best_ysh" in the **avshift** output. These are the offsets that should be used when applying the offsets in **drizzle**.

Create Mask Files for Static Pixels and Cosmic Rays

The mask file consists of two components: a static pixel mask and a cosmic ray mask. These components are multiplied together to create a final mask file used in the final **drizzle** step.

1. Create a static pixel mask file that flags permanent bad pixels and warm pixels in the images. This step is generally recommended.

(tasks: **gcombine**, **imcalc**)

The "pyramid edges" of the WFPC2 CCDs have a gradual transition from zero to full illumination due to the aberrated OTA beam (see the *WFPC2 Instrument Handbook*, Version 5, page 36 for more details). This area should be masked out in the static pixel mask file. Please refer to Table 2.5 in the *WFPC2 Instrument Handbook* for the exact locations of these vignetted areas. In WFPC2 Examples 3 and 4 in Chapter 6:, we have masked off the sections x < 50 and y < 50. Larger and more conservative areas might be masked if there are many input images or large shifts (many arcseconds).

A. For sparse, faint fields:

For sparse or faint objects with shifts larger than the astrophysical objects, combining the (unshifted) images together is an effective way to identify defects such as bad pixels and hot pixels. Astrophysical objects are smeared out in the combined image, yielding an image that predominantly shows static pixel defects. The combined image is then converted to a mask file: low level counts (from smeared-out astrophysical objects and background) are set to 1, denoting good pixel values. The higher counts - contributions from bad pixels - are set to 0.

B. Crowded fields and large, bright nebulae:

Crowded fields and very large nebulae are impossible to process with the technique described in Section A. An alternative is to use the data quality images that accompany the data (.c1h/.c1d files for WFPC2) to create a static pixel mask file. Pipeline-calibrated images retrieved from the archive use slightly out-of-date dark files. Most warm pixels in the CCDs are transient in nature, and in order to get the best warm pixel information for the images, the data should be recalibrated with a dark reference file created around the time of the observations. The resulting data quality images will be flagged with warm pixels that existed at the time of the observations. One of those data quality images can then be converted to a static pixel mask file of 1's and 0's (but first, saturated pixels and repaired warm pixels should be re-flagged as good pixels).

Another method for crowded fields and large nebulae is to convert the static pixel reference mask files, used in pipeline calibration, to 1's and 0's. The most current versions of these files can be downloaded from:

`http://www.stsci.edu/ftp/cdbs/`

This mask file does not contain warm pixel information, it only flags permanent bad pixels. The identification and correction of hot pixels may be done later in the procedure (when cosmic rays are identified).

2. Create a cosmic ray mask: shift and drizzle each input image, then combine them to make a cosmic ray-free image. Compare this clean image with each of the original input images so cosmic rays can be identified.

A. Drizzle each input image.
(task: **drizzle**).

Each input image is drizzled onto a finer grid with larger dimensions. This step corrects for geometric distortion in the WFPC2.

The drizzled "science" image is inserted into a larger "backdrop" image, like a picture surrounded by matting. Within that backdrop, each science image is shifted such that if the output drizzled images (science plus backdrop) were stacked atop each other, all the science images would be aligned. The backdrop image, which is padded with zeros, has to be large enough so that the biggest science image shifts still keeps the entire science image within the backdrop image.

In the first WFPC2 example, where each input image has dimensions 800x800, the **drizzle** parameters are set such that each input pixel is box-replicated by 2 (`drizzle.scale=0.5`); This creates a science image that is 1600x1600 pixels. The output image dimensions however are 2048x2048 (`drizzle.outnx, outny=2048`). Therefore, the output drizzled image is such that the 1600x1600 science image section is imbedded in a 2048x2048 backdrop image.

B. Create a mask file for Step C, where the images are combined, using the drizzle weight files created in Step A.

(task: **mask_head**)

The **mask_head** task converts the drizzle weight files created in Step A to a pixel list mask file with values 1 and 0. This mask file will be used when the drizzled images are combined in the next Section C; it will mask out the "matting" (backdrop) area of the drizzled image.

The **mask_head** task also inserts a keyword called "BPM" into the drizzled image header file, and sets that keyword value to the name of the pixel list mask file that will be later used by **imcombine**.

For example, if the drizzled image "img1_dr.hhh" has a weight file called "img1_drw.hhh", **mask_head** converts the weight file to a mask file called "img1_drw.pl"; the name of that pixel list mask file is written as the value for the keyword BPM (BPM=img1_drw). When **imcombine** is run, it will look for the value of the BPM keyword in each input image to see if a mask file is available for the input image.

C. Combine the images from Step A to create a median image.

(task: **imcombine**).

imcombine is used to create a median image of the shifted drizzled images that will serve as an approximation of what the images would look like without cosmic rays.

D. "Blot" (reverse-drizzle) the median image to create images that match the positions and dimensions of the original input images.

(task: **blot**).

A copy of the median image is reversed-drizzled, shifted, and trimmed to match the position and geometric distortions of each input image. If you were to "blink" the input and blotted image, both images should align perfectly.

E. Create a derivative image that estimates the effects of blurring in the median image and possible errors in the input shifts.

(task: **deriv**)

In the next section (F), the blotted images will be directly compared to the original images, so that cosmic rays can be identified. But first, we need to take into account the effect of blurring and possible errors in shifts in the blotted images. To estimate these errors, the absolute value of the difference of each pixel with its four surrounding neighbors is obtained, and the largest value associated with each pixel is saved in an output image called the derivative image.

F. Compare each input image with their counterpart blotted image to identify cosmic rays and create a cosmic ray mask.

(task: **driz_cr**)

Significant differences between an input image and its counterpart blotted image are flagged as cosmic rays in the output mask file. The task also looks at the neighbors of obvious cosmic ray hits since cosmic ray signatures usually span across several pixels. The output is a cosmic ray mask file (in the pixel list format, ".pl").

Cosmic rays are flagged following this rule:

|data_image - blotted_image| > scale*deriv_image + SNR*noise

where "scale" is a user-supplied parameter in the **driz_cr** task, noise is calculated from the gain and read noise of the image (also supplied by the user).

The user must specify a cut-off signal-to-noise (SNR) value for determining if a pixel should be masked. (Actually, two cut-off signal-to-noise ratios are needed, one for detecting bad pixels outlying a cosmic ray, and a second to mask pixels adjacent to those found in the first pass.)

After running the task, the user should blink the mask with the original image to visually ascertain that all cosmic rays were flagged. If it appears that the central pixels of some stars are unnecessarily masked, the **driz_cr.scale** parameter values should be increased. Or if not enough cosmic rays on stars are masked out, those parameter values should be decreased.

3. Multiply the static pixel mask file(s) with each cosmic ray mask file to create a master mask file for each input image. This will be used in the next step, where the final drizzled image is created.

Drizzle the Input Images into a Single Output Image

Drizzle each input image, applying the shifts, and using the mask files, into a single output image.

(task: **drizzle**)

Each input image is drizzled, using the mask files previously created. This time, the output of each **drizzle** operation is written to a single output image, "stacking" the drizzled images atop each other. The drop size (**drizzle.pixfrac**) of the drizzled image is also shrunk to recover some resolution from the collection of subsampled images. Once again, the drizzled image is box-replicated to a finer grid (**drizzle.scale**) and shifts are applied to align all the images to the reference image.

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; this suggests that the **drizzle.scale** parameter should be about 0.4 or 0.5 for WFPC2, 0.5 for NIC3, and 0.5 for STIS/CCD, NIC1, and NIC2.

For users who generally have 3 to 4 dither pointings with 0.5 pixel shift increments, the drop size or **drizzle.pixfrac** parameter should be slightly larger than the **scale** value (for example, for WFPC2, `scale=0.5` and `pixfrac=0.6`); this 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` for `scale=0.5`). Also, image statistics of the center portion of the final drizzle weight image should have an R.M.S. less than 20% to 30% of the median (use **imstat** to get these numbers). The user is advised to try different settings to see which yields the best results for his or her data.

Troubleshooting

Why are the brightest portions of my final drizzled image masked out?

- Check if the correct exposure times were used in **blot**. A common mistake is omitting the output exposure time parameter in **blot** (**blot.expout**). This is especially important if the input images have unequal exposure times. **blot** takes the median image and scales the counts to match the exposure time of the input image, so that a “level playing field” is created when the input and blotted images are compared for cosmic ray identification in **driz_cr**. If the blotted image is not correctly scaled to match the input image, **driz_cr** will flag bright features in the image as cosmic rays.
- Check to see if the static pixel mask file was properly created. If you used a recalibrated image data quality file (.c1h/.c1d for WFPC2) to create your static pixel mask, be sure that the saturated pixels and repaired warm pixels were set to represent “good” pixels.
- Check the **imcombine** parameters. If your input images have unequal exposure time, and you’re using the parameter `imcobine.reject =minmax`, make sure that the **imcombine.scale** parameter is set to “`exposure.`” Otherwise, the median image will be incorrectly scaled.

Why are there multiple occurrences of astrophysical objects in my final drizzled image?

Check the shifts. It’s easy for typos to sneak in, especially if many images are being drizzled. When you’re drizzling the images for the first time (at the stage where you’re creating the cosmic ray mask), blink all the drizzled images together to make sure they overlap. Also, after blotting the images, blink them with the original input images to make sure they’re aligned.

Why are there so many cosmic rays left in my final drizzled image?

- Check the **blot** parameters, especially the **blot.expout** parameter, to ensure that the correct exposure times have been provided for each image.
- Try different values for the **driz_cr.scale** parameter. The **driz_cr.scale** parameter values are quite sensitive to cosmic ray detection. If the values are too large, cosmic rays on stars will be ignored. If the values are too small, the peaks of stars will be masked. Try various values till you get the best results. (Note: from our experience, these sort of problems are generally caused by the **blot.expout** parameter not being properly set, so be sure to rule that out before changing the **driz_cr.scale** parameter values.)

Why does the final drizzled image have a few residual cosmic rays, hot pixels, bad columns, and other bad pixels?

Check the median image. This problem often occurs because the **imcombine.nlow** and **imcombine.nhigh** parameter values (for cases where `imcombine.reject=minmax`) are not optimally set. Bright bad pixels like cosmic rays may fall on the same pixel for two or more images. The **nhigh** parameter should be set to make sure that pixel values from those images are rejected. The same situation arises for bad pixels with low DN values; the **nlow** value should be increased. This is an iterative process that depends on the number of input images available to make the median image. The median image should always be carefully inspected for remaining bad pixels before proceeding.

Why does my final drizzled image look so noisy?

- Check the **blot** parameters, especially the **blot.expout** parameter, to ensure that the correct exposure times have been provided for each image.
- Check to see if the sky subtraction was done properly. After running the **sky** task, check your images to make sure that the background is around zero, using a task like **imhistogram**. If it isn't, re-check the *sky* parameters to make sure that background subtraction was done properly. In some cases, like images with extended nebulosity, it is impossible to run *sky*. To ensure that the image counts are similar from image to image (where all images have the same exposure time), run **imstat** on selected sections of the shifted images (so the same astrophysical objects are measured in all images) and set the **imstat.upper** cutoff to exclude cosmic rays. In general, images with the same exposure time should have equal background values (unless there is significant scattered light - this is often a problem for targets

in the continuous viewing zone). If it does not, check with your Contact Scientist or e-mail the STScI Help Desk (help@stsci.edu) for assistance in determining the reason for the unequal background.

Why can’t I run mask_head, deriv, and driz_cr?

These tasks are new additions to the **dither** suite of tasks, and they are now available in the latest STSDAS release (summer 2000). If this version of STSDAS is not yet installed on your system, then you can download the tasks from the **ditherII** package directly from:

`http://www.stsci.edu/~fruchter/dither/#DitherII`

Why do error messages appear on the screen when drizzle or other related tasks are being run?

- Check to see if you’ve run out of disk space.
- Check the image, it may have been corrupted.
- It could be an IRAF or STSDAS problem. Flush the remaining processes by typing “flprc” (do it three times for luck). If that does not work, reset the task using “unlearn.” If that does not work, log out of IRAF and back in again. If that does not work. If that does not work, contact your local IRAF guru or the STScI help desk.

Bibliography

Casertano, S., et al., 2000, "WFPC2 Observations of the Hubble Deep Field - South", *AJ (in press)*; astro-ph/0010245,

`http://xxx.lanl.gov/abs/astro-ph/0010245`

Dickinson, M., et al., 1999, "NICMOS Data Handbook, Version 4.0", (Baltimore: STScI),

`http://www.stsci.edu/cgi-bin/NICMOS/si.pl?nav=documents:handbooks&sel=id:512`

Fruchter, A. S., "Andy’s Dither Page",

`http://www.stsci.edu/~fruchter/dither/`

Fruchter, A. S., "The Hubble Deep Field - Drizzling"

`http://www.stsci.edu/ftp/science/hdf/combination/drizzle.html`

Fruchter, A. S., "The Hubble Deep Field - Image Registration and Combination",

`http://www.stsci.edu/ftp/science/hdf/combination/combination.html`

Fruchter, A. S., Hook, R. N., 1997, "A Method for the Linear Reconstruction of Undersampled Images", *PASP (submitted)*; astro-ph/9808087,

`http://xxx.lanl.gov/abs/astro-ph/9808087`

Fruchter, A. S., Hook, R. N., "Linear Reconstruction of the Hubble Deep Field",

`http://www.stsci.edu/~fruchter/dither/drizzle.html`

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, "A Package for the Reduction of Dithered Undersampled Images", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/fruchtera.ps`

Fruchter, A. S., Mutchler, M., "Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration",

`http://www.stsci.edu/~fruchter/dither/ditherII.ps`

Gonzaga, S., Biretta, J., Wiggs, M., et al., 1998, "The Drizzling Cookbook", *ISR WFPC2 98-04*,

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/drizzle_c ookbook.ps`

Gonzaga, S., Wiggs, M., "WFPC2 Drizzling Cookbook Page",

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/wfpc2_dri z_ckbk.html`

HDF-S Team, "The Hubble Deep Field South - Data Reduction / Technical Information",

`http://www.stsci.edu/ftp/science/hdfsouth/hdfs.html`

Mutchler, M., Fruchter, A. S., 1997, "Drizzling Dithered WFPC2 Images - A Demonstration", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/mutchlerm.ps`

CHAPTER 6:

Using "Drizzle" - Specific Examples

In This Chapter...

This chapter contains specific examples of using the **drizzle** software.

Introduction

This chapter illustrates specific examples of using the **drizzle** software for six different examples: four WFPC2 cases, one STIS/CCD case, and one NICMOS case. Text in `courier font` in each example is what was typed in IRAF.

Example #1: PC images of the planetary nebula IRAS 17150-3224.
Example #2:WFPC2 images of the edge-on galaxy NGC4565.
Example #3: WFPC2 images of a crowded stellar field in the LMC.
Example #4: WFPC2 images of a sparse deep field (SSA22).
Example #5: STIS/CCD images of the Hubble Deep Field - North.
Example #6: NICMOS-2 images of the Cygnus Egg Nebula.

Note: Examples 1, 3, and 4 illustrate three different ways of constructing a static pixel mask file:

- Example 1 uses the data quality file of the recalibrated images.
- Example 3 uses the static pixel mask reference file (used during pipeline calibration of the data).
- Example 4 uses the actual input images to create a static pixel mask file - this only works for faint fields with offsets larger than the astrophysical objects.

No static pixel mask file is created in example 2 so that users can evaluate for themselves if it is a necessary step for that example.

Another feature to take note of, as shown in Table 6.1, is the correlation between the number of images used to construct the final drizzled image, the instrument used, and two of the primary **drizzle** parameters (**scale** and **pixfrac**). Examples 4 and 5 have many input images, and can therefore be processed with smaller **pixfrac** values. But examples 1, 2, 3, and 6 have few input images, and therefore require larger **pixfrac** values.

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. Table 6.1 shows the amount of space and run-time needed for each example in this document. The machine in this example is a 360MHz Sparc Ultra60.

Table 6.1: Summary of disk space and CPU requirements for each example.

Example	Instrument	# of input images	Image dimensions	Disk space for input files (MB)	Disk space for input, intermediate, and final files (MB)	Total time on a 360MHz Ultra (minutes)
1	WFPC2 (PC)	5	800x800	36.9	307.6	6
2	WFPC2	3	800x800x4	29.4	795.8	14
3	WFPC2	8	800x800x4	122.5	1730.1	36
4	WFPC2	12	800x800x4	176.7	2497.9	54
5	STIS/CCD	18	1024x1024	185.4	1291.3	66
6	NIC2	4	256x256	4.4	53.7	1.5

Table 6.2: Checklist of steps for each example.

Steps	Ex. 1 WFPC2 (PC)	Ex. 2 WFPC2	Ex. 3 WFPC2	Ex. 4 WFPC2	Ex. 5 STIS/CCD	Ex. 6 NICMOS (NIC2)
1. Create working images.	yes	yes	yes	yes	yes	yes
2. Subtract sky from images.	yes	see note[a]	yes	yes	yes	yes
3. Prepare images for cross-correlation.	yes	yes	yes	yes	yes	yes
4. Find offsets between images.	yes	yes	yes	yes	yes	yes
5. Determine average image shifts (4-group WFPC2 images only).	no	yes	yes	yes	no	no
6. Create static pixel mask file.	yes	yes	yes	yes	yes	yes
7. Make cosmic ray/bad pixel mask file: **7a.** Drizzle each input image.	yes	yes	yes	yes	yes	yes
7b. Create median image.	yes	yes	yes	yes	yes	yes
7c. Blot the median image.	yes	yes	yes	yes	yes	yes
7d. Create derivative image.	yes	yes	yes	yes	yes	yes
7e. Identify cosmic rays and bad pixels.	yes	yes	yes	yes	yes	yes
8. Create master mask files.	no	yes	yes	yes	yes	yes
9. Create final drizzled image.	yes	yes	yes	yes	yes	yes

a. Sky subtraction was not possible for images in Example 2, but a test was done to determine that the background level for each image was the same.

All the initial input images for each of these examples, together with the IRAF example scripts containing the commands necessary to create the final, drizzled images can be downloaded from:

`http://www.stsci.edu/instruments/wfpc2/dither/examples.html`

The command scripts for each example are divided into two parts: the first section determines the offsets between the images, and the second part constructs the combined drizzled images. For instance, to run the first part of example 1, type:

```
cl < ex1_A.cl
```

Edit the second part of the script to enter the shifts found from ex1_A.cl, and then type:

```
cl < ex1_B.cl
```

Alternatively, you can display the text of the scripts on a terminal or editor window, and run the commands by cutting and pasting pieces of it onto your IRAF window.

Before running any of the examples in this chapter, please make sure you have completed the following steps:

- Load the following tasks in IRAF:

```
stsdas toolbox  imgtools  ttools analysis
fourier  fitting dither cl.images.imutil
```

- Set the default image type in IRAF:

```
set imtype = hhh
```

- Set the image display area to an appropriately large size for displaying drizzled images (e.g., 2048x2048):

```
set stdimage = imt2048
```

- If necessary, set the scratch directory to a disk with sufficient space (for example, your current working directory, if that is large enough):

```
set tmp = "./"
```

Example 1 (PC): Planetary Nebula IRAS 17150-3224

In this example, we process five observations (u3lr0205r-209r) of the planetary nebula IRAS17150-3224 taken with the F606W filter. (HST proposal 6565, PI: Sun Kwok.) Four of the observations, each with exposure time 120 seconds, were implemented in the phase 2 proposal with the optional parameters DITHER-TYPE=BOX. The fifth image, at 200 seconds, was implemented with a POS TARG. The main target of interest, the planetary nebula, falls in the PC. Therefore, the processing of other chips will be disregarded.

A little Organizing to keep track of the Files

Copy the calibrated images (PC only) to "working" images, then store the originals in a subdirectory for safekeeping.

```
imcopy u3lr0205r.c0h[1] img1.hhh

imcopy u3lr0206r.c0h[1] img2.hhh

imcopy u3lr0207r.c0h[1] img3.hhh

imcopy u3lr0208r.c0h[1] img4.hhh

imcopy u3lr0209r.c0h[1] img5.hhh

!mkdir orig

!mv *.c0? orig
```

Subtract the Sky Background from the Images

Before proceeding, the value for the **sky.width** parameter should be determined for each image, using the task **imhist**, and using a combination of **sgraph** and **gkimosaic** to display the results on one page:

```
unlearn imhist
imhist.z1 = -10
imhist.z2 = 4096
imhist.binwidth = INDEF
imhist.nbins = 4096
imhist.autoscale = no
imhist.top_closed = yes
imhist.hist_type = “normal”
imhist.listout = yes
imhist.logy = no
imhist img1.hhh | sgraph(wl=-10,wr=10,>G "sky_pre.gki")
imhist img2.hhh | sgraph(wl=-10,wr=10,>>G "sky_pre.gki")
imhist img3.hhh | sgraph(wl=-10,wr=10,>>G "sky_pre.gki")
imhist img4.hhh | sgraph(wl=-10,wr=10,>>G "sky_pre.gki")
imhist img5.hhh | sgraph(wl=-10,wr=10,>>G "sky_pre.gki")
gkimosaic sky_pre.gki nx=1 ny=5 fill+ inter-
```

The results of **imhist** yield an approximate sky width of 4. This value is used in the **sky.width** parameter below.

```
unlearn sky
sky.masks = “”
sky.lower = -99.
sky.upper = 4096.
sky.dq = “”
sky.subsky = yes
sky.width = 4
sky.stat = “mean”
sky.skyname = “BACKGRND”
sky.skyvalue = 0
sky.verbose = no
sky *.hhh
```

The image sky subtraction is done in-place, and the sky value obtained from this task is written into the “BACKGRND” header keyword.

Determine Offsets for all the Images

a) Prepare the Images for Cross-correlation

Before the images can be cross-correlated to determine their shifts with respect to a reference image, cosmic rays should be removed to make cross-correlation easier. This is done using the **precor** task.

```
unlearn precor

precor.box_siz = “5”

precor.min_pix = 16

precor.min_val = 6.

precor.ngroup = 1

precor.do_sky = no

precor.sig2n = yes

precor img*.hhh
```

The output of this task is cosmic ray-cleaned images having the same name of the input image but with an “_obj” appended to the name.

b) Find the Offsets between the Reference (First) Image and the Input Images

Since only one chip is being processed, use the task **crossdriz** (instead of **precor**) to cross-correlate the input images with the reference image (“img1.hhh”, in this example). Be sure to load the **fourier** package first, if it has not already been loaded.

```
fourier
```

Create an input list and output list.

```
!\ls -1 *_obj.hhh > obj.list

print(“cross1x1.hhh”, > “cross.list”)

print(“cross1x2.hhh”, >> “cross.list”)

print(“cross1x3.hhh”, >> “cross.list”)

print(“cross1x4.hhh”, >> “cross.list”)

print(“cross1x5.hhh”, >> “cross.list”)
```

Run **crossdriz**:

```
unlearn crossdriz

crossdriz.dinp = yes

crossdriz.dref = yes

crossdriz.margin = 50

crossdriz.tapersz = 50

crossdriz.mintheta = 0.
crossdriz.maxtheta = 0.

crossdriz.stptheta = 1.

crossdriz.coeffs = “header”

crossdriz.expkey = “exptime”

crossdriz.xout = INDEF

crossdriz.yout = INDEF

crossdriz.pad = no

crossdriz @obj.list img1_obj.hhh @cross.list
```

where “img1_obj.hhh” is the reference image, and “cross1x” is the rootname for the output cross-correlation images.

Next, use **shiftfind** to determine the shifts between the images and the reference image. Be sure to load the **fitting** package first, if it has not already been loaded:

```
fitting

unlearn shiftfind

shiftfind.x = INDEF

shiftfind.y = INDEF

shiftfind.boxsize = INDEF

shiftfind.fwhm = 7

shiftfind.ellip = 0.05

shiftfind.pa = 45

shiftfind.fitbox = 7

shiftfind.kellip = yes

shiftfind.kpa = yes

shiftfind @cross.list shifts.txt
```

The output is written into “shifts.txt”, a text file containing the image shifts.

Create Static Pixel Mask Files

Earlier, the images were recalibrated using the best available dark reference files, so that the best available warm pixel information would be flagged in the calibrated image data quality file (.c1h/.c1d).

In the data quality image, the following pixel values represent specific types of bad pixels:

1 Reed-Solomon decoding error
2 calibration file defect
4 static defect
8 saturation
32 other bad pixels

128 permanent charge trap
512 unrepairable hot pixels
1024 repaired hot pixels

The Reed-Solomon error is due to incomplete data transfers from the spacecraft; this occurs infrequently and for the purposes of this exercise, we will ignore it. The value 8 flags saturated pixels. However, we want to retain these in the final drizzled images, so they will be reset to represent good pixels in the mask file. The value 1024 represents warm pixels that have been restored to the actual value. These are also reset in the mask file to represent good pixels.

Note: If you are drizzling a collection of images with different exposure times, where objects are saturated in long-exposure images but not in short-exposure images, you may want to consider constructing static pixel mask files for each image where saturated objects are flagged in their respective mask files. This way, your final drizzled image will only contain contributions from the unsaturated images.

The other flagged pixels, like calibration file defects, static defects, charge traps, and unrepairable hot pixels, are the same in all the data quality files (even the warm pixels because the images were taken close in time). Therefore, we only need to use one of the data quality files to create a static pixel mask file.

Copy one of the data quality files to a temporary working file, and move all data quality files to a directory for safe-keeping.

```
imcopy u3lr0205r.c1h[1] static_temp.hhh

!mv *.c1? orig
```

Replace all pixels representing saturation (8) and repaired hot pixels (1024), as well as pixels representing saturation and hot pixels (1032), to the good pixel flag (0).

```
unlearn imreplace

imreplace static_temp.hhh 0 lower=8 upper=8

imreplace static_temp.hhh 0 lower=1024 upper=1024

imreplace static_temp.hhh 0 lower=1032 upper=1032
```

Replace all other non-zero values in the data quality file with 0 (to represent bad pixels), then replace all pixels flagged as good in the data quality file with 1. (In the .c1h files, good pixels are flagged with 0, but in the static pixel mask file used in **drizzle**, good pixels are flagged as 1.)

```
unlearn imcalc

imcalc static_temp static “if im1 .gt. 0 then 0 else 1”
```

Make Cosmic Ray Masks

a) Drizzle each Input Image

Each input image is drizzled, applying the shifts obtained earlier.

```
unlearn drizzle

drizzle.wt_scl="exptime"

drizzle.expkey="exptime"

drizzle.scale=0.5

drizzle.pixfrac=1.0

drizzle.coeffs="header"

drizzle.outnx=2048

drizzle.outny=2048

drizzle.out_un="counts"

drizzle.in_mask="static"

drizzle.rot=0.0
```

Here we use the task **loop_driz**, which reads in the shifts from the file "shifts.txt" for each image and passes them as parameters to **drizzle**. It replaces the old manual way of doing this, which would have involved running **drizzle** manually for each individual image and typing in the shifts by hand.

```
!\ls -1 img?.hhh > img.list

loop_driz input=@img.list output=@img.list mask=static
   shifts.txt shifttask="shiftfind" group=1 d="_dr" w="_drw"
```

The output drizzled images have the same root names as the input images ("img*"), except that the output images are given an additional suffix "_dr", and their associated weight images have the suffix "_drw".

b) Create a Median Image from the Drizzled Images

Using **imcombine**, combine the five drizzled and shifted images to create a median image. But first, create a mask file to be used with **imcombine**. (Note: this mask file should not be confused with the static pixel mask file created earlier.) This mask file is made from the weight images created in the previous **drizzle** steps, and will only be used with **imcombine**. The task **mask_head** converts the weight images to 1's and 0's. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file. The output is a pixel list mask file for each image, "img*_drw.pl".

```
unlearn mask_head

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no
```

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine
imcombine.plfile = ““
imcombine.sigma = ““
imcombine.combine = “median”
imcombine.reject = “minmax”
imcombine.project = no
imcombine.outtype = “real”
imcombine.offsets = “none”
imcombine.masktype = “badvalue”
imcombine.maskvalue = 0.
imcombine.nlow = 2
imcombine.nhigh = 2
imcombine.nkeep = 1
imcombine.scale = “exposure”
imcombine *dr.hhh dr_med.hhh
```

The output median image is “dr_med.hhh”. Note: the `nlow=2` and `nhigh=2` parameter values were selected to keep hot pixels (in the form of dark spots in the image) and bright residual cosmic rays from appearing in the median image.

c) Blot the Combined Image

The median image is “blotted” or reverse-drizzled back to the 800x800 dimensions and positions of each original image. The results are a blotted counterpart image for each input image. If the exposure times were equal then we would be able to use **loop_blot**, but in this particular case they are not and therefore we need to run each blot iteration separately, manually inserting the exposure times and the shifts as found in the file "shifts.txt"

```
unlearn blot
blot.scale = 0.5
blot.outnx = 800
blot.outny = 800
blot.coeffs = “header”
blot.rot = 0.
blot dr_med.hhh img1_bl.hhh expo=120 xsh=0.0002 ysh=-0.0005
blot dr_med.hhh img2_bl.hhh expo=120 xsh=11.0080 ysh=5.4022
blot dr_med.hhh img3_bl.hhh expo=120 xsh=16.4763 ysh=16.3705
blot dr_med.hhh img4_bl.hhh expo=120 xsh=5.5907 ysh=11.0087

blot dr_med.hhh img5_bl.hhh expo=200 xsh=0.0974 ysh=0.1534
```

Note: The **expout** parameter is explicitly stated in each **blot** command to provide the exposure time of the corresponding original input image. This is done because the exposure times of the original input images in this example are not equal, and therefore, **expout** should not be defined with the other common parameter values (to avoid confusion).

The output blotted images are named “img*_bl.hhh”.

d) Create Derivative Images

This step estimates effects caused by blurring in the median image and possible errors in the input shifts.

```
unlearn deriv

!\ls -1 *bl.hhh > blot.list

deriv @blot.list
```

The output is a derivative image associated with each blotted image, called “img*_bl_deriv.hhh”.

e) Compare each Input Image with its Counterpart Blotted Image to Identify Cosmic Rays

```
unlearn driz_cr

driz_cr.gain = 7.

driz_cr.rn = 5.

driz_cr.SNR = “4.0 3.0”

driz_cr.scale = “0.5 0.4”

driz_cr.backg = “backgrnd”

driz_cr img?.hhh 1
```

The output for each image is a cosmic ray pixel list mask file called “img*cr.pl”.

Create a "Master" Mask for each Image

The static mask file previously created is multiplied with each individual cosmic ray mask file to create a series of master mask files that will be used in the final **drizzle** operation.

```
unlearn imarith

imarith static.hhh * img1_cr.pl img1_cr.pl

imarith static.hhh * img2_cr.pl img2_cr.pl

imarith static.hhh * img3_cr.pl img3_cr.pl

imarith static.hhh * img4_cr.pl img4_cr.pl

imarith static.hhh * img5_cr.pl img5_cr.pl
```

Construct the Final Drizzled Image

Each input image is drizzled, applying the shifts and the mask files previously created.

```
unlearn drizzle

drizzle.wt_scl = “exptime”

drizzle.expkey = “exptime”

drizzle.scale = 0.5

drizzle.pixfrac = 0.8

drizzle.coeffs = “header”

drizzle.outnx = 2048

drizzle.outny = 2048

drizzle.out_un = “counts”

drizzle.fillval = 0

drizzle.rot = 0.

!\ls -1 *_cr.pl > masks.list

loop_driz input=@img.list output=final.hhh mask=@masks.list
   shifts.txt shifttask="shiftfind" group=1 d="" w="_w"
```

The final drizzled image is “final.hhh”, along with its associated drizzle weight image, “final_w.hhh”. Be sure to tell the image display software to display all 2048x2048 pixels of the drizzled image.

```
set stdimage = imt2048

display final.hhh
```

Example 2 (WFPC2): Edge-on Galaxy NGC 4565

This example contains images of an edge-on spiral galaxy NGC 4565: the nucleus falls primarily on WF2 while other chips have diffuse nebulosity from the galaxy edges. The data was obtained from HST proposal 6092 (PI: Keith Ashman). The datasets used were u31s0101t-103t (corresponding to exposures 1-3 in visit 1).

A little Organizing to keep track of Files

Copy the original images to "working" images, then store the originals in a subdirectory for safekeeping.

```
!cp u31s0101t.c0h img1.hhh

!cp u31s0101t.c0d img1.hhd

!cp u31s0102t.c0h img2.hhh

!cp u31s0102t.c0d img2.hhd
```

```
!cp u31s0103t.c0h img3.hhh

!cp u31s0103t.c0d img3.hhd

!mkdir orig

!mv *.c?? orig
```

Subtract the Sky Background from the Images

The use of the **imhist** and **sky** tasks to perform the sky subtraction here is omitted since the galaxy is too extended. However, the sky parameters are still set since they are used in the next step by **precor**.

```
unlearn sky

sky.masks = ""

sky.lower = -99.

sky.upper = 4096.

sky.dq = ""

sky.subsky = yes

sky.width = 8

sky.stat = "mean"

sky.skyname = "BACKGRND"

sky.skyvalue = 0

sky.verbose = no
```

Determine Offsets for all the Images

a) Prepare the Images for Cross-correlation

Before the images can be cross-correlated to determine their shifts with respect to a reference image, cosmic rays should be removed to make cross-correlation easier. This is done using the **precor** task.

```
unlearn precor

precor.box_siz = "5"

precor.min_pix = 16

precor.min_val = 3.

precor.ngroup = 4

precor.do_sky = yes

precor.sig2n = no

precor img*.hhh
```

The output of this task is cosmic ray-cleaned images having the same name of the input image but with an "_obj" appended to the name.

b) Find the Offsets between the Reference (First) Image and the Input Images

Run **offsets** to cross-correlate the input images with the reference image (in this example, the first image, "img1.hhh", is picked as the reference image.) Be sure to load the **fourier** package first, if it has not already been loaded.

```
fourier
```

Set up the parameters for **cdriz**, a pset (sub-task) called by **offsets**:

```
cdriz.margin = 50

cdriz.tapersz = 50

cdriz.pad = no
```

Create a list of "*obj.hhh" in numerical order.

```
!\ls -1 *_obj.hhh > obj.list
```

Run the **offsets** task.

```
unlearn offsets

offsets @obj.list img1_obj.hhh cross1x coeffs="header"
```

where "img1_obj.hhh" is the reference image, and "cross1x" is the rootname for the output cross-correlation images.

Next, use **shiftfind** to determine the shifts between the images and the reference image. Be sure to load the **fitting** package first, if it has not already been loaded.

```
fitting

unlearn shiftfind

shiftfind.x = INDEF

shiftfind.y = INDEF

shiftfind.boxsize = INDEF

shiftfind.fwhm = 7

shiftfind.ellip = 0.05

shiftfind.pa = 45

shiftfind.fitbox = 7

shiftfind.kellip = yes

shiftfind.kpa = yes

shiftfind cross1x*.hhh shifts.txt
```

The output is written into "shifts.txt", a text file containing the shifts for each group in each image.

c) Determine the Average Shift for the Images

For a given WFPC2 observation, **avshift** averages the shifts in each group to produce an overall best estimate of the shift. The results for each image are stored in the output file under the columns "best_xsh" and "best_ysh". However, the only chip that contains enough data to give a good cross-correlation is WF2. Therefore, the **avshift.weight** parameter is set to “0 1 0 0”.

```
unlearn avshift
```

```
avshift shifts.txt angle=0 weight=”0. 1. 0. 0.” >
   average_shifts
```

Before proceeding, a little housekeeping; delete or save all “*.obj.hhh” and “cross*” files.

```
!mkdir tempfiles
```

```
!mv *obj*.hh? cross1x* tempfiles
```

Determine if the Count Level is the same for all Images

In the other examples, the **sky** task is used to determine a background level that is subtracted from each image prior to running **drizzle**. This is done to set the background to a constant (zero) in all images because unequal backgrounds would introduce additional noise to the final drizzled image.

In the present example, determining the background is almost impossible because of the high levels of emission in each chip. Therefore, we run some image statistics to make sure that the counts do not significantly deviate from image to image.

Since most of the interesting astrophysical features appear in WF2, we perform statistics on those images. The second and third WF2 images are shifted to align with the first image, so that statistics could be easily obtained in the same region for all three images.

```
unlearn imshift

imcopy img1.hhh[2] tempfiles/img1_g2.hhh

imcopy img2.hhh[2] tempfiles/img2_g2.hhh

imcopy img3.hhh[2] tempfiles/img3_g2.hhh

cd tempfiles

imcopy  img1_g2.hhh img1_g2s.hhh

imshift img2_g2.hhh img2_g2s.hhh 4.974 5.022

imshift img3_g2.hhh img3_g2s.hhh -5.024 -5.015
```

Statistics are obtained for two image subsections. Since these are nebulous areas with no stars, counts are generally below 30 DN - therefore, we pselect an upper level cut-off of 50 DN in **imstat** to exclude cosmic rays.

```
unlearn imstat

imstat *s.hhh[369:468,730:779] field=”image,npix,mean” up=50

Results:

#                IMAGE           NPIX   MEAN

img1_g2s.hhh[369:468,730:779]  4997   7.612

img2_g2s.hhh[369:468,730:779]  4997   7.64

img3_g2s.hhh[369:468,730:779]  4979   7.867

imstat *s.hhh[349:450,190:239] field=”image,npix,mean” up=50

Results:

#                IMAGE           NPIX   MEAN

img1_g2s.hhh[349:450,190:239]  5085   27.37

img2_g2s.hhh[349:450,190:239]  5084   27.41

img3_g2s.hhh[349:450,190:239]  5093   27.33
```

In this case, the counts are generally identical in all three images, and it is therefore not necessary to scale them. Remember to move up to the higher-level directory before continuing:

```
cd ..
```

Create Static Pixel Mask Files

For this example, this step is omitted. You may want to try creating a static pixel mask file based on the other examples, and compare the results with and without the static pixel mask.

Make Cosmic Ray Masks

From this point onwards, each chip will be processed separately. Therefore, create a subdirectory for each CCD chip.

```
!mkdir pc wf2 wf3 wf4
```

a) Drizzle each Input Image.

Each input image is drizzled, applying the shifts obtained earlier. Note that we set `masks=""` since no static mask is used here.

```
unlearn drizzle

drizzle.wt_scl=”exptime”

drizzle.expkey=”exptime”

drizzle.scale=0.5

drizzle.pixfrac=1.0

drizzle.coeffs=”header”

drizzle.outnx=2048

drizzle.outny=2048
drizzle.out_un="counts"

drizzle.in_mask=""

drizzle.rot=0.0

cd pc

!\ls -1 ../img?.hhh > ../img.list

loop_driz input=@../img.list output=@../img.list mask=""
   ../average_shifts group=1 d="_dr" w="_drw"

cd ../wf2

loop_driz input=@../img.list output=@../img.list mask=""
   ../average_shifts group=2 d="_dr" w="_drw"

cd ../wf3

loop_driz input=@../img.list output=@../img.list mask=""
   ../average_shifts group=3 d="_dr" w="_drw"

cd ../wf4

loop_driz input=@../img.list output=@../img.list mask=""
   ../average_shifts group=4 d="_dr" w="_drw"

cd ..
```

The output drizzled images have the same root names as the input images ("img*"), except that the output images are given an additional suffix "_dr", and their associated weight images have the suffix "_drw".

b) Create a Median Image from the Drizzled Images

Create a mask file to be used with **imcombine**. **mask_head** converts the weight images created in the previous section to 1's and 0's. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file.

```
unlearn mask_head

cd pc

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no

cd ../wf2

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no

cd ../wf3

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no
```

```
cd ../wf4
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..
```

The output is a pixel list mask file for each image, “img*_drw.pl”.

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine
imcombine.plfile = ““
imcombine.sigma = ““
imcombine.combine = “median”
imcombine.reject = “minmax”
imcombine.project = no
imcombine.outtype = “real”
imcombine.offsets = “none”
imcombine.masktype = “badvalue”
imcombine.maskvalue = 0.
imcombine.scale = “exposure”
imcombine.nkeep = 1
imcombine.nlow = 1
imcombine.nhigh = 1
cd pc
imcombine *dr.hhh dr_med.hhh
cd ../wf2
imcombine *dr.hhh dr_med.hhh
cd ../wf3
imcombine *dr.hhh dr_med.hhh
cd ../wf4
imcombine *dr.hhh dr_med.hhh
cd ..
```

The output for each group is a median-combined image “dr_med.hhh”.

c) Blot the Combined Image for each Group

The median image for each group is “blotted” or reverse-drizzled back to the 800x800 dimensions of each original image, and also shifted. The results are a blotted counterpart image for each input image. Note that the images all have the same exposure time, 160s, so we use the **loop_blot** task here to simplifiy the process.

```
unlearn blot

blot.scale = 0.5

blot.outnx = 800

blot.outny = 800

blot.coeffs = “header”

blot.rot = 0.

blot.expout = 160

cd pc

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=1 b="_bl"

cd ../wf2

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=2 b="_bl"

cd ../wf3

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=3 b="_bl"

cd ../wf4

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=4 b="_bl"

cd ..
```

The output blotted images are named “img*_bl.hhh”.

d) Create Derivative Images

This step estimates effects caused by blurring in the median image and possible errors in the input shifts.

```
unlearn deriv

cd pc

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf2

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf3
```

```
!\ls -1 *bl.hhh > blot.list
deriv @blot.list
cd ../wf4
!\ls -1 *bl.hhh > blot.list
deriv @blot.list
cd ..
```

The output is a derivative image associated with each blotted image, called “img*_bl_deriv.hhh”.

e) Compare each Input Image with its Counterpart Blotted Image to Identify Cosmic Rays

```
unlearn driz_cr
driz_cr.gain = 7.
driz_cr.rn = 5.
driz_cr.SNR = “4.0 2.5”
driz_cr.scale = “0.9 0.5”
driz_cr.backg = “backgrnd”
cd pc
driz_cr ../img*.hhh 1
cd ../wf2
driz_cr ../img*.hhh 2
cd ../wf3
driz_cr ../img*.hhh 3
cd ../wf4
driz_cr ../img*.hhh 4
cd ..
```

The output from this file is a cosmic ray pixel list mask file called “img*_cr.pl”.

Create a “Master” Mask File for each Image

This step is omitted because static pixel mask files were not created in this example.

Construct the Final Drizzled Images

Each input image is drizzled, applying the shifts and the pixel mask files previously created.

```
unlearn drizzle
drizzle.wt_scl = “exptime”
drizzle.expkey = “exptime”
drizzle.scale = 0.5

drizzle.pixfrac = 0.9

drizzle.coeffs = “header”

drizzle.outnx = 2048

drizzle.outny = 2048

drizzle.out_un = “counts”

drizzle.fillval = 0

drizzle.rot = 0.

cd pc

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=1 d=""  w="_w"

cd ../wf2

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=2 d=""  w="_w"

cd ../wf3

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=3 d=""  w="_w"

cd ../wf4

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=4 d=""  w="_w"

cd ..
```

The output drizzled image for each group is “final_g*.hhh”, with an associated weight file, “final_g*w.hhh”. Be sure to tell the image display software to display all 2048x2048 pixels of the drizzled image.

```
set stdimage = imt2048

display pc/final.hhh

display wf2/final.hhh

display wf3/final.hhh

display wf4/final.hhh
```

Example 3 (WFPC2) LMC Crowded Stellar Field

This example uses eight images of the Large Magellanic Cloud Bar, taken in the F547M filter, from HST proposal 6102 (PI: Bengt Gustafsson). The datasets used are u2z30201t-208t (corresponding to exposure line 110 in visit 2). The observations were implemented as a spatial scan: a two line scan with two dwell points per line. This resulted in a parallelogram-shaped dither with two exposures at each "corner." In such a case, it is possible to combine the two observations at the same pointing, using a task like **crrej**, to remove cosmic rays. But tests by Andy Fruchter have indicated that reducing such data as single point dithers usually yields better results.

A little Organizing to keep track of Files

Copy the original images to "working" images, then store the originals in a subdirectory for safe-keeping.

```
!cp u2z30201t.c0h img1.hhh
!cp u2z30201t.c0d img1.hhd
!cp u2z30202t.c0h img2.hhh
!cp u2z30202t.c0d img2.hhd
!cp u2z30203t.c0h img3.hhh
!cp u2z30203t.c0d img3.hhd
!cp u2z30204t.c0h img4.hhh
!cp u2z30204t.c0d img4.hhd
!cp u2z30205t.c0h img5.hhh
!cp u2z30205t.c0d img5.hhd
!cp u2z30206t.c0h img6.hhh
!cp u2z30206t.c0d img6.hhd
!cp u2z30207t.c0h img7.hhh
!cp u2z30207t.c0d img7.hhd
!cp u2z30208t.c0h img8.hhh
!cp u2z30208t.c0d img8.hhd
!mkdir orig
!mv *.c?? orig
```

Subtract the Sky Background from the Images

Before proceeding, the value for the **sky.width** parameter should be determined for each image, using the task **imhistogram**, and using a combination of sgraph and gkimosaic to display the results on one page:

```
unlearn imhist
imhist.z1=-50
imhist.z2=4096
imhist.nbins=4096
imhist.autoscale=no
imhist.top_closed=yes
imhist.listout=yes
imhist.logy=no
for (i=1; i <=8; i+=1) {
  infile="img"//i//".hhh"
  imhist(infile) | sgraph(wl=-5,wr=10,>>G "sky_pre.gki")
}
gkimosaic sky_pre.gki nx=2 ny=4 fill+ inter-
```

The graphical display indicates a sky full width of 6; this value is used in the **sky.width** parameter below. (The full width is about 7 for the WF chips but the difference ultimately will not make much difference in the background calculations done in **sky**.)

```
unlearn sky
sky.masks = ““
sky.lower = -99.
sky.upper = 4096.
sky.dq = ““
sky.subsky = yes
sky.width = 6
sky.stat = “mean”
sky.skyname = “BACKGRND”
sky.skyvalue = 0
sky.verbose = no
sky *.hhh
```

The image sky subtraction is done in-place, and the sky value obtained by this task is written in the “BACKGRND” header keyword.

Determine Offsets for all the Images

a) Prepare the Images for Cross-Correlation

Before the images can be cross-correlated to determine their shifts with respect to a reference image, cosmic rays should be removed to make cross-correlation easier. This is done using the **precor** task.

```
unlearn precor
precor.box_siz = "5"
precor.min_pix = 16
precor.min_val = 6.
precor.ngroup = 4
precor.do_sky = no
precor.sig2n = yes
precor img*.hhh
```

The output of this task is cosmic ray-cleaned images having the same name as the input image but with an "_obj" appended to the name.

b) Find the Offsets between the Reference (First) Image and the Input Images

Run **offsets** to cross-correlate the input images with the reference image (in this example, the first image, "img1.hhh", is picked as the reference image.) Be sure to load the **fourier** package first, if it has not already been loaded.

```
fourier
```

Set up the parameters for **cdriz**, a pset (sub-task) called by **offsets**:

```
unlearn cdriz
cdriz.margin = 50
cdriz.tapersz = 50
cdriz.pad = no
```

Create a list of "*obj.hhh" files.

```
!\ls -1 *_obj.hhh > obj.list
```

Run the **offsets** task.

```
unlearn offsets
offsets @obj.list img1_obj.hhh cross1x coeffs="header"
```

where "img1_obj.hhh" is the reference image, and "cross1x" is the rootname for the output cross-correlation images.

Next, use **shiftfind** to determine the shifts between the images and the reference image. Be sure to load the **fitting** package first, if it has not already been loaded.

```
fitting
unlearn shiftfind
shiftfind.x = INDEF
shiftfind.y = INDEF
shiftfind.boxsize = INDEF
shiftfind.fwhm = 7

shiftfind.ellip = 0.05

shiftfind.pa = 45

shiftfind.fitbox = 7

shiftfind.kellip = yes

shiftfind.kpa = yes

shiftfind cross1x*.hhh shifts.txt
```

The output is written into “shifts.txt”, a text file containing the shifts for each group in each image.

c) Determine the Average Shift for the Images

For a given WFPC2 observation, **avshift** averages the shifts in each group to produce an overall best estimate of the shift. The results for each image are stored in the output file under the columns "best_xsh" and "best_ysh". Since the PC typically has fewer objects than the WF chips, its cross-correlation is not as good and it is therefore given zero weight.

```
unlearn avshift

avshift shifts.txt angle=0 weight=”0. 1. 1. 1.” >
   average_shifts
```

Before proceeding, a little housekeeping; delete or save all *.obj.hhh and cross* files.

```
!mkdir tempfiles

!mv *obj*.hh? cross1x* tempfiles
```

Create Static Pixel Mask Files

Here, we illustrate how to use the static pixel mask reference file, f8213081u.r0h, .r0d), used in pipeline calibration, to create a simple static pixel mask. This file can be downloaded from

http://www.stsci.edu/ftp/cdbs/

In the static pixel mask reference file, all good pixels are set to 0, and bad pixels are set to 2 and 256. Convert the image to a mask file where good pixels are set to 1, and bad pixels to 0.

```
unlearn imcalc

imcalc f8213081u.r0h static “if im1 .gt. 0 then 0 else 1”

!mv f82*.* orig
```

From this point on, we will be working on the images group by group, therefore create a separate subdirectory for each CCD chip.

```
!mkdir pc wf2 wf3 wf4

imcopy static.hhh[1] pc/static.hhh

imcopy static.hhh[2] wf2/static.hhh
imcopy static.hhh[3] wf3/static.hhh

imcopy static.hhh[4] wf4/static.hhh
```

Areas near the edges of the mask could also have bad pixels. To avoid unwanted anomalies, set the pyramid edges to zero.

```
unlearn imreplace

cd pc

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf2

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf3

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf4

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ..
```

Make Cosmic Ray Masks

a) Drizzle each Input Image

Each input image is drizzled, applying the shifts obtained earlier.

```
unlearn drizzle

drizzle.wt_scl="exptime"

drizzle.expkey="exptime"

drizzle.scale=0.5

drizzle.pixfrac=1.0

drizzle.coeffs="header"
drizzle.outnx=2048

drizzle.outny=2048

drizzle.out_un="counts"

drizzle.in_mask="static"

drizzle.rot=0.0

cd pc

!\ls -1 ../img?.hhh > ../img.list

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=1 d="_dr" w="_drw"

cd ../wf2

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=2 d="_dr" w="_drw"

cd ../wf3

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=3 d="_dr" w="_drw"

cd ../wf4

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=4 d="_dr" w="_drw"

cd ..
```

The output drizzled images have the same root names as the input images ("img*"), except that the output images are given an additional suffix "_dr", and their associated weight images have the suffix "_drw".

b) Create a Median Image from the Drizzled Images

Create a mask file to be used with **imcombine**. (Note: this mask file should not be confused with the static pixel mask file created earlier. This mask file is made from the weight images created in the previous **drizzle** steps, and will only be used with **imcombine**.) **mask_head** converts the weight image to 1's and 0's. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file.

```
unlearn mask_head

cd pc

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no

cd ../wf2

!\ls -1 *dr.hhh > dr.list

!\ls -1 *drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no
```

```
cd ../wf3
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ../wf4
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..
```

The output is a pixel list mask file for each image, “img*_drw.pl”.

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine
imcombine.plfile = ““
imcombine.sigma = ““
imcombine.combine = “median”
imcombine.reject = “minmax”
imcombine.project = no
imcombine.outtype = “real”
imcombine.offsets = “none”
imcombine.masktype = “badvalue”
imcombine.maskvalue = 0.
imcombine.nlow = 2
imcombine.nhigh = 1
imcombine.nkeep = 1
imcombine.scale = “exposure”
cd pc
imcombine *dr.hhh dr_med.hhh
cd ../wf2
imcombine *dr.hhh dr_med.hhh
cd ../wf3
imcombine *dr.hhh dr_med.hhh
cd ../wf4
imcombine *dr.hhh dr_med.hhh
```

```
cd ..
```

The output median image for each group is “dr_med.hhh”.

c) Blot the Combined Image for each Group

The median image for each group is “blotted” or reverse-drizzled back to the 800x800 dimensions and positions of each original image. The results are a blotted counterpart image for each input image. Note that the images all have the same exposure time, 200s, so we use the **loop_blot** task here to simplify the process.

```
unlearn blot

blot.scale = 0.5

blot.outnx = 800

blot.outny = 800

blot.expout = “200”

blot.coeffs = “header”

blot.rot = 0.

cd pc

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=1 b="_bl"

cd ../wf2

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=2 b="_bl"

cd ../wf3

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=3 b="_bl"

cd ../wf4

loop_blot input=dr_med.hhh output=@../img.list
   ../average_shifts group=4 b="_bl"

cd ..
```

The output blotted images are named “img*_bl.hhh”.

d) Create Derivative Images

This step estimates effects caused by blurring in the median image and possible errors in the input shifts.

```
unlearn deriv

cd pc

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf2
```

```
!\ls -1 *bl.hhh > blot.list
deriv @blot.list
cd ../wf3
!\ls -1 *bl.hhh > blot.list
deriv @blot.list
cd ../wf4
!\ls -1 *bl.hhh > blot.list
deriv @blot.list
cd ..
```

The output is a derivative image associated with each blotted image called “img*_bl_deriv.hhh”.

e) Compare each Input Image with its Counterpart Blotted Image to identify Cosmic Rays

```
unlearn driz_cr
driz_cr.gain = 7.
driz_cr.rn = 5.
driz_cr.SNR = “4.0 3.0”
driz_cr.scale = “0.5 0.4”
driz_cr.backg = “backgrnd”
cd pc
driz_cr ../img*.hhh 1
cd ../wf2
driz_cr ../img*.hhh 2
cd ../wf3
driz_cr ../img*.hhh 3
cd ../wf4
driz_cr ../img*.hhh 4
cd ..
```

The output for each image is a cosmic ray pixel list mask file called “img*_cr.pl”.

Create a "Master" Mask File for each Image

For each chip, the static mask file previously created is multiplied with each individual cosmic ray mask file to create a series of master mask files that will be used in the final **drizzle** operation.

```
unlearn imarith
cd  pc
```

```
imarith static.hhh * img1_cr.pl img1_cr.pl
imarith static.hhh * img2_cr.pl img2_cr.pl
imarith static.hhh * img3_cr.pl img3_cr.pl
imarith static.hhh * img4_cr.pl img4_cr.pl
imarith static.hhh * img5_cr.pl img5_cr.pl
imarith static.hhh * img6_cr.pl img6_cr.pl
imarith static.hhh * img7_cr.pl img7_cr.pl
imarith static.hhh * img8_cr.pl img8_cr.pl
cd ../wf2
imarith static.hhh * img1_cr.pl img1_cr.pl
imarith static.hhh * img2_cr.pl img2_cr.pl
imarith static.hhh * img3_cr.pl img3_cr.pl
imarith static.hhh * img4_cr.pl img4_cr.pl
imarith static.hhh * img5_cr.pl img5_cr.pl
imarith static.hhh * img6_cr.pl img6_cr.pl
imarith static.hhh * img7_cr.pl img7_cr.pl
imarith static.hhh * img8_cr.pl img8_cr.pl
cd ../wf3
imarith static.hhh * img1_cr.pl img1_cr.pl
imarith static.hhh * img2_cr.pl img2_cr.pl
imarith static.hhh * img3_cr.pl img3_cr.pl
imarith static.hhh * img4_cr.pl img4_cr.pl
imarith static.hhh * img5_cr.pl img5_cr.pl
imarith static.hhh * img6_cr.pl img6_cr.pl
imarith static.hhh * img7_cr.pl img7_cr.pl
imarith static.hhh * img8_cr.pl img8_cr.pl
cd ../wf4
imarith static.hhh * img1_cr.pl img1_cr.pl
imarith static.hhh * img2_cr.pl img2_cr.pl
imarith static.hhh * img3_cr.pl img3_cr.pl
imarith static.hhh * img4_cr.pl img4_cr.pl
imarith static.hhh * img5_cr.pl img5_cr.pl
imarith static.hhh * img6_cr.pl img6_cr.pl
imarith static.hhh * img7_cr.pl img7_cr.pl
imarith static.hhh * img8_cr.pl img8_cr.pl
```

```
cd ..
```

Construct the Final Drizzled Images

Each input image is drizzled, applying the shifts and the mask files previously created.

```
unlearn drizzle

drizzle.wt_scl = “exptime”

drizzle.expkey = “exptime”

drizzle.scale = 0.5

drizzle.pixfrac = 0.8

drizzle.coeffs = “header”

drizzle.outnx = 2048

drizzle.outny = 2048

drizzle.out_un = “counts”

drizzle.fillval = 0

drizzle.rot = 0.

cd pc

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=1 d=""  w="_w"

cd ../wf2

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=2 d=""  w="_w"

cd ../wf3

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=3 d=""  w="_w"

cd ../wf4

!\ls -1 img*_cr.pl > mask.list

loop_driz input=@../img.list output=final mask=@mask.list
   ../average_shifts group=4 d=""  w="_w"

cd ..
```

The output for each group is a drizzled image called “final_g*.hhh”, with an associated weight file called “final_g*w.hhh”. Be sure to tell the image display software to display all 2048x2048 pixels of the drizzled image.

```
set stdimage = imt2048
```

```
display pc/final.hhh
display wf2/final.hhh
display wf3/final.hhh
display wf4/final.hhh
```

Example 4 (WFPC2): Sparse Deep Field SSA22

The images in this example are twelve WFPC2 images taken with the F814W filter. These images are the Hawaii Deep Survey field SSA22, taken from HST program 5399 (PI: Lennox Cowie). The observations were implemented as POS TARGs in the phase 2 program. Datasets used in this example are u2h90101t-106t (corresponding to exposures 1-6 in visit 1) and u2h90201t-206t (visits 71-76 in visit 1).

Note: this is the same set of images used in “Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration” by Fruchter and Mutchler (available at *http://www.stsci.edu/~fruchter/dither/#DitherII*). The example in this document is similar to that paper, the only difference being that the example in that paper covered just WF2, whereas this document covers all four chips.

A little Organizing to keep track of Files

Copy the original images to working image names, and store them in a directory for safekeeping.

```
!cp u2h90101t.c0h img01.hhh
!cp u2h90101t.c0d img01.hhd
!cp u2h90102t.c0h img02.hhh
!cp u2h90102t.c0d img02.hhd
!cp u2h90103t.c0h img03.hhh
!cp u2h90103t.c0d img03.hhd
!cp u2h90104t.c0h img04.hhh
!cp u2h90104t.c0d img04.hhd
!cp u2h90105t.c0h img05.hhh
!cp u2h90105t.c0d img05.hhd
!cp u2h90106t.c0h img06.hhh
!cp u2h90106t.c0d img06.hhd
!cp u2h90201t.c0h img07.hhh
!cp u2h90201t.c0d img07.hhd
!cp u2h90202t.c0h img08.hhh
```

```
!cp u2h90202t.c0d img08.hhd
!cp u2h90203t.c0h img09.hhh
!cp u2h90203t.c0d img09.hhd
!cp u2h90204t.c0h img10.hhh
!cp u2h90204t.c0d img10.hhd
!cp u2h90205t.c0h img11.hhh
!cp u2h90205t.c0d img11.hhd
!cp u2h90206t.c0h img12.hhh
!cp u2h90206t.c0d img12.hhd
!mkdir orig
!mv *.c?? orig
```

Subtract the Sky from the Images.

First, the value for the **sky.width** parameter should be determined using the task **imhistogram**.

Example: display the image histogram for img01.hhh[1].

```
unlearn imhistogram
imhist.z1=-50
imhist.z2=4096
imhist.nbins=4096
imhist.autoscale=no
imhist.top_closed=yes
imhist.listout=yes
imhist.logy=no
imhist img01.hhh | sgraph wl=0 wr=20
```

The **imhist** graphical display indicates a sky full width of 8; this value is used in the **sky.width** parameter below. (The full width is about 14 for the WF chips but the difference ultimately will not make much difference in the background calculations done in **sky**.)

```
unlearn sky
sky.masks = ""
sky.lower = -99.
sky.upper = 4096.
sky.dq = ""
sky.subsky = yes
sky.width = 8
sky.stat = "mean"
sky.skyname = "BACKGRND"

sky.skyvalue = 0

sky.verbose = no

sky *.hhh
```

The image sky subtraction is done in-place, and the sky value obtained from this task is written in the "BACKGRND" header keyword.

Determine the Shifts of the all the Images

a) Prepare the Images for Cross-correlation

Before the images can be cross-correlated to determine their shifts with respect to a reference image, cosmic rays should be removed to make cross-correlation easier. This is done using the **precor** task.

```
unlearn precor

precor.box_siz = “5”

precor.min_pix = 16

precor.min_val = 6.

precor.ngroup = 4

precor.do_sky = no

precor.sig2n = yes

precor img*.hhh
```

The output of this task is cosmic ray-cleaned images having the same name of the input image but with an "_obj" appended to the name.

b) Find the Offsets between the Reference (First) Image and the Input Images

Run **offsets** to cross-correlate the input images with the reference image. (In this example, the first image, "img1.hhh", is picked as the reference image.) Be sure to load the **fourier** package first, if it has not already been loaded, and set the parameters for the **cdriz** pset (sub-task) which is called by offsets.

```
fourier

cdriz.margin = 50

cdriz.tapersz = 50

cdriz.pad = no
```

Create a list of "*obj.hhh" files.

```
!\ls -1 *obj.hhh > obj_list
```

Run the **offsets** task.

```
unlearn offsets

offsets @obj_list img01_obj.hhh cross1x coeffs=“header”
```

where `"img01_obj.hhh"` is the reference image, and `"cross1x"` is the rootname for the output cross-correlation images.

Next, use **shiftfind** to determine the shifts between the images and the reference image. Be sure to load the **fitting** package first, if it has not already been loaded.

```
fitting

unlearn shiftfind

shiftfind.x = INDEF

shiftfind.y = INDEF

shiftfind.boxsize = INDEF

shiftfind.fwhm = 7

shiftfind.ellip = 0.05

shiftfind.pa = 45

shiftfind.fitbox = 7

shiftfind.kellip = yes

shiftfind cross1x*.hhh shifts.txt
```

The output is written in “shifts.txt”, a text file containing the shifts for each group in each image.

c) Determine the Average Shift for the Images

For a given WFPC2 observation, **avshift** averages the shifts in each group to produce an overall best estimate of the shift. The results for each image are stored in the output file under the columns “best_xsh” and “best_ysh”.

```
unlearn avshift

avshift shifts.txt angle=0 weight="0. 1. 1. 1." >
   average_shifts
```

Before proceeding, a little housekeeping; delete or save all “*.obj.hhh” and “cross*” files.

```
!mkdir tempfiles

!mv *obj*.hh? cross1x* tempfiles
```

Create Static Pixel Mask Files

Combine all the unshifted input images to create an image where the astrophysical objects are smeared out, and static bad pixels are enhanced.

```
unlearn gcombine

gcombine img*.hhh med_temp.hhh groups="*" combine=median
```

The resulting combined image has faint low level features from the smeared-out astrophysical objects. Set those counts and the background (in this example, all counts less than 5 DN) to 1, flagging them as good pixels.

Counts equal to and greater than 5 DN represent static bad pixels, and these are set to 0.

From this point on, we will be working on the images group by group.

```
!mkdir pc wf2 wf3 wf4

unlearn imcalc

imcalc med_temp[1] pc/static.hhh  "if (abs(im1) .gt. 5) then
   0 else 1"

imcalc med_temp[2] wf2/static.hhh "if (abs(im1) .gt. 5) then
   0 else 1"

imcalc med_temp[3] wf3/static.hhh "if (abs(im1) .gt. 5) then
   0 else 1"

imcalc med_temp[4] wf4/static.hhh "if (abs(im1) .gt. 5) then
   0 else 1"
```

Areas near the edges of the mask could also have bad pixels. To avoid unwanted anomalies, set the edges to zero.

```
unlearn imreplace

cd pc

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf2

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf3

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ../wf4

imrep static.hhh[1:50,1:800] 0

imrep static.hhh[1:800,1:50] 0

imrep static.hhh[799:800,1:800] 0

imrep static.hhh[1:800,799:800] 0

cd ..
```

Make Cosmic Ray Masks

a) Drizzle each Input Image

Each input image is drizzled, applying the shifts obtained earlier.

```
unlearn drizzle

drizzle.wt_scl=”exptime”

drizzle.expkey=”exptime”

drizzle.scale=0.5

drizzle.pixfrac=1.0

drizzle.coeffs=”header”

drizzle.outnx=2048

drizzle.outny=2048

drizzle.out_un=”counts”

drizzle.in_mask=”static”

drizzle.rot=0.0

cd pc

!\ls -1 ../img??.hhh > ../img.list

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts

group=1 d="_dr" w="_drw"

cd ../wf2

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=2 d="_dr" w="_drw"

cd ../wf3

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=3 d="_dr" w="_drw"

cd ../wf4

loop_driz input=@../img.list output=@../img.list mask=static
   ../average_shifts group=4 d="_dr" w="_drw"

cd ..
```

Here, “img*_dr.hhh” are the drizzled images, and “img*_drw.hhh” are its associated drizzled weight images.

b) Create a Median Image from the Drizzled Images

Create a mask file to be used with **imcombine**. (Note: this mask file should not be confused with the static pixel mask file created earlier. This mask file is made from the weight images created in the previous **drizzle** steps, and will only be used with **imcombine**.) **mask_head** converts the weight image to 1’s and 0’s. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file.

```
unlearn mask_head
cd pc
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ../wf2
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ../wf3
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ../wf4
!\ls -1 *dr.hhh > dr.list
!\ls -1 *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..
```

The output is a pixel list mask file for each image, called “img*_drw.pl”.

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine
imcombine.plfile = ““
imcombine.sigma = ““
imcombine.combine = “median”
imcombine.reject = “minmax”
imcombine.project = no
imcombine.outtype = “real”
imcombine.offsets = “none”
imcombine.masktype = “badvalue”
imcombine.maskvalue = 0.
imcombine.nlow = 1
imcombine.nhigh = 4
imcombine.nkeep = 1
imcombine.scale = “exposure”

cd pc

imcombine *dr.hhh dr_med.hhh

cd ../wf2

imcombine *dr.hhh dr_med.hhh

cd ../wf3

imcombine *dr.hhh dr_med.hhh

cd ../wf4

imcombine *dr.hhh dr_med.hhh

cd ..
```

The output median image for each group is called “dr_med.hhh”.

c) Blot the Combined Image for each Group

The median image for each group is “blotted” or reverse-drizzled back to the 800x800 dimensions of each original image, and also shifted. The results are a blotted counterpart image for each input image.

```
unlearn blot

blot.scale = 0.5

blot.outnx = 800

blot.outny = 800

blot.expout = “2400”  # exposure time of each input image.

blot.coeffs = “header”

blot.rot = 0.

cd pc

loop_blot input=dr_med output=@../img.list ../average_shifts
   group=1 b="_bl"

cd ../wf2

loop_blot input=dr_med output=@../img.list ../average_shifts
   group=2 b="_bl"

cd ../wf3

loop_blot input=dr_med output=@../img.list ../average_shifts
   group=3 b="_bl"

cd ../wf4

loop_blot input=dr_med output=@../img.list ../average_shifts
   group=4 b="_bl"

cd ..
```

The output blotted images are named “img*_bl.hhh”.

d) Create Derivative Images

This step estimates effects caused by blurring in the median image and possible errors in the input shifts.

```
unlearn deriv

cd pc

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf2

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf3

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ../wf4

!\ls -1 *bl.hhh > blot.list

deriv @blot.list

cd ..
```

The output is a derivative image associated with each blotted image, called “img*_bl_deriv.hhh”.

e) Compare each Input Image with its Counterpart Blotted Image to Identify Cosmic Rays.

```
unlearn driz_cr

driz_cr.gain = 7.

driz_cr.rn = 5.

driz_cr.SNR = “4.0 2.5”

driz_cr.scale = “0.9 0.5”

driz_cr.backg = “backgrnd”

cd pc

driz_cr ../img*.hhh 1

cd ../wf2

driz_cr ../img*.hhh 2

cd ../wf3

driz_cr ../img*.hhh 3

cd ../wf4

driz_cr ../img*.hhh 4

cd ..
```

The output for each image is a cosmic ray pixel list mask file called "img*_cr.pl".

Create a "Master" Mask for each Image.

For each chip, the static mask file previously created is multiplied with each individual cosmic ray mask file to create a series of master mask files that will be used in the final **drizzle** operation.

```
unlearn imarith

cd pc

imarith static.hhh * img01_cr.pl img01_cr.pl

imarith static.hhh * img02_cr.pl img02_cr.pl

imarith static.hhh * img03_cr.pl img03_cr.pl

imarith static.hhh * img04_cr.pl img04_cr.pl

imarith static.hhh * img05_cr.pl img05_cr.pl

imarith static.hhh * img06_cr.pl img06_cr.pl

imarith static.hhh * img07_cr.pl img07_cr.pl

imarith static.hhh * img08_cr.pl img08_cr.pl

imarith static.hhh * img09_cr.pl img09_cr.pl

imarith static.hhh * img10_cr.pl img10_cr.pl

imarith static.hhh * img11_cr.pl img11_cr.pl

imarith static.hhh * img12_cr.pl img12_cr.pl

cd ../wf2

imarith static.hhh * img01_cr.pl img01_cr.pl

imarith static.hhh * img02_cr.pl img02_cr.pl

imarith static.hhh * img03_cr.pl img03_cr.pl

imarith static.hhh * img04_cr.pl img04_cr.pl

imarith static.hhh * img05_cr.pl img05_cr.pl

imarith static.hhh * img06_cr.pl img06_cr.pl

imarith static.hhh * img07_cr.pl img07_cr.pl

imarith static.hhh * img08_cr.pl img08_cr.pl

imarith static.hhh * img09_cr.pl img09_cr.pl

imarith static.hhh * img10_cr.pl img10_cr.pl

imarith static.hhh * img11_cr.pl img11_cr.pl

imarith static.hhh * img12_cr.pl img12_cr.pl

cd ../wf3

imarith static.hhh * img01_cr.pl img01_cr.pl

imarith static.hhh * img02_cr.pl img02_cr.pl
imarith static.hhh * img03_cr.pl img03_cr.pl
imarith static.hhh * img04_cr.pl img04_cr.pl
imarith static.hhh * img05_cr.pl img05_cr.pl
imarith static.hhh * img06_cr.pl img06_cr.pl
imarith static.hhh * img07_cr.pl img07_cr.pl
imarith static.hhh * img08_cr.pl img08_cr.pl
imarith static.hhh * img09_cr.pl img09_cr.pl
imarith static.hhh * img10_cr.pl img10_cr.pl
imarith static.hhh * img11_cr.pl img11_cr.pl
imarith static.hhh * img12_cr.pl img12_cr.pl
cd ../wf4
imarith static.hhh * img01_cr.pl img01_cr.pl
imarith static.hhh * img02_cr.pl img02_cr.pl
imarith static.hhh * img03_cr.pl img03_cr.pl
imarith static.hhh * img04_cr.pl img04_cr.pl
imarith static.hhh * img05_cr.pl img05_cr.pl
imarith static.hhh * img06_cr.pl img06_cr.pl
imarith static.hhh * img07_cr.pl img07_cr.pl
imarith static.hhh * img08_cr.pl img08_cr.pl
imarith static.hhh * img09_cr.pl img09_cr.pl
imarith static.hhh * img10_cr.pl img10_cr.pl
imarith static.hhh * img11_cr.pl img11_cr.pl
imarith static.hhh * img12_cr.pl img12_cr.pl
cd ..
```

Construct the Final Drizzled Images.

Each input image is drizzled, applying the shifts and the mask files previously created.

```
unlearn drizzle
drizzle.wt_scl = “exptime”
drizzle.expkey = “exptime”
drizzle.scale = 0.5
drizzle.pixfrac = 0.6
drizzle.coeffs = “header”
drizzle.outnx = 2048
drizzle.outny = 2048
drizzle.out_un = “counts”
drizzle.fillval = 0

drizzle.rot = 0.

cd pc

!\ls -1 img*_cr.pl > mask.list

loop_driz @../img.list final @mask.list ../average_shifts
   group=1 d=""  w="_w"

cd ../wf2

!\ls -1 img*_cr.pl > mask.list

loop_driz @../img.list final @mask.list ../average_shifts
   group=2 d=""  w="_w"

cd ../wf3

!\ls -1 img*_cr.pl > mask.list

loop_driz @../img.list final @mask.list ../average_shifts
   group=3 d=""  w="_w"

cd ../wf4

!\ls -1 img*_cr.pl > mask.list

loop_driz @../img.list final @mask.list ../average_shifts
   group=4 d=""  w="_w"

cd ..
```

The output file for each group is the drizzled image, “final_g*.hhh”, with its associated drizzle weight file, “final_g*w.hhh”. Be sure to tell the image display software to display all 2048x2048 pixels of the drizzled image.

```
set stdimage = imt2048

display pc/final.hhh

display wf2/final.hhh

display wf3/final.hhh

display wf4/final.hhh
```

Example 5 (STIS/CCD): Hubble Deep Field - North

The STIS images in this section are a subset of 50 images taken in parallel to the WFPC2 and NICMOS Hubble Deep Field observations in December 1997. For this example, we picked eighteen datasets; each of these observations are a “cr-split” pair that were combined and cosmic ray-cleaned in the STIS pipeline (**calstis**). The images are from proposal 7781 (PI: Stefi Baum), use the CLEAR filter and have image dimensions of 1024x1024 pixels. (Note: this proposal uses a base32 numbering scheme for visits after 100. These values have been converted to numbers in the LINENUM header keyword.)

A little Organizing to keep track of Files

Copy the original images to working image names (for the science and data quality mask data), then store them in a directory for safekeeping.

```
imcopy o46p7y010_crj.fits[sci] img01.hhh  # visit 7w, exp 10
imcopy o46p7y010_crj.fits[dq]  img01_m.hhh
imcopy o46p81010_crj.fits[sci] img02.hhh  # visit 81, exp 10
imcopy o46p81010_crj.fits[dq]  img02_m.hhh
imcopy o46p82010_crj.fits[sci] img03.hhh  # visit 82, exp 10
imcopy o46p82010_crj.fits[dq]  img03_m.hhh
imcopy o46p83010_crj.fits[sci] img04.hhh  # visit 83, exp 10
imcopy o46p83010_crj.fits[dq]  img04_m.hhh
imcopy o46p84010_crj.fits[sci] img05.hhh  # visit 84, exp 10
imcopy o46p84010_crj.fits[dq]  img05_m.hhh
imcopy o46p8c010_crj.fits[sci] img06.hhh # visit 8a, exp 10
imcopy o46p8c010_crj.fits[dq]  img06_m.hhh
imcopy o46p8d010_crj.fits[sci] img07.hhh  # visit 8b, exp 10
imcopy o46p8d010_crj.fits[dq]  img07_m.hhh
imcopy o46p8e010_crj.fits[sci] img08.hhh  # visit 8c, exp 10
imcopy o46p8e010_crj.fits[dq]  img08_m.hhh
imcopy o46p8f010_crj.fits[sci] img09.hhh  # visit 8d, exp 10
imcopy o46p8f010_crj.fits[dq]  img09_m.hhh
imcopy o46p8g010_crj.fits[sci] img10.hhh  # visit 8e, exp 10
imcopy o46p8g010_crj.fits[dq]  img10_m.hhh
imcopy o46p8h010_crj.fits[sci] img11.hhh  # visit 8f, exp 10
imcopy o46p8h010_crj.fits[dq]  img11_m.hhh
imcopy o46p8k010_crj.fits[sci] img12.hhh  # visit 8g, exp 10
imcopy o46p8k010_crj.fits[dq]  img12_m.hhh
imcopy o46p8l010_crj.fits[sci] img13.hhh  # visit 8h, exp 10
imcopy o46p8l010_crj.fits[dq]  img13_m.hhh
imcopy o46p8m010_crj.fits[sci] img14.hhh  # visit 8i, exp 10
imcopy o46p8m010_crj.fits[dq]  img14_m.hhh
imcopy o46p8n010_crj.fits[sci] img15.hhh  # visit 8j, exp 10
imcopy o46p8n010_crj.fits[dq]  img15_m.hhh
imcopy o46p8o010_crj.fits[sci] img16.hhh  # visit 8k, exp 10
imcopy o46p8o010_crj.fits[dq]  img16_m.hhh
imcopy o46p8p010_crj.fits[sci] img17.hhh  # visit 81, exp 10
imcopy o46p8p010_crj.fits[dq]  img17_m.hhh
imcopy o46p97010_crj.fits[sci] img18.hhh  # visit 97, exp 10
imcopy o46p97010_crj.fits[dq]  img18_m.hhh
```

Store original images for safe-keeping.

```
!mkdir orig
!mv *crj.fits orig
```

Move mask files to a separate working directory.

```
!mkdir mask
!mv *_m.hh? mask
```

Subtract the Sky from the Images

The **sky** task subtracts a constant background value in the image, and populates the header keyword BACKGRND with the value that was subtracted from the image. (STIS images do not have this keyword, but it is added by the **sky** task.)

In the Fruchter-Mutchler drizzle demonstration paper, a suggested method for getting the **width** parameter for the **sky** task was to run **imhistogram** and measure the width of the sky histogram, for example:

```
unlearn imhistogram
imhistogram.z1 = -100.
imhistogram.z2 = 200.
imhistogram.binwidth = INDEF
imhistogram.nbins = 1000
imhistogram.autoscale = yes
imhistogram.top_closed = yes
imhistogram.hist_type = “normal”
imhistogram.listout = no
imhistogram.plot_type = “line”
imhistogram.logy = no
imhist img01.hhh
```

The measured histogram width is quite wide (about 60 DN). Therefore, a test was done: the **sky** task was run using a range of **sky.width** values to see which would yield a final background value near zero. The tested **sky.width** values were 10, 15, 20, 30, 35, 40, and 60. The `sky.width=35` run yielded the best background values. This was evaluated by running **imstat** on the images; these images are rather sparse so the mean value is a good representation of the background. However, for

STIS images with bright sources, you would want to set limits in **imstat.lower** and **upper** to exclude pixels with high counts.

Subtract the sky from the images.

```
unlearn sky
sky.masks = ""
sky.lower = -99.
sky.upper = 30000.   # value close to gain 1 saturation
sky.subsky = yes
sky.width = 35
sky.stat = "mean"
sky.skyname = "BACKGRND"
sky img??.hhh
```

The image sky subtraction is done in-place, and the sky value obtained is written to the "BACKGRND" header keyword.

Prepare the Images for Cross-correlation

This step, where **precor** is run, is skipped because these images were "cr-rejected" in the pipeline.

a) Find the Offsets between the Reference (First) Image and the Input Images.

The images are cross-correlated using the **crossdriz** task. Before proceeding, load the **fourier** package if it has not already been loaded.

```
fourier
```

Next, run **crossdriz** to cross-correlate each image with the reference image (img01.hhh)

```
unlearn crossdriz
crossdriz.dinp=yes
crossdriz.dref=yes
crossdriz.margin=50
crossdriz.tapersz=50
crossdriz.mintheta=0.
crossdriz.maxtheta=0.
crossdriz.stptheta=0.1
crossdriz.coeffs="stis-ccd"
crossdriz.expkey="exptime"
crossdriz.xout=INDEF
crossdriz.yout=INDEF
crossdriz.pad=no
crossdriz img01.hhh img01.hhh cross1x01
crossdriz img02.hhh img01.hhh cross1x02
crossdriz img03.hhh img01.hhh cross1x03
crossdriz img04.hhh img01.hhh cross1x04
crossdriz img05.hhh img01.hhh cross1x05
crossdriz img06.hhh img01.hhh cross1x06
crossdriz img07.hhh img01.hhh cross1x07
crossdriz img08.hhh img01.hhh cross1x08
crossdriz img09.hhh img01.hhh cross1x09
crossdriz img10.hhh img01.hhh cross1x10
crossdriz img11.hhh img01.hhh cross1x11
crossdriz img12.hhh img01.hhh cross1x12
crossdriz img13.hhh img01.hhh cross1x13
crossdriz img14.hhh img01.hhh cross1x14
crossdriz img15.hhh img01.hhh cross1x15
crossdriz img16.hhh img01.hhh cross1x16
crossdriz img17.hhh img01.hhh cross1x17
crossdriz img18.hhh img01.hhh cross1x18
```

Next, use **shiftfind** to determine the shifts between the images and the reference image.

Be sure to load the **fitting** package first.

```
fitting
unlearn shiftfind
shiftfind.x = INDEF
shiftfind.y = INDEF
shiftfind.boxsize = INDEF
shiftfind.fwhm = 7
shiftfind.ellip = 0.05
shiftfind.pa = 45
shiftfind.fitbox = 7
shiftfind.kellip = yes
shiftfind cross1x*.hhh shifts.txt
```

The output is written in “shifts.txt”, a text file containing the shifts.

You’re done with the cross-correlation images, and can delete them.

```
!rm cross1x*
```

Determine the Average Shifts.

This step is omitted because the STIS/CCD image used is a single group image.

Create Static Pixel Mask Files.

Use the data quality file to create a static pixel mask for each image. Note: the objects in this field are mostly faint so pixels marked as saturated are caused by cosmic ray hits. Therefore, they do not have to be excluded from the mask file like we did in Example 1. However, if your image has a bright astrophysical object that uniquely saturates each image (if exposure times are different), then the saturated pixels should be treated as "good" pixels and removed from the mask file.

Convert the data quality files extracted earlier to mask files of 0's and 1's.

```
cd mask
imcalc img01_m.hhh static_01.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img02_m.hhh static_02.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img03_m.hhh static_03.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img04_m.hhh static_04.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img05_m.hhh static_05.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img06_m.hhh static_06.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img07_m.hhh static_07.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img08_m.hhh static_08.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img09_m.hhh static_09.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img10_m.hhh static_10.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img11_m.hhh static_11.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img12_m.hhh static_12.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img13_m.hhh static_13.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img14_m.hhh static_14.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img15_m.hhh static_15.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img16_m.hhh static_16.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img17_m.hhh static_17.hhh "if im1 .gt. 0 then 0 else 1"
imcalc img18_m.hhh static_18.hhh "if im1 .gt. 0 then 0 else 1"
cd ..
```

Make Mask Files to remove Bad Pixels that are unique to each Image

a) Drizzle each Input Image

Each input image is drizzled, applying the shifts obtained earlier in the “shifts.txt” file. The STIS images have dimensions 1024x1024. With *drizzle.scale=0.5*, the drizzled science image portion will be 2048x2048 pixels, imbedded in an image with dimensions 2400x2400.

```
unlearn drizzle

drizzle.wt_scl=”exptime”

drizzle.expkey=”exptime”

drizzle.scale=0.5

drizzle.pixfrac=1.0

drizzle.coeffs=”stis-ccd”

drizzle.outnx=2400

drizzle.outny=2400

drizzle.out_un=”counts”

drizzle img01.hhh img01_dr.hhh outweig=img01_drw.hhh xsh=0.0
   ysh=0.0 in_mask=”mask/static_01.hhh”

drizzle img02.hhh img02_dr.hhh outweig=img02_drw.hhh
   xsh=-2.7615  ysh=27.3945 in_mask=”mask/static_02.hhh”

drizzle img03.hhh img03_dr.hhh outweig=img03_drw.hhh
   xsh=-9.2948  ysh=-8.6604 in_mask=”mask/static_03.hhh”

drizzle img04.hhh img04_dr.hhh outweig=img04_drw.hhh
   xsh=-5.7421  ysh=15.2586 in_mask=”mask/static_04.hhh”

drizzle img05.hhh img05_dr.hhh outweig=img05_drw.hhh
   xsh=4.1039   ysh=8.7520 in_mask=”mask/static_05.hhh”

drizzle img06.hhh img06_dr.hhh outweig=img06_drw.hhh
   xsh=-19.2907 ysh=31.9132 in_mask=”mask/static_06.hhh”

drizzle img07.hhh img07_dr.hhh outweig=img07_drw.hhh
   xsh=12.0354  ysh=24.3447 in_mask=”mask/static_07.hhh”

drizzle img08.hhh img08_dr.hhh outweig=img08_drw.hhh
   xsh=7.0445   ysh=-4.3256 in_mask=”mask/static_08.hhh”

drizzle img09.hhh img09_dr.hhh outweig=img09_drw.hhh
   xsh=9.7879   ysh=17.3008 in_mask=”mask/static_09.hhh”

drizzle img10.hhh img10_dr.hhh outweig=img10_drw.hhh
   xsh=-21.3818 ysh=13.1979 in_mask=”mask/static_10.hhh”

drizzle img11.hhh img11_dr.hhh outweig=img11_drw.hhh
   xsh=-12.7487 ysh=-0.7059 in_mask=”mask/static_11.hhh”

drizzle img12.hhh img12_dr.hhh outweig=img12_drw.hhh
   xsh=-6.3511  ysh=5.2976 in_mask=”mask/static_12.hhh”
```

```
drizzle img13.hhh img13_dr.hhh outweig=img13_drw.hhh
   xsh=2.6087   ysh=-15.2493 in_mask="mask/static_13.hhh"

drizzle img14.hhh img14_dr.hhh outweig=img14_drw.hhh
   xsh=15.9743  ysh=11.8381 in_mask="mask/static_14.hhh"

drizzle img15.hhh img15_dr.hhh outweig=img15_drw.hhh
   xsh=-29.6657 ysh=8.3762 in_mask="mask/static_15.hhh"

drizzle img16.hhh img16_dr.hhh outweig=img16_drw.hhh
   xsh=-17.2250 ysh=17.8361 in_mask="mask/static_16.hhh"

drizzle img17.hhh img17_dr.hhh outweig=img17_drw.hhh
   xsh=-23.9004 ysh=-10.5746 in_mask="mask/static_17.hhh"

drizzle img18.hhh img18_dr.hhh outweig=img18_drw.hhh
   xsh=-7.6149  ysh=5.2057 in_mask="mask/static_18.hhh"
```

Here, "img*_dr.hhh" are the drizzled images, and "img*_drw.hhh" are its associated weight file.

b) Create a Median Image from the Drizzled Images

Create a mask file to be used with **imcombine**. (Note: this mask file should not be confused with the static pixel mask file created earlier. This mask file is made from the weight images created in the previous **drizzle** steps, and will only be used with **imcombine**.) **mask_head** converts the weight images to 1's and 0's. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file.

```
unlearn mask_head

!\ls -1 *_dr.hhh > dr.list

!\ls -1 *_drw.hhh > drw.list

mask_head @dr.list @drw.list invert=no
```

The output is a pixel list mask file for each image, "img*_drw.pl".

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine

imcombine.plfile = ""

imcombine.sigma = ""

imcombine.combine = "median"

imcombine.reject = "minmax"

imcombine.project = no

imcombine.outtype = "real"

imcombine.offsets = "none"

imcombine.masktype = "badvalue"
imcombine.maskvalue = 0.
imcombine.nlow = 3
imcombine.nhigh = 3
imcombine.nkeep = 1
imcombine.scale = “exposure”
imcombine *dr.hhh dr_med.hhh
```

The output is a median-combined image, “dr_med.hhh”.

c) Blot the Combined Image

The median image is “blotted” or reverse-drizzled back to the 800x800 dimensions of each original image, and also shifted. The results are a blotted counterpart image for each input image.

```
unlearn blot
blot.scale = 0.5
blot.outnx = 1024
blot.outny = 1024
blot.expout = “300”
blot.coeffs = “stis-ccd”
blot.rot = 0.
blot dr_med.hhh img01_bl.hhh  xsh=0.0      ysh=0.0
blot dr_med.hhh img02_bl.hhh  xsh=-2.7615  ysh=27.3945
blot dr_med.hhh img03_bl.hhh  xsh=-9.2948  ysh=-8.6604
blot dr_med.hhh img04_bl.hhh  xsh=-5.7421  ysh=15.2586
blot dr_med.hhh img05_bl.hhh  xsh=4.1039   ysh=8.7520
blot dr_med.hhh img06_bl.hhh  xsh=-19.2907 ysh=31.9132
blot dr_med.hhh img07_bl.hhh  xsh=12.0354  ysh=24.3447
blot dr_med.hhh img08_bl.hhh  xsh=7.0445   ysh=-4.3256
blot dr_med.hhh img09_bl.hhh  xsh=9.7879   ysh=17.3008
blot dr_med.hhh img10_bl.hhh  xsh=-21.3818 ysh=13.1979
blot dr_med.hhh img11_bl.hhh  xsh=-12.7487 ysh=-0.7059
blot dr_med.hhh img12_bl.hhh  xsh=-6.3511  ysh=5.297
blot dr_med.hhh img13_bl.hhh  xsh=2.6087   ysh=-15.2493
blot dr_med.hhh img14_bl.hhh  xsh=15.9743  ysh=11.8381
blot dr_med.hhh img15_bl.hhh  xsh=-29.6657 ysh=8.3762
blot dr_med.hhh img16_bl.hhh  xsh=-17.2250 ysh=17.8361
blot dr_med.hhh img17_bl.hhh  xsh=-23.9004 ysh=-10.5746
blot dr_med.hhh img18_bl.hhh  xsh=-7.6149  ysh=5.2057
```

The output blotted images are named “img*_bl.hhh”.

d) Create Derivative Images

This step estimates effects caused by blurring in the median image and possible errors in the input shifts.

```
unlearn deriv

!ls *bl.hhh > blot.list

deriv @blot.list
```

The output is a derivative image associated with each blotted image called “img*_bl_deriv.hhh”.

e) Compare each Input Image with its Counterpart Blotted Image to identify Bad Pixels

```
unlearn driz_cr

driz_cr.gain = 1.

driz_cr.rn = 4.

driz_cr.SNR = “4.0 2.5”

driz_cr.scale = “0.9 0.5”

driz_cr.backg = “backgrnd”

driz_cr img??.hhh 0
```

The resulting mask files are cosmic ray pixel list mask files, “img*_cr.pl”. Be sure to blink the resulting mask files with the original image to make sure all the bad pixels are flagged. If necessary, adjust the **driz_cr.scale** parameters till satisfactory results are obtained.

Create a "Master" Mask for each Image

The static mask file previously created is multiplied with each individual bad pixel mask file. This creates a series of “master” mask files to be used in the final **drizzle** step.

```
unlearn imarith

imarith img01_cr.pl * mask/static_01.hhh  img01_cr.pl

imarith img02_cr.pl * mask/static_02.hhh  img02_cr.pl

imarith img03_cr.pl * mask/static_03.hhh  img03_cr.pl

imarith img04_cr.pl * mask/static_04.hhh  img04_cr.pl

imarith img05_cr.pl * mask/static_05.hhh  img05_cr.pl

imarith img06_cr.pl * mask/static_06.hhh  img06_cr.pl

imarith img07_cr.pl * mask/static_07.hhh  img07_cr.pl

imarith img08_cr.pl * mask/static_08.hhh  img08_cr.pl

imarith img09_cr.pl * mask/static_09.hhh  img09_cr.pl
imarith img10_cr.pl * mask/static_10.hhh  img10_cr.pl
imarith img11_cr.pl * mask/static_11.hhh  img11_cr.pl
imarith img12_cr.pl * mask/static_12.hhh  img12_cr.pl
imarith img13_cr.pl * mask/static_13.hhh  img13_cr.pl
imarith img14_cr.pl * mask/static_14.hhh  img14_cr.pl
imarith img15_cr.pl * mask/static_15.hhh  img15_cr.pl
imarith img16_cr.pl * mask/static_16.hhh  img16_cr.pl
imarith img17_cr.pl * mask/static_17.hhh  img17_cr.pl
imarith img18_cr.pl * mask/static_18.hhh  img18_cr.pl
```

To save space, compress the static and data quality files, or delete them.

```
cd mask
!compress static*.hhd
!compress *m.hhd
cd ..
```

Construct the Final Drizzled Image.

Each input image is drizzled, applying the shifts and the mask files previously created.

```
unlearn drizzle
drizzle.wt_scl = “exptime”
drizzle.expkey = “exptime”
drizzle.scale = 0.5
drizzle.pixfrac = 0.6
drizzle.coeffs = “stis-ccd”
drizzle.outnx = 2400
drizzle.outny = 2400
drizzle.out_un = “counts”
drizzle.fillval = 0
drizzle.rot = 0.
drizzle img01.hhh final.hhh outweig=final_w.hhh
   in_mask=img01_cr.pl xsh=0.0 ysh=0.0
drizzle img02.hhh final.hhh outweig=final_w.hhh
   in_mask=img02_cr.pl xsh=-2.7615 ysh=27.3945
drizzle img03.hhh final.hhh outweig=final_w.hhh
   in_mask=img03_cr.pl xsh=-9.2948 ysh=-8.6604
drizzle img04.hhh final.hhh outweig=final_w.hhh
   in_mask=img04_cr.pl xsh=-5.7421 ysh=15.2586
drizzle img05.hhh final.hhh outweig=final_w.hhh
```

```
   in_mask=img05_cr.pl xsh=4.1039 ysh=8.7520

drizzle img06.hhh final.hhh outweig=final_w.hhh
   in_mask=img06_cr.pl xsh=-19.2907 ysh=31.9132

drizzle img07.hhh final.hhh outweig=final_w.hhh
   in_mask=img07_cr.pl xsh=12.0354 ysh=24.3447

drizzle img08.hhh final.hhh outweig=final_w.hhh
   in_mask=img08_cr.pl xsh=7.0445 ysh=-4.3256

drizzle img09.hhh final.hhh outweig=final_w.hhh
   in_mask=img09_cr.pl xsh=9.7879 ysh=17.3008

drizzle img10.hhh final.hhh outweig=final_w.hhh
   in_mask=img10_cr.pl xsh=-21.3818 ysh=13.1979

drizzle img11.hhh final.hhh outweig=final_w.hhh
   in_mask=img11_cr.pl xsh=-12.7487 ysh=-0.7059

drizzle img12.hhh final.hhh outweig=final_w.hhh
   in_mask=img12_cr.pl xsh=-6.3511 ysh=5.2976

drizzle img13.hhh final.hhh outweig=final_w.hhh
   in_mask=img13_cr.pl xsh=2.6087 ysh=-15.2493

drizzle img14.hhh final.hhh outweig=final_w.hhh
   in_mask=img14_cr.pl xsh=15.9743 ysh=11.8381

drizzle img15.hhh final.hhh outweig=final_w.hhh
   in_mask=img15_cr.pl xsh=-29.6657 ysh=8.3762

drizzle img16.hhh final.hhh outweig=final_w.hhh
   in_mask=img16_cr.pl xsh=-17.2250 ysh=17.8361

drizzle img17.hhh final.hhh outweig=final_w.hhh
   in_mask=img17_cr.pl xsh=-23.9004 ysh=-10.5746

drizzle img18.hhh final.hhh outweig=final_w.hhh
   in_mask=img18_cr.pl xsh=-7.6149 ysh=5.2057
```

The final output drizzled image is “final.hhh”, with an associated drizzle weight image, “final_w.hhh”. Be sure to tell the image display software to display all 3200x3200 pixels of the drizzled image.

```
set stdimage = imt3200

display final.hhh
```

Note: If your IRAF setup does not support 3200x3200 image displays, please ask your local IRAF guru to set it up in the dev$imtoolrc and dev$graphcap files. For more information, please send mail to *help@stsci.edu.*

Example 6. (NICMOS-2): The Cygnus Egg Nebula

Note: Drizzling has the greatest benefit for NIC3 data since it is severely under-sampled. However, there was no suitable non-proprietary data available for use. Therefore, this example uses NIC2 data to illustrate drizzling for NICMOS. Please note that drizzling NIC1 and most NIC2 data does not buy much spatial resolution when compared with the "shift-and-add" mosaicing done in **calnicb**; drizzling, however, will correct for geometric distortion in these cases.

This example uses NICMOS images of the Egg Nebula (CRL 2688) taken in August 1997 as part of the Early Release Observations following Servicing Mission 2 (program ID: 7115, visit 1 exposure 50, PI: Dean Hines). The observations were taken using the NIC2 camera in MULTIACCUM mode. They were implemented using the following optional parameters:

```
PATTERN=SPIRAL-DITH,

NUM-POS=4,

DITH-SIZE=1.5375,

SAMP-SEQ=STEP16,

NSAMP=16
```

This resulted in a pipeline-mosaiced image (n3uv01050_mos.fits) that had been constructed using the following association member images:

```
n3uv01b2r_cal.fits, n3uv01b3r_cal.fits, n3uv01b4r_cal.fits,
   and n3uv01b6r_cal.fits.
```

A little Organizing to keep track of Files

Copy the original images to working image name (for the science and mask data).

```
imcopy n3uv01b2r_cal.fits[1] img1.hhh

imcopy n3uv01b3r_cal.fits[1] img2.hhh

imcopy n3uv01b4r_cal.fits[1] img3.hhh

imcopy n3uv01b6r_cal.fits[1] img4.hhh

imcopy n3uv01b2r_cal.fits[3] img1_dq.hhh

imcopy n3uv01b3r_cal.fits[3] img2_dq.hhh

imcopy n3uv01b4r_cal.fits[3] img3_dq.hhh

imcopy n3uv01b6r_cal.fits[3] img4_dq.hhh
```

Place the original images in a directory for safe-keeping, and compress them to save space.

```
!mkdir orig_imgs
```

```
!mv n3* orig_imgs/

!compress orig_imgs/n3*
```

Note: Scale the Images from Countrate to Counts.

When the NICMOS science image header keyword “unitdone” is set to PERFORMED, the NICMOS image is in units of countrates. Since the **driz_cr** task currently requires total counts, the images should first be multiplied by their exposure time.

```
unlearn keypar

unlearn imcalc

keypar img1.hhh exptime

x = real(keypar.value)

print(“imcalc img1.hhh img1.hhh im1*”//x) | cl

unlearn keypar

unlearn imcalc

keypar img2.hhh exptime

x = real(keypar.value)

print(“imcalc img2.hhh img2.hhh im1*”//x) | cl

unlearn keypar

unlearn imcalc

keypar img3.hhh exptime

x = real(keypar.value)

print(“imcalc img3.hhh img3.hhh im1*”//x) | cl

unlearn keypar

unlearn imcalc

keypar img4.hhh exptime

x = real(keypar.value)

print(“imcalc img4.hhh img4.hhh im1*”//x) | cl
```

Subtract the Sky from the Images.

In the overview, we mentioned that if the background in the input images are not identical, drizzling will create additional noise. Therefore, we use the **sky** task to set the background to zero. However in NICMOS images, this issue becomes quite complicated.

The NICMOS “Pedestal” Effect is a bias level that varies with time. It is also a quadrant-dependent effect, affecting each of the four quadrants in a camera with different levels. Because this effect is random, it cannot be removed in pipeline processing and each image must be dealt with individually. Another bias-related anomaly is called “residual shading”, where a shading effect is seen across each quadrant. This is a temperature-dependent phenomenon that varies with the instrument's thermal state, caused by such things as seasons, level of detector activity, and the slow warm-up of the "ice" over the lifetime of the instrument.

We recommend you read more about the "Pedestal" Effect before proceeding, to determine if it is an important consideration in your data reduction and analysis. For more details, please refer to the NICMOS website:

http://www.stsci.edu/instruments/nicmos/

The images used in this example are of an extended nebulous field, and have a high signal-to-noise ratio. Therefore, the "Pedestal" Effect is relatively minor and we will ignore it, and omit the sky subtraction step.

Find the Offsets between the Reference (First) Image and the Input Images

The images are cross-correlated using the **crossdriz** task. Before proceeding, load the **fourier** package if it has not already been loaded.

```
fourier
```

Next, run **crossdriz** to cross-correlate each image with the reference image ("img1.hhh").

```
unlearn crossdriz

crossdriz.dinp = yes

crossdriz.dref = yes

crossdriz.margin = 10

crossdriz.tapersz = 10

crossdriz.mintheta = 0.

crossdriz.maxtheta = 0.

crossdriz.stptheta = 0.1

crossdriz.coeffs = "nic2_coeffs"

crossdriz.expkey = "exptime"

crossdriz.xout = INDEF

crossdriz.yout = INDEF

crossdriz.pad = no

crossdriz img1.hhh img1.hhh cross1x1

crossdriz img2.hhh img1.hhh cross1x2

crossdriz img3.hhh img1.hhh cross1x3

crossdriz img4.hhh img1.hhh cross1x4
```

Next, use **shiftfind** to determine the shifts between the images and the reference image. Be sure to load the **fitting** package first.

```
fitting
```

```
unlearn shiftfind

shiftfind.x = INDEF

shiftfind.y = INDEF

shiftfind.boxsize = INDEF

shiftfind.fwhm = 7.

shiftfind.ellip = 0.05

shiftfind.pa = 45.

shiftfind.fitbox = 7

shiftfind.kellip = yes

shiftfind.kpa = yes

shiftfind cross1x*.hhh shifts.txt
```

The output is written to "shifts.txt", a text file containing the shifts. You're done with the cross-correlation images, and can delete them.

```
!rm cross1x*
```

Create a Static Pixel Mask File

Here, we use the data quality files to create a static pixel mask for each image. All four data quality files were compared, and found to be identical. Therefore, one was picked to create the static pixel mask file.

Copy one data quality file to a temporary working file.

```
!cp img1_dq.hhh static_tmp.hhh
```

Remove the data quality files; we don't need them anymore.

```
!rm *dq.hh?
```

Convert the temporary working data quality file to a mask file of 0's and 1's, then delete it.

```
unlearn imcalc

imcalc static_tmp.hhh static.hhh “if im1 .gt. 0 then 0 else 1”

!rm static_tmp.hhh
```

Make Mask Files to remove Bad Pixels that are Unique to each Image

a) Drizzle each Input Image

Each input image is drizzled, applying the shifts obtained earlier in the “shifts.txt” file. (The NIC2 images have dimensions 256x256. With a scale=0.5, the drizzled science image portion will be 512x512 pixels, imbedded in a 1024x1024 image.)

```
unlearn drizzle

drizzle.wt_scl = “exptime”

drizzle.expkey = “exptime”
drizzle.scale = 0.5
drizzle.pixfrac = 1.0
drizzle.coeffs = "nic2_coeffs"
drizzle.outnx = 1024
drizzle.outny = 1024
drizzle.in_un = "counts"
drizzle.out_un = "counts"
drizzle.in_mask = "static.hhh"
drizzle.rot = 0.0
drizzle img1.hhh img1_dr.hhh outweig=img1_drw.hhh xsh=0.0
   ysh=0.0
drizzle img2.hhh img2_dr.hhh outweig=img2_drw.hhh xsh=20.1161
   ysh=-0.0445
drizzle img3.hhh img3_dr.hhh outweig=img3_drw.hhh xsh=20.1786
   ysh=20.0245
drizzle img4.hhh img4_dr.hhh outweig=img4_drw.hhh xsh=0.0231
   ysh=20.1031
```

Here, "img*_dr.hhh" are the drizzled images, and "img*_drw.hhh" are its associated drizzle weight files.

b) Create a Median Image from the Drizzled Images

Create a mask file to be used with **imcombine**. (Note: this mask file should not be confused with the static pixel mask file created earlier. This mask file is made from the weight images created in the previous **drizzle** steps, and will only be used with **imcombine**.) **mask_head** converts the weight image to 1's and 0's. It also inserts a new keyword in the drizzled image header file, called BPM, that contains the name of the mask file.

```
!\ls -1 img?_dr.hhh > dr.list
!\ls -1 img?_drw.hhh > drw.list
unlearn mask_head
mask_head @dr.list @drw.list invert=no
```

The output is a pixel list mask file for each image, "img*_drw.pl".

Combine the drizzled images into a single median image. The **imcombine** task will also check for the BPM keyword in the drizzled image headers to determine if the input images are associated with mask files.

```
unlearn imcombine
imcombine.combine = "median"
imcombine.reject = "minmax"
imcombine.maskvalue = 0.
imcombine.masktype = "badvalue"

imcombine.nlow = 2

imcombine.nhigh = 1

imcombine.nkeep = 1

imcombine *dr.hhh dr_med.hhh
```

The output is a median-combined image, "dr_med.hhh".

c) Blot the Combined Image.

The median image is "blotted" or reverse-drizzled back to the 256x256 dimensions of each original image, and also shifted. The results are a blotted counterpart image for each input image.

```
unlearn blot

blot.scale=0.5

blot.outnx=256

blot.outny=256

blot.expout=159.909760  # exp time of each input image

blot.coeffs="nic2_coeffs"

blot.in_un = "counts"

blot.out_un = "counts"

blot.rot=0.0

blot dr_med.hhh img1_bl.hhh xsh=0.0     ysh=0.0

blot dr_med.hhh img2_bl.hhh xsh=20.1161 ysh=-0.0445

blot dr_med.hhh img3_bl.hhh xsh=20.1786 ysh=20.0245

blot dr_med.hhh img4_bl.hhh xsh=0.0231  ysh=20.1031
```

The output blotted images are named "img*_bl.hhh".

d) Create Derivative Images

This step estimates effects caused by blurring in the medianed image and possible errors in the input shifts.

```
!\ls -1 *bl.hhh > blot.list

deriv @blot.list
```

The output is a derivative image associated with each blotted image called "img*_bl_deriv.hhh".

e) Compare each Input Image with its Counterpart Blotted Image to identify Bad Pixels

```
unlearn driz_cr

driz_cr.gain=5.4

driz_cr.rn=30
driz_cr.SNR=”4.0 3.0”

driz_cr.scale=”3.0 2.5”

driz_cr img?.hhh ““
```

The results are cosmic ray pixel list mask files, “img*_cr.pl”. Be sure to blink these cosmic ray mask files with its associated original image to make sure all the bad pixels are flagged. This can be an iterative process; for this example, we tried different values for **driz_cr.scale** until satisfactory results were achieved.

Note: for NIC3 data, the peaks of stars are sometimes flagged as cosmic rays in the cosmic ray mask file created by **driz_cr**. This is due to undersampling of the images that result in pixel sensitivity changing over the field-of-view from one dithered image to another. (For example, the peak of a star may fall in the center of the pixel for one image, and near the corner for another image.) As a result, the peak value of the PSF could change significantly, making the **driz_cr** software think it detected a cosmic ray. One way to fix this problem is to blink your final drizzled image with its weight file. If you notice zero weights at the peaks of your stars, it is likely that **driz_cr** interpreted it as a cosmic ray. You would therefore have to manually edit the cosmic ray mask to re-flag that pixel as a good value.

Create a "Master" Mask for each Image

The static mask file is multiplied with each individual bad pixel mask file. This creates a series of “master” mask files to be used in the final **drizzle** operation.

```
unlearn imarith

imarith img1_cr.pl * static.hhh img1_cr.pl

imarith img2_cr.pl * static.hhh img2_cr.pl

imarith img3_cr.pl * static.hhh img3_cr.pl

imarith img4_cr.pl * static.hhh img4_cr.pl
```

Construct the Final Drizzled Image

Each input image is drizzled onto a single output image, applying the shifts and the mask files previously created.

```
unlearn drizzle

drizzle.wt_scl = “exptime”

drizzle.expkey = “exptime”

drizzle.scale = 0.5

drizzle.pixfrac = 0.9

drizzle.coeffs = “nic2_coeffs”

drizzle.outnx = 1024
drizzle.outny = 1024

drizzle.in_un = "counts"

drizzle.out_un = "counts"

drizzle.fillval = 0.0

drizzle.rot = 0.

drizzle img1.hhh final.hhh outweig=final_w.hhh
   in_mask=img1_cr.pl xsh=0.0 ysh=0.0

drizzle img2.hhh final.hhh outweig=final_w.hhh
   in_mask=img2_cr.pl xsh=20.1161 ysh=-0.0445

drizzle img3.hhh final.hhh outweig=final_w.hhh
   in_mask=img3_cr.pl xsh=20.1786 ysh=20.0245

drizzle img4.hhh final.hhh outweig=final_w.hhh
   in_mask=img4_cr.pl xsh=0.0231  ysh=20.1031
```

The final drizzled image is called "final.hhh", with its associated drizzle weight file "final_w.hhh". Be sure to tell the image display software to display all 1024x1024 pixels of the drizzled image.

```
set stdimage = imt1024

display final.hhh
```

Bibliography

Dickinson, M., et al., 1999, "NICMOS Data Handbook, Version 4.0", (Baltimore: STScI),

`http://www.stsci.edu/cgi-bin/NICMOS/si.pl?nav=documents:handbooks&sel=id:512`

Fruchter, A. S., "Andy's Dither Page",

`http://www.stsci.edu/~fruchter/dither/`

Fruchter, A. S., "The Hubble Deep Field - Drizzling"

`http://www.stsci.edu/ftp/science/hdf/combination/drizzle.html`

Fruchter, A. S., "The Hubble Deep Field - Image Registration and Combination",

`http://www.stsci.edu/ftp/science/hdf/combination/combination.html`

Fruchter, A. S., Hook, R. N., 1997, "A Method for the Linear Reconstruction of Undersampled Images", *PASP (submitted)*; astro-ph/9808087,

`http://xxx.lanl.gov/abs/astro-ph/9808087`

Fruchter, A. S., Hook, R. N., "Linear Reconstruction of the Hubble Deep Field",

`http://www.stsci.edu/~fruchter/dither/drizzle.html`

Fruchter, A. S., Hook, R. N., Busko, I. C., Mutchler, M., 1997, "A Package for the Reduction of Dithered Undersampled Images", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/fruchtera.ps`

Fruchter, A. S., Mutchler, M., "Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration",

`http://www.stsci.edu/~fruchter/dither/ditherII.ps`

Gonzaga, S., Biretta, J., Wiggs, M., et al., 1998, "The Drizzling Cookbook", *ISR WFPC2 98-04*,

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/drizzle_cookbook.ps`

Gonzaga, S., Wiggs, M., "WFPC2 Drizzling Cookbook Page",

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/wfpc2_driz_ckbk.html`

Gonzaga, S., Wiggs, M., "Datasets and IRAF Scripts for use with the Drizzle Cookbook",

`http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/wfpc2_driz_ckbk.html`

HDF-S Team, "The Hubble Deep Field South - Data Reduction / Technical Information",

`http://www.stsci.edu/ftp/science/hdfsouth/hdfs.html`

Mutchler, M., Fruchter, A. S., 1997, "Drizzling Dithered WFPC2 Images - A Demonstration", in *1997 HST Calibration Workshop*, ed. S. Casertano et al.,

`http://icarus.stsci.edu/~stefano/CAL97_PAPERS/mutchlerm.ps`