Instrument Science Report WFPC2 98-04

The Drizzling Cookbook

Shireen Gonzaga, John Biretta, Michael S. Wiggs, J. C. Hsu, T. Ed Smith, Louis Bergeron, and the STScI WFPC2 Group[1]
December 15, 1998.

ABSTRACT

The drizzle software combines dithered images while preserving photometric accuracy, enhancing resolution, and removing geometric distortion. A recent upgrade also allows removal of cosmic rays from single images at each dither pointing. This document gives detailed examples illustrating drizzling procedures for six cases: WFPC2 observations of a deep field, a crowded field, a large galaxy, and a planetary nebula, STIS/CCD observations of a Hubble Deep Field-North field, and NICMOS/NIC2 observations of the Egg Nebula. Command scripts and input images for each example are available at the WFPC2 WWW website. Users are encouraged to retrieve the data for the case that most closely resembles their own data, and then practice and experiment drizzling the example.

1. Sylvia Baggett, John Biretta, Stefano Casertano, Shireen Gonzaga, Inge Heyer, Matt McMaster, Christopher O'Dea, Michael Wiggs.

Table of Contents

1.0 Introduction.

The Drizzle software was developed by Andrew Fruchter (Space Telescope Science Institute), and Richard Hook (Space Telescope European Coordinating Facility) as a tool for image reconstruction of undersampled, dithered WFPC2 images. A suite of tasks for processing dithered data can be found in STSDAS V2.0.2, in the *stsdas.analysis.dither* package. A beta version of new tasks (that will eventually be included in the STSDAS *dither* package), can be found in the *ditherII* package, and should be downloaded from the Dither webpage *http://www.stsci.edu/~fruchter/dither/#DitherII* . This tutorial uses several of the *ditherII* tasks so we strongly recommend downloading this software.

This document will not cover details behind the algorithms used in the *dither* package. For more information, users should refer to several excellent papers by Fruchter et al., that are available at the above-mentioned Drizzle webpage. A brief description and figure, extracted from "A Method for the Linear Reconstruction of Undersampled Images," by Fruchter and Hook, to be published in the PASP, is provided below:

> Although the effect of Drizzle on the quality of the image can be profound, the 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, we allow the user to shrink the pixel before it is averaged into the output image [as shown in Figure 1].
>
> The new shrunken pixels, or "drops," rain down upon the subsampled output. In the case of the HDF [Hubble Deep Field], the drops used had linear dimensions one-half that of the input pixel--slightly larger than the dimensions of the output subsampled pixels. The value of an input pixel is averaged into an output pixel with a weight proportional to the area of overlap between the "drop" and the output pixel. Note that if the drop size is sufficiently small, not all output pixels have data added to them from each input image. One must therefore choose a drop size that is small enough to avoid degrading the image, but large enough so that after all images are drizzled the coverage is reasonably uniform.
>
> The drop size is controlled by a user-adjustable parameter called pixfrac, which is simply the ratio of the linear size of the drop to the input pixel (before any adjustment due to the geometric distortion of the camera). Thus interlacing is equivalent to Drizzle in the limit of pixfrax --> 0.0, while shift-and-add is equivalent to pixfrac=1.0.

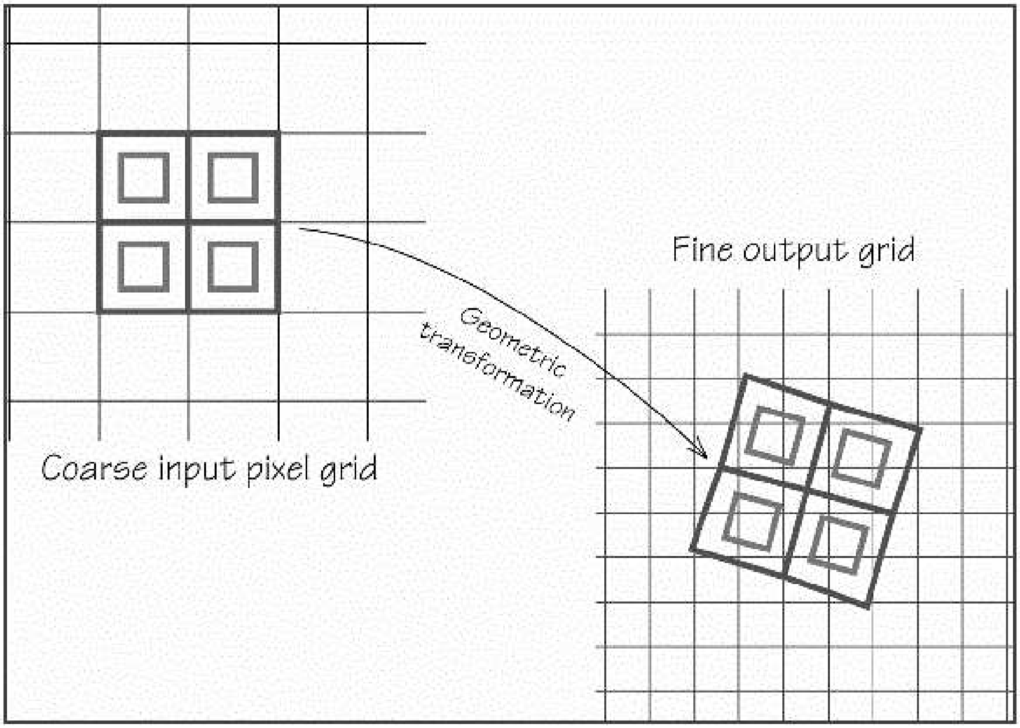

Figure 1: A schematic representation of Drizzle. The input pixel grid (shown on the left) is mapped onto a finer output grid (shown on right), taking into account shift, rotation, and geometric distortion. The user is allowed to “shrink” the input pixels to smaller pixels, which we refer to as drops (inner squares). A given input image only affects output image pixels under drops. In this particular case, the central output pixel receives no information from the input image.

2.0 Overview of Processing Dithered Images:

The *stsdas.analysis.dither* package contains the following tasks:

avshift	Averages the shifts measured on the 4 WFPC chips.
blot	Image sampling using interpolation (reverse “drizzling”).
crossdriz	Builds a set of cross-correlation images (shift + rotation).
drizzle	Perform linear image reconstruction using “drizzling”.
dunlearn	Resets all task parameters to default values.
sky	Sky processing by “*crrej*-like” algorithm.
imextreme	Locates the maximum and minimum pixels in an image.
offsets	Builds cross-correlation image (shift only).
precor	Cosmic-ray processing.
rotfind	Finds rotation angle from a set of cross-correlation images.
shiftfind	Finds X,Y shifts in a cross-correlation image.
cdriz @	Pset for the *offsets* task.
dq @	Pset for the *sky* task.

The *ditherII* package, available from the Dither website, has these tasks:

deriv	Take the absolute derivative of an image.
driz_cr	Create cosmic ray mask based for dithered data.
mask_head	Attach mask names to image headers; invert mask if requested.

The Drizzling Procedure:

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

In drizzling, the pixels from different images are interlaced. If the background in the images are not identical, drizzling will create additional noise. Unequal background levels sometimes occurs 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.

2.2. 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 is 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 *precor* 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*.

2.3. 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 4, page 32 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 1 and 2 in Chapter 3, 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 file (f8213081u.r0h/.r0d), used in pipeline calibration, to 1's and 0's. This file can be downloaded from *http://www.stsci.edu/ftp/cdbs/cdbs3/uref/*. This mask file does not contain warm pixel information, it only flags static 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 (Section 3.1), 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 mage.

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 neighbours 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 neighbours 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.

2.4. 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 created in Section 2.3. 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 RMS 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.

3.0 Examples of using the Drizzling Software.

We illustrate the drizzling 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: WFPC2 images of the Hawaii Deep Survey Field SSA22, a sparse deep field.
Example #2: WFPC2 images of a crowded stellar field in the Large Magellanic Cloud bar.
Example #3: WFPC2 images of the nucleus of galaxy NGC4565.
Example #4: PC images of planetary nebula IRAS 17150-3224.
Example #5: STIS/CCD images of the Hubble Deep Field - North.
Example #6: NICMOS Camera 2 images of the Egg Nebula.

Note: Examples 1, 2, and 4 also illustrate three different ways of constructing a static pixel mask file: the first example 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. The second example uses the static pixel mask reference file (used during pipeline calibration of the data). The fourth example uses the data quality file of the recalibrated images. No static pixel mask file is created in example 3 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 1, is the correlation between the number of images used to construct the final drizzled image, the instrument used, and the *drizzle* parameters (*drizzle.scale* and drizzle.pixfrac). Examples 1 and 5 have many input image, and can therefore be processed with smaller *drizzle.pixfrac* values. But examples 2, 3, 4, and 6 have few input images, and therefore require larger *drizzle.pixfrac* values.

The intermediate and final drizzled images can take up a lot of space, and running the tasks in the *dither* and *ditherII* packages can be quite CPU-intensive. Table 1 shows the amount of space and run-time needed for each example in this document. The machine in this example is a Sparc Ultra10.

Table 1:

Example	Instrument	# of input images	Image dimensions	Disk space for input files (MB)	Disk space for input, intermediate, and final files (MB)	CPU (minutes)
1	WFPC2	12	800x800x4	181	2558.4	83
2	WFPC2	8	800x800x4	120.7	1727.2	53
3	WFPC2	3	800x800x4	60.3	796.9	20
4	WFPC2 (PC)	5	800x800	37.9	314.9	8
5	STIS/CCD	18	1024x1024	185.4	1291.3	66
6	NIC2	4	256x256	4.4	53.7	1.5

Table 2 contains a checklist of steps for each example in this document.

Table 2:

Steps	Ex. 1 WFPC2	Ex. 2 WFPC2	Ex. 3 WFPC2	Ex. 4 WFPC2 (PC)	Ex. 5 STIS/ CCD	Ex. 6 NICMOS (NIC2)
1. Create working images.	yes	yes	yes	yes	yes	yes
2. Subtract sky from images.	yes	yes	see note[1]	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).	yes	yes	yes	no	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.	yes	yes	no	yes	yes	yes
9. Create final drizzled image.	yes	yes	yes	yes	yes	yes

1. Sky subtraction was not possible for images in Example 3, but a test was done to determine that the background level for each image was the same.

Before running any of the examples in this document, please make sure you have:

A. Installed the *ditherII* package, downloadable from *http://www.stsci.edu/~fruchter/dither/#DitherII*. Instructions on how to do this can be found on the website.

B. Loaded the following tasks in *iraf*.

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

C. Set the default image type in *iraf*:

```
set imtype = hhh
```

D. Set the image display area.

For instance, a WFPC2 2048x2048 drizzled image needs a 2048x2048 display area:

```
set stdimage = imt2048
```

E. Set the scratch directory to your working directory.

Many *dither* tasks use the “tmp” directory for scratch. The temporary files can be quite big and there may not be enough space in your default “tmp” directory (usually /tmp) to accomodate them. By setting your “tmp” directory to your working area, you could avoid running out of space.

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

F. Download the images for the example of your choice from the WFPC2 WWW site.

The examples in this document are available as scripts and images that can be downloaded from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.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.

3.1. Example #1: A Sparse Deep Field.

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). These images can be downloaded from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html*. This site also has two scripts containing all commands used in this example.

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. Please also refer to the above-mentioned paper for additional information about drizzling these datasets.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.)

3.1.1. A little Organization 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/
!compress orig/*.c1d
```

3.1.2. Subtract the Sky from the Images.

Before proceeding, 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.

3.1.3. 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.

3.1.4. 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.

```
fourier
```

<u>Note:</u> *cdriz* is a pset (sub-task) called by *offsets*.

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

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

```
!ls *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:

```
fitting

unlearn shiftfind
shiftfind.xcenter = INDEF
shiftfind.ycenter = 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.

3.1.5. 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 temp
!mv *obj*.hh? cross1x* temp/
```

3.1.6. 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     # zero out a pyramid edge
imrep static.hhh[1:800,1:50] 0     # zero out a pyramid edge
imrep static.hhh[799:800,1:800] 0  # zero out an edge
imrep static.hhh[1:800,799:800] 0  # zero out an edge
cd ..

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 ..

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 ..

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 ..
```

3.1.7. Make Cosmic Ray Masks.

3.1.7.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
drizzle ../img01.hhh[1] img01_dr.hhh outweig=img01_drw.hhh xsh=0.0  ysh=0.0
drizzle ../img02.hhh[1] img02_dr.hhh outweig=img02_drw.hhh xsh=-109.724 ysh=0.473
drizzle ../img03.hhh[1] img03_dr.hhh outweig=img03_drw.hhh xsh=-110.251 ysh=-109.38
drizzle ../img04.hhh[1] img04_dr.hhh outweig=img04_drw.hhh xsh=-0.434  ysh=-109.688
drizzle ../img05.hhh[1] img05_dr.hhh outweig=img05_drw.hhh xsh=108.885 ysh=-110.605
drizzle ../img06.hhh[1] img06_dr.hhh outweig=img06_drw.hhh xsh=109.589  ysh=-0.622
drizzle ../img07.hhh[1] img07_dr.hhh outweig=img07_drw.hhh xsh=-0.740  ysh=0.785
```

```
drizzle ../img08.hhh[1] img08_dr.hhh outweig=img08_drw.hhh xsh=-110.412  ysh=1.317
drizzle ../img09.hhh[1] img09_dr.hhh outweig=img09_drw.hhh xsh=-110.887 ysh=-108.51
drizzle ../img10.hhh[1] img10_dr.hhh outweig=img10_drw.hhh xsh=-1.103  ysh=-108.927
drizzle ../img11.hhh[1] img11_dr.hhh outweig=img11_drw.hhh xsh=108.880 ysh=-109.438
drizzle ../img12.hhh[1] img12_dr.hhh outweig=img12_drw.hhh xsh=108.999  ysh=-0.193
cd ..

cd wf2
drizzle ../img01.hhh[2] img01_dr.hhh outweig=img01_drw.hhh xsh=0.0  ysh=0.0
drizzle ../img02.hhh[2] img02_dr.hhh outweig=img02_drw.hhh xsh=-0.239  ysh=50.197
drizzle ../img03.hhh[2] img03_dr.hhh outweig=img03_drw.hhh xsh=-50.498  ysh=49.982
drizzle ../img04.hhh[2] img04_dr.hhh outweig=img04_drw.hhh xsh=-50.180  ysh=-0.257
drizzle ../img05.hhh[2] img05_dr.hhh outweig=img05_drw.hhh xsh=-50.146  ysh=-50.270
drizzle ../img06.hhh[2] img06_dr.hhh outweig=img06_drw.hhh xsh=0.170  ysh=-50.136
drizzle ../img07.hhh[2] img07_dr.hhh outweig=img07_drw.hhh xsh=0.356  ysh=0.342
drizzle ../img08.hhh[2] img08_dr.hhh outweig=img08_drw.hhh xsh=0.144  ysh=50.515
drizzle ../img09.hhh[2] img09_dr.hhh outweig=img09_drw.hhh xsh=-50.100  ysh=50.276
drizzle ../img10.hhh[2] img10_dr.hhh outweig=img10_drw.hhh xsh=-49.835  ysh=0.052
drizzle ../img11.hhh[2] img11_dr.hhh outweig=img11_drw.hhh xsh=-49.612  ysh=-50.263
drizzle ../img12.hhh[2] img12_dr.hhh outweig=img12_drw.hhh xsh=0.365  ysh=-49.864
cd ..

cd wf3
drizzle ../img01.hhh[3] img01_dr.hhh outweig=img01_drw.hhh xsh=0.0  ysh=0.0
drizzle ../img02.hhh[3] img02_dr.hhh outweig=img02_drw.hhh xsh=50.197  ysh=-0.032
drizzle ../img03.hhh[3] img03_dr.hhh outweig=img03_drw.hhh xsh=50.254  ysh=50.227
drizzle ../img04.hhh[3] img04_dr.hhh outweig=img04_drw.hhh xsh=0.014  ysh=50.181
drizzle ../img05.hhh[3] img05_dr.hhh outweig=img05_drw.hhh xsh=-49.998  ysh=50.417
drizzle ../img06.hhh[3] img06_dr.hhh outweig=img06_drw.hhh xsh=-50.136  ysh=0.101
drizzle ../img07.hhh[3] img07_dr.hhh outweig=img07_drw.hhh xsh=0.340   ysh=-0.358
drizzle ../img08.hhh[3] img08_dr.hhh outweig=img08_drw.hhh xsh=50.514  ysh=-0.417
drizzle ../img09.hhh[3] img09_dr.hhh outweig=img09_drw.hhh xsh=50.547  ysh=49.828
drizzle ../img10.hhh[3] img10_dr.hhh outweig=img10_drw.hhh xsh=0.322  ysh=49.834
drizzle ../img11.hhh[3] img11_dr.hhh outweig=img11_drw.hhh xsh=-49.994  ysh=49.883
drizzle ../img12.hhh[3] img12_dr.hhh outweig=img12_drw.hhh xsh=-49.865  ysh=-0.095
cd ..

cd wf4
drizzle ../img01.hhh[4] img01_dr.hhh outweig=img01_drw.hhh xsh=0.0  ysh=0.0
drizzle ../img02.hhh[4] img02_dr.hhh outweig=img02_drw.hhh xsh=-0.514  ysh=-50.195
drizzle ../img03.hhh[4] img03_dr.hhh outweig=img03_drw.hhh xsh=49.743  ysh=-50.734
drizzle ../img04.hhh[4] img04_dr.hhh outweig=img04_drw.hhh xsh=50.179  ysh=-0.496
drizzle ../img05.hhh[4] img05_dr.hhh outweig=img05_drw.hhh xsh=50.895  ysh=49.512
drizzle ../img06.hhh[4] img06_dr.hhh outweig=img06_drw.hhh xsh=0.582  ysh=50.133
drizzle ../img07.hhh[4] img07_dr.hhh outweig=img07_drw.hhh xsh=-0.361  ysh=-0.336
drizzle ../img08.hhh[4] img08_dr.hhh outweig=img08_drw.hhh xsh=-0.902  ysh=-50.507
drizzle ../img09.hhh[4] img09_dr.hhh outweig=img09_drw.hhh xsh=49.340  ysh=-51.023
drizzle ../img10.hhh[4] img10_dr.hhh outweig=img10_drw.hhh xsh=49.829  ysh=-0.800
drizzle ../img11.hhh[4] img11_dr.hhh outweig=img11_drw.hhh xsh=50.361  ysh=49.513
drizzle ../img12.hhh[4] img12_dr.hhh outweig=img12_drw.hhh xsh=0.384  ysh=49.864
cd ..
```

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

3.1.7.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 *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf2
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf3
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf4
!ls *dr.hhh > dr.list
!ls *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 ..

cd wf2
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf3
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf4
imcombine *dr.hhh dr_med.hhh
cd ..
```

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

3.1.7.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"  # This is the exposure time of each input image.
blot.coeffs = "header"
blot.rot = 0.

cd pc
blot dr_med img01_bl  xsh=0.0  ysh=0.0
blot dr_med img02_bl  xsh=-109.724  ysh=0.473
blot dr_med img03_bl  xsh=-110.251  ysh=-109.387
blot dr_med img04_bl  xsh=-0.434  ysh=-109.688
blot dr_med img05_bl  xsh=108.885  ysh=-110.605
blot dr_med img06_bl  xsh=109.589  ysh=-0.622
blot dr_med img07_bl  xsh=-0.740  ysh=0.785
blot dr_med img08_bl  xsh=-110.412  ysh=1.317
blot dr_med img09_bl  xsh=-110.887  ysh=-108.511
blot dr_med img10_bl  xsh=-1.103  ysh=-108.927
blot dr_med img11_bl  xsh=108.880  ysh=-109.438
blot dr_med img12_bl  xsh=108.999  ysh=-0.193
cd ..

cd wf2
blot dr_med img01_bl  xsh=0.0  ysh=0.0
blot dr_med img02_bl  xsh=-0.239  ysh=50.197
blot dr_med img03_bl  xsh=-50.498  ysh=49.982
```

```
blot dr_med img04_bl  xsh=-50.180  ysh=-0.257
blot dr_med img05_bl  xsh=-50.146  ysh=-50.270
blot dr_med img06_bl  xsh=0.170  ysh=-50.136
blot dr_med img07_bl  xsh=0.356  ysh=0.342
blot dr_med img08_bl  xsh=0.144  ysh=50.515
blot dr_med img09_bl  xsh=-50.100  ysh=50.276
blot dr_med img10_bl  xsh=-49.835  ysh=0.052
blot dr_med img11_bl  xsh=-49.612  ysh=-50.263
blot dr_med img12_bl  xsh=0.365  ysh=-49.864
cd ..

cd wf3
blot dr_med img01_bl  xsh=0.0  ysh=0.0
blot dr_med img02_bl  xsh=50.197  ysh=-0.032
blot dr_med img03_bl  xsh=50.254  ysh=50.227
blot dr_med img04_bl  xsh=0.014  ysh=50.181
blot dr_med img05_bl  xsh=-49.998  ysh=50.417
blot dr_med img06_bl  xsh=-50.136  ysh=0.101
blot dr_med img07_bl  xsh=0.340  ysh=-0.358
blot dr_med img08_bl  xsh=50.514  ysh=-0.41
blot dr_med img09_bl  xsh=50.547  ysh=49.828
blot dr_med img10_bl  xsh=0.322  ysh=49.834
blot dr_med img11_bl  xsh=-49.994  ysh=49.883
blot dr_med img12_bl  xsh=-49.865  ysh=-0.095
cd ..

cd wf4
blot dr_med img01_bl  xsh=0.0  ysh=0.0
blot dr_med img02_bl  xsh=-0.514  ysh=-50.195
blot dr_med img03_bl  xsh=49.743  ysh=-50.734
blot dr_med img04_bl  xsh=50.179  ysh=-0.496
blot dr_med img05_bl  xsh=50.895  ysh=49.512
blot dr_med img06_bl  xsh=0.582  ysh=50.133
blot dr_med img07_bl  xsh=-0.361  ysh=-0.336
blot dr_med img08_bl  xsh=-0.902  ysh=-50.507
blot dr_med img09_bl  xsh=49.340  ysh=-51.023
blot dr_med img10_bl  xsh=49.829  ysh=-0.800
blot dr_med img11_bl  xsh=50.361  ysh=49.513
blot dr_med img12_bl  xsh=0.384  ysh=49.864
cd ..
```

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

3.1.7.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 *bl.hhh > blot.list
```

```
deriv @blot.list
cd ..

cd wf2
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf3
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf4
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..
```

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

3.1.7.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 ..

cd wf2
driz_cr ../img*.hhh 2
cd ..

cd wf3
driz_cr ../img*.hhh 3
cd ..

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”.

3.1.8. Create a “Master” Mask for each Image.

For each chip, the static mask file created in Section 3.1.6 is multiplied with each cosmic ray mask file created in Section 3.1.7 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 ..

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 ..

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 ..

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 ..
```

3.1.9. Construct the Final Drizzled Images.

Each input image is drizzled, applying the shifts and the mask files created in Section 3.1.8.

```
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
drizzle ../img01.hhh[1] final_g1 outweig=final_g1w in_mask=img01_cr.pl \
xsh=0.0  ysh=0.0
drizzle ../img02.hhh[1] final_g1 outweig=final_g1w in_mask=img02_cr.pl \
xsh=-109.724  ysh=0.473
drizzle ../img03.hhh[1] final_g1 outweig=final_g1w in_mask=img03_cr.pl \
xsh=-110.251  ysh=-109.387
drizzle ../img04.hhh[1] final_g1 outweig=final_g1w in_mask=img04_cr.pl \
xsh=-0.434  ysh=-109.688
drizzle ../img05.hhh[1] final_g1 outweig=final_g1w in_mask=img05_cr.pl \
xsh=108.885  ysh=-110.605
drizzle ../img06.hhh[1] final_g1 outweig=final_g1w in_mask=img06_cr.pl \
xsh=109.589  ysh=-0.622
drizzle ../img07.hhh[1] final_g1 outweig=final_g1w in_mask=img07_cr.pl \
xsh=-0.740  ysh=0.785
drizzle ../img08.hhh[1] final_g1 outweig=final_g1w in_mask=img08_cr.pl \
xsh=-110.412  ysh=1.317
drizzle ../img09.hhh[1] final_g1 outweig=final_g1w in_mask=img09_cr.pl \
xsh=-110.887  ysh=-108.511
drizzle ../img10.hhh[1] final_g1 outweig=final_g1w in_mask=img10_cr.pl \
xsh=-1.103  ysh=-108.927
drizzle ../img11.hhh[1] final_g1 outweig=final_g1w in_mask=img11_cr.pl \
xsh=108.880  ysh=-109.438
drizzle ../img12.hhh[1] final_g1 outweig=final_g1w in_mask=img12_cr.pl \
xsh=108.999  ysh=-0.193
cd ..
```

```
cd wf2
drizzle ../img01.hhh[2] final_g2 outweig=final_g2w in_mask=img01_cr.pl \
xsh=0.0  ysh=0.0
drizzle ../img02.hhh[2] final_g2 outweig=final_g2w in_mask=img02_cr.pl \
xsh=-0.239  ysh=50.197
drizzle ../img03.hhh[2] final_g2 outweig=final_g2w in_mask=img03_cr.pl \
xsh=-50.498  ysh=49.982
drizzle ../img04.hhh[2] final_g2 outweig=final_g2w in_mask=img04_cr.pl \
xsh=-50.180  ysh=-0.257
drizzle ../img05.hhh[2] final_g2 outweig=final_g2w in_mask=img05_cr.pl \
xsh=-50.146  ysh=-50.270
drizzle ../img06.hhh[2] final_g2 outweig=final_g2w in_mask=img06_cr.pl \
xsh=0.170  ysh=-50.136
drizzle ../img07.hhh[2] final_g2 outweig=final_g2w in_mask=img07_cr.pl \
xsh=0.356  ysh=0.342
drizzle ../img08.hhh[2] final_g2 outweig=final_g2w in_mask=img08_cr.pl \
xsh=0.144  ysh=50.515
drizzle ../img09.hhh[2] final_g2 outweig=final_g2w in_mask=img09_cr.pl \
xsh=-50.100  ysh=50.276
drizzle ../img10.hhh[2] final_g2 outweig=final_g2w in_mask=img10_cr.pl \
xsh=-49.835  ysh=0.052
drizzle ../img11.hhh[2] final_g2 outweig=final_g2w in_mask=img11_cr.pl \
xsh=-49.612  ysh=-50.263
drizzle ../img12.hhh[2] final_g2 outweig=final_g2w in_mask=img12_cr.pl \
xsh=0.365  ysh=-49.864
cd ..

cd wf3
drizzle ../img01.hhh[3] final_g3 outweig=final_g3w in_mask=img01_cr.pl \
xsh=0.0  ysh=0.0
drizzle ../img02.hhh[3] final_g3 outweig=final_g3w in_mask=img02_cr.pl \
xsh=50.197  ysh=-0.032
drizzle ../img03.hhh[3] final_g3 outweig=final_g3w in_mask=img03_cr.pl \
xsh=50.254  ysh=50.227
drizzle ../img04.hhh[3] final_g3 outweig=final_g3w in_mask=img04_cr.pl \
xsh=0.014    ysh=50.181
drizzle ../img05.hhh[3] final_g3 outweig=final_g3w in_mask=img05_cr.pl \
xsh=-49.998  ysh=50.417
drizzle ../img06.hhh[3] final_g3 outweig=final_g3w in_mask=img06_cr.pl \
xsh=-50.136  ysh=0.101
drizzle ../img07.hhh[3] final_g3 outweig=final_g3w in_mask=img07_cr.pl \
xsh=0.340  ysh=-0.358
drizzle ../img08.hhh[3] final_g3 outweig=final_g3w in_mask=img08_cr.pl \
xsh=50.514  ysh=-0.417
drizzle ../img09.hhh[3] final_g3 outweig=final_g3w in_mask=img09_cr.pl \
xsh=50.547  ysh=49.828
drizzle ../img10.hhh[3] final_g3 outweig=final_g3w in_mask=img10_cr.pl \
xsh=0.322  ysh=49.834
drizzle ../img11.hhh[3] final_g3 outweig=final_g3w in_mask=img11_cr.pl \
xsh=-49.994  ysh=49.883
drizzle ../img12.hhh[3] final_g3 outweig=final_g3w in_mask=img12_cr.pl \
xsh=-49.865  ysh=-0.095
cd ..

cd wf4
drizzle ../img01.hhh[4] final_g4 outweig=final_g4w in_mask=img01_cr.pl \
xsh=0.0  ysh=0.0
drizzle ../img02.hhh[4] final_g4 outweig=final_g4w in_mask=img02_cr.pl \
```

```
xsh=-0.514  ysh=-50.195
drizzle ../img03.hhh[4] final_g4 outweig=final_g4w in_mask=img03_cr.pl \
xsh=49.743  ysh=-50.734
drizzle ../img04.hhh[4] final_g4 outweig=final_g4w in_mask=img04_cr.pl \
xsh=50.179  ysh=-0.496
drizzle ../img05.hhh[4] final_g4 outweig=final_g4w in_mask=img05_cr.pl \
xsh=50.895  ysh=49.512
drizzle ../img06.hhh[4] final_g4 outweig=final_g4w in_mask=img06_cr.pl \
xsh=0.582  ysh=50.133
drizzle ../img07.hhh[4] final_g4 outweig=final_g4w in_mask=img07_cr.pl \
xsh=-0.361  ysh=-0.336
drizzle ../img08.hhh[4] final_g4 outweig=final_g4w in_mask=img08_cr.pl \
xsh=-0.902  ysh=-50.507
drizzle ../img09.hhh[4] final_g4 outweig=final_g4w in_mask=img09_cr.pl \
xsh=49.340  ysh=-51.023
drizzle ../img10.hhh[4] final_g4 outweig=final_g4w in_mask=img10_cr.pl \
xsh=49.829  ysh=-0.800
drizzle ../img11.hhh[4] final_g4 outweig=final_g4w in_mask=img11_cr.pl \
xsh=50.361  ysh=49.513
drizzle ../img12.hhh[4] final_g4 outweig=final_g4w in_mask=img12_cr.pl \
xsh=0.384  ysh=49.864
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
```

3.2. Example #2: Crowded Stellar Field in the Large Magellanic Cloud Bar.

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), and can be downloaded from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html*. This site also has two scripts containing all commands used in this example. 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 Andrew Fruchter have indicated that reducing such data as single point dithers usually yields better results.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.).

3.2.1. A little "Housekeeping" to keep track of the Files.

Copy the calibrated images to working images, then store them in a directory 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/
```

3.2.2. Subtract the Sky from the Images.

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

Display the image histogram for "img1.hhh".

```
unlearn imhist
imhist.z1=-50
imhist.z2=4096
imhist.nbins=4096
imhist.autoscale=no
imhist.top_closed=yes
imhist.listout=yes
imhist.logy=no
imhist img1.hhh | sgraph wl=-5 wr=15
```

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.

3.2.3. 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.

3.2.4. 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.

```
fourier
```

Note: *cdriz* is 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 img?_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:

```
fitting

unlearn shiftfind
shiftfind.xcenter = INDEF
shiftfind.ycenter = 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.

3.2.5. 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 temp
!mv *obj*.hh? cross1x* temp/
```

3.2.6. 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/cdbs3/uref/*

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"
```

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

```
!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 edges to zero.

```
unlearn imreplace

cd pc
imrep static.hhh[1:50,1:800] 0     # zero out a pyramid edge
imrep static.hhh[1:800,1:50] 0     # zero out a pyramid edge
imrep static.hhh[799:800,1:800] 0  # zero out an edge
imrep static.hhh[1:800,799:800] 0  # zero out an edge
cd ..

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 ..

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 ..

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 ..
```

3.2.7. Make Cosmic Ray Masks.

3.2.7.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
drizzle ../img1.hhh[1] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0    ysh=0.0
drizzle ../img2.hhh[1] img2_dr.hhh outweig=img2_drw.hhh xsh=-0.020 ysh=-0.005
drizzle ../img3.hhh[1] img3_dr.hhh outweig=img3_drw.hhh xsh=-5.447 ysh=9.292
drizzle ../img4.hhh[1] img4_dr.hhh outweig=img4_drw.hhh xsh=-5.463 ysh=9.362
drizzle ../img5.hhh[1] img5_dr.hhh outweig=img5_drw.hhh xsh=3.886  ysh=15.005
drizzle ../img6.hhh[1] img6_dr.hhh outweig=img6_drw.hhh xsh=3.903  ysh=15.039
drizzle ../img7.hhh[1] img7_dr.hhh outweig=img7_drw.hhh xsh=9.144  ysh=5.845
drizzle ../img8.hhh[1] img8_dr.hhh outweig=img8_drw.hhh xsh=9.208  ysh=5.782
cd ..

cd wf2
drizzle ../img1.hhh[2] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0    ysh=0.0
drizzle ../img2.hhh[2] img2_dr.hhh outweig=img2_drw.hhh xsh=-0.002 ysh=0.009
drizzle ../img3.hhh[2] img3_dr.hhh outweig=img3_drw.hhh xsh=4.228  ysh=2.530
drizzle ../img4.hhh[2] img4_dr.hhh outweig=img4_drw.hhh xsh=4.260  ysh=2.538
drizzle ../img5.hhh[2] img5_dr.hhh outweig=img5_drw.hhh xsh=6.880  ysh=-1.715
drizzle ../img6.hhh[2] img6_dr.hhh outweig=img6_drw.hhh xsh=6.896  ysh=-1.723
drizzle ../img7.hhh[2] img7_dr.hhh outweig=img7_drw.hhh xsh=2.712  ysh=-4.159
drizzle ../img8.hhh[2] img8_dr.hhh outweig=img8_drw.hhh xsh=2.683  ysh=-4.188
cd ..

cd wf3
```

```
drizzle ../img1.hhh[3] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0    ysh=0.0
drizzle ../img2.hhh[3] img2_dr.hhh outweig=img2_drw.hhh xsh=0.009  ysh=0.002
drizzle ../img3.hhh[3] img3_dr.hhh outweig=img3_drw.hhh xsh=2.508  ysh=-4.242
drizzle ../img4.hhh[3] img4_dr.hhh outweig=img4_drw.hhh xsh=2.515  ysh=-4.274
drizzle ../img5.hhh[3] img5_dr.hhh outweig=img5_drw.hhh xsh=-1.752 ysh=-6.871
drizzle ../img6.hhh[3] img6_dr.hhh outweig=img6_drw.hhh xsh=-1.760 ysh=-6.887
drizzle ../img7.hhh[3] img7_dr.hhh outweig=img7_drw.hhh xsh=-4.174 ysh=-2.689
drizzle ../img8.hhh[3] img8_dr.hhh outweig=img8_drw.hhh xsh=-4.203 ysh=-2.661
cd ..

cd wf4
drizzle ../img1.hhh[4] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0    ysh=0.0
drizzle ../img2.hhh[4] img2_dr.hhh outweig=img2_drw.hhh xsh=0.002  ysh=-0.009
drizzle ../img3.hhh[4] img3_dr.hhh outweig=img3_drw.hhh xsh=-4.266 ysh=-2.467
drizzle ../img4.hhh[4] img4_dr.hhh outweig=img4_drw.hhh xsh=-4.298 ysh=-2.474
drizzle ../img5.hhh[4] img5_dr.hhh outweig=img5_drw.hhh xsh=-6.854 ysh=1.818
drizzle ../img6.hhh[4] img6_dr.hhh outweig=img6_drw.hhh xsh=-6.869 ysh=1.826
drizzle ../img7.hhh[4] img7_dr.hhh outweig=img7_drw.hhh xsh=-2.649 ysh=4.199
drizzle ../img8.hhh[4] img8_dr.hhh outweig=img8_drw.hhh xsh=-2.620 ysh=4.228
cd ..
```

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

3.2.7.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 *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf2
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf3
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf4
!ls *dr.hhh > dr.list
```

```
!ls *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 ..

cd wf2
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf3
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf4
imcombine *dr.hhh dr_med.hhh
cd ..
```

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

3.2.7.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.

```
unlearn blot
blot.scale = 0.5
```

```
blot.outnx = 800
blot.outny = 800
blot.expout = "200"  # exposure time for each input image
blot.coeffs = "header"
blot.rot = 0.

cd pc
blot dr_med img1_bl.hhh xsh=0.0    ysh=0.0
blot dr_med img2_bl.hhh xsh=-0.020 ysh=-0.005
blot dr_med img3_bl.hhh xsh=-5.447 ysh=9.292
blot dr_med img4_bl.hhh xsh=-5.463 ysh=9.362
blot dr_med img5_bl.hhh xsh=3.886  ysh=15.005
blot dr_med img6_bl.hhh xsh=3.903  ysh=15.039
blot dr_med img7_bl.hhh xsh=9.144  ysh=5.845
blot dr_med img8_bl.hhh xsh=9.208  ysh=5.782
cd ..

cd wf2
blot dr_med img1_bl.hhh xsh=0.0    ysh=0.0
blot dr_med img2_bl.hhh xsh=-0.002 ysh=0.009
blot dr_med img3_bl.hhh xsh=4.228  ysh=2.530
blot dr_med img4_bl.hhh xsh=4.260  ysh=2.538
blot dr_med img5_bl.hhh xsh=6.880  ysh=-1.715
blot dr_med img6_bl.hhh xsh=6.896  ysh=-1.723
blot dr_med img7_bl.hhh xsh=2.712  ysh=-4.159
blot dr_med img8_bl.hhh xsh=2.683  ysh=-4.188
cd ..

cd wf3
blot dr_med img1_bl.hhh xsh=0.0    ysh=0.0
blot dr_med img2_bl.hhh xsh=0.009  ysh=0.002
blot dr_med img3_bl.hhh xsh=2.508  ysh=-4.242
blot dr_med img4_bl.hhh xsh=2.515  ysh=-4.274
blot dr_med img5_bl.hhh xsh=-1.752 ysh=-6.871
blot dr_med img6_bl.hhh xsh=-1.760 ysh=-6.887
blot dr_med img7_bl.hhh xsh=-4.174 ysh=-2.689
blot dr_med img8_bl.hhh xsh=-4.203 ysh=-2.661
cd ..

cd wf4
blot dr_med img1_bl.hhh xsh=0.0    ysh=0.0
blot dr_med img2_bl.hhh xsh=0.002  ysh=-0.009
blot dr_med img3_bl.hhh xsh=-4.266 ysh=-2.467
blot dr_med img4_bl.hhh xsh=-4.298 ysh=-2.474
blot dr_med img5_bl.hhh xsh=-6.854 ysh=1.818
blot dr_med img6_bl.hhh xsh=-6.869 ysh=1.826
blot dr_med img7_bl.hhh xsh=-2.649 ysh=4.199
blot dr_med img8_bl.hhh xsh=-2.620 ysh=4.228
cd ..
```

The output blotted images are named "img*_bl.hhh".

3.2.7.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 *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf2
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf3
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf4
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..
```

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

3.2.7.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 ..

cd wf2
driz_cr ../img*.hhh 2
cd ..

cd wf3
driz_cr ../img*.hhh 3
cd ..

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”.

3.2.8. Create a "Master" Mask for each Image.

For each chip, the static mask file created in Section 3.2.6 is multiplied with each cosmic ray mask file created in Section 3.2.7 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 ..

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 ..

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 ..

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 ..
```

3.2.9. Construct the Final Drizzled Images.

Each input image is drizzled, applying the shifts and the mask files created in Section 3.2.8.

```
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
drizzle ../img1.hhh[1] final_g1 outweig=final_g1w in_mask=img1_cr.pl \
xsh=0.0 ysh=0.0
drizzle ../img2.hhh[1] final_g1 outweig=final_g1w in_mask=img2_cr.pl \
xsh=-0.020 ysh=-0.005
drizzle ../img3.hhh[1] final_g1 outweig=final_g1w in_mask=img3_cr.pl \
xsh=-5.447 ysh=9.292
drizzle ../img4.hhh[1] final_g1 outweig=final_g1w in_mask=img4_cr.pl \
xsh=-5.463 ysh=9.362
drizzle ../img5.hhh[1] final_g1 outweig=final_g1w in_mask=img5_cr.pl \
xsh=3.886  ysh=15.005
drizzle ../img6.hhh[1] final_g1 outweig=final_g1w in_mask=img6_cr.pl \
xsh=3.903  ysh=15.039
drizzle ../img7.hhh[1] final_g1 outweig=final_g1w in_mask=img7_cr.pl \
xsh=9.144  ysh=5.845
drizzle ../img8.hhh[1] final_g1 outweig=final_g1w in_mask=img8_cr.pl \
xsh=9.208  ysh=5.782
cd ..

cd wf2
drizzle ../img1.hhh[2] final_g2 outweig=final_g2w in_mask=img1_cr.pl \
xsh=0.0 ysh=0.0
drizzle ../img2.hhh[2] final_g2 outweig=final_g2w in_mask=img2_cr.pl \
xsh=-0.002 ysh=0.009
drizzle ../img3.hhh[2] final_g2 outweig=final_g2w in_mask=img3_cr.pl \
xsh=4.228  ysh=2.530
drizzle ../img4.hhh[2] final_g2 outweig=final_g2w in_mask=img4_cr.pl \
xsh=4.260  ysh=2.538
drizzle ../img5.hhh[2] final_g2 outweig=final_g2w in_mask=img5_cr.pl \
xsh=6.880  ysh=-1.715
drizzle ../img6.hhh[2] final_g2 outweig=final_g2w in_mask=img6_cr.pl \
xsh=6.896  ysh=-1.723
drizzle ../img7.hhh[2] final_g2 outweig=final_g2w in_mask=img7_cr.pl \
xsh=2.712  ysh=-4.159
drizzle ../img8.hhh[2] final_g2 outweig=final_g2w in_mask=img8_cr.pl \
```

```
xsh=2.683  ysh=-4.188
cd ..

cd wf3
drizzle ../img1.hhh[3] final_g3 outweig=final_g3w in_mask=img1_cr.pl \
xsh=0.0 ysh=0.0
drizzle ../img2.hhh[3] final_g3 outweig=final_g3w in_mask=img2_cr.pl \
xsh=0.009  ysh=0.002
drizzle ../img3.hhh[3] final_g3 outweig=final_g3w in_mask=img3_cr.pl \
xsh=2.508  ysh=-4.242
drizzle ../img4.hhh[3] final_g3 outweig=final_g3w in_mask=img4_cr.pl \
xsh=2.515  ysh=-4.274
drizzle ../img5.hhh[3] final_g3 outweig=final_g3w in_mask=img5_cr.pl \
xsh=-1.752 ysh=-6.871
drizzle ../img6.hhh[3] final_g3 outweig=final_g3w in_mask=img6_cr.pl \
xsh=-1.760 ysh=-6.887
drizzle ../img7.hhh[3] final_g3 outweig=final_g3w in_mask=img7_cr.pl \
xsh=-4.174 ysh=-2.689
drizzle ../img8.hhh[3] final_g3 outweig=final_g3w in_mask=img8_cr.pl \
xsh=-4.203 ysh=-2.661
cd ..

cd wf4
drizzle ../img1.hhh[4] final_g4 outweig=final_g4w in_mask=img1_cr.pl \
xsh=0.0    ysh=0.0
drizzle ../img2.hhh[4] final_g4 outweig=final_g4w in_mask=img2_cr.pl \
xsh=0.002  ysh=-0.009
drizzle ../img3.hhh[4] final_g4 outweig=final_g4w in_mask=img3_cr.pl \
xsh=-4.266 ysh=-2.467
drizzle ../img4.hhh[4] final_g4 outweig=final_g4w in_mask=img4_cr.pl \
xsh=-4.298 ysh=-2.474
drizzle ../img5.hhh[4] final_g4 outweig=final_g4w in_mask=img5_cr.pl \
xsh=-6.854 ysh=1.818
drizzle ../img6.hhh[4] final_g4 outweig=final_g4w in_mask=img6_cr.pl \
xsh=-6.869 ysh=1.826
drizzle ../img7.hhh[4] final_g4 outweig=final_g4w in_mask=img7_cr.pl \
xsh=-2.649 ysh=4.199
drizzle ../img8.hhh[4] final_g4 outweig=final_g4w in_mask=img8_cr.pl \
xsh=-2.620 ysh=4.228
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
```

3.3. Example #3: Edge-on Galaxy Nucleus 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), and can be downloaded from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html*. This site also has two scripts containing all commands used in this example.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.).

3.3.1. A little Organization to keep track of Files.

Copy the original images to a working image name, then store them in a directory 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_imgs
!mv u31* orig_imgs
```

3.3.2. Subtract the Sky from the Images.

This step is omitted. Please see the note in section 3.3.5 for more information.

3.3.3. 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 = 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.

3.3.4. 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.

```
fourier
```

Note: *cdriz* is 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 img?_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:

```
fitting

unlearn shiftfind
shiftfind.xcenter = INDEF
shiftfind.ycenter = 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 to "shifts.txt", a text file containing the shifts for each group in each image.

3.3.5. 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 temp
!mv *obj*.hh? cross1x* temp/
```

Note: Determine if the Count Level is the same for all Images.

In the other examples, the *sky* task was used to determine a background level that was subtracted from each image. This was 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 this situation, determining the background is almost impossible because of the nebulosity in each chip. Therefore, we ran 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, statistics were done on those images. The second and third WF2 images were shifted to align with the first image, so that statistics could be easily obtained in the same place for all three images.

```
!mkdir temp_dir
unlearn imshift

imcopy img1.hhh[2] temp_dir/img1_g2.hhh
imcopy img2.hhh[2] temp_dir/img2_g2.hhh
imcopy img3.hhh[2] temp_dir/img3_g2.hhh

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

```
unlearn imstat
imstat *s.hhh[369:468,730:779] fields=”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] fields="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 were generally identical in all three images, and it was therefore not necessary to scale them.

A little more housekeeping chores:

```
cd ..
!rm temp_dir/*
!rmdir temp_dir
```

3.3.6. Create a Static Pixel Mask File.

For this example, this step will be 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.

3.3.7. Make Cosmic Ray Masks.

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

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

3.3.7.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=""          # note, no static mask file used here
drizzle.rot=0.0

cd pc
drizzle ../img1.hhh[1] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0000  ysh=0.0000
```

```
drizzle ../img2.hhh[1] img2_dr.hhh outweig=img2_drw.hhh xsh=-10.878 ysh=10.971
drizzle ../img3.hhh[1] img3_dr.hhh outweig=img3_drw.hhh xsh=10.861  ysh=-11.081
cd ..

cd wf2
drizzle ../img1.hhh[2] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[2] img2_dr.hhh outweig=img2_drw.hhh xsh=4.974  ysh=5.022
drizzle ../img3.hhh[2] img3_dr.hhh outweig=img3_drw.hhh xsh=-5.024 ysh=-5.015
cd ..

cd wf3
drizzle ../img1.hhh[3] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[3] img2_dr.hhh outweig=img2_drw.hhh xsh=4.995  ysh=-5.001
drizzle ../img3.hhh[3] img3_dr.hhh outweig=img3_drw.hhh xsh=-4.987 ysh=5.051
cd ..

cd wf4
drizzle ../img1.hhh[4] img1_dr.hhh outweig=img1_drw.hhh xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[4] img2_dr.hhh outweig=img2_drw.hhh xsh=-5.049 ysh=-4.947
drizzle ../img3.hhh[4] img3_dr.hhh outweig=img3_drw.hhh xsh=5.099  ysh=4.939
cd ..
```

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

3.3.7.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 *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf2
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf3
!ls *dr.hhh > dr.list
!ls *drw.hhh > drw.list
mask_head @dr.list @drw.list invert=no
cd ..

cd wf4
!ls *dr.hhh > dr.list
!ls *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 ..

cd wf2
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf3
imcombine *dr.hhh dr_med.hhh
cd ..

cd wf4
imcombine *dr.hhh dr_med.hhh
cd ..
```

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

3.3.7.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.coeffs = "header"
blot.rot = 0.
blot.expout = 160  # exposure time of each input image.

cd pc
blot dr_med.hhh img1_bl.hhh xsh=0.0000  ysh=0.0000
blot dr_med.hhh img2_bl.hhh xsh=-10.878 ysh=10.971
blot dr_med.hhh img3_bl.hhh xsh=10.861  ysh=-11.081
cd ..

cd wf2
blot dr_med.hhh img1_bl.hhh xsh=0.0000  ysh=0.0000
blot dr_med.hhh img2_bl.hhh xsh=4.974   ysh=5.022
blot dr_med.hhh img3_bl.hhh xsh=-5.024  ysh=-5.015
cd ..

cd wf3
blot dr_med.hhh img1_bl.hhh xsh=0.0000  ysh=0.0000
blot dr_med.hhh img2_bl.hhh xsh=4.995   ysh=-5.001
blot dr_med.hhh img3_bl.hhh xsh=-4.987  ysh=5.051
cd ..

cd wf4
blot dr_med.hhh img1_bl.hhh xsh=0.0000  ysh=0.0000
blot dr_med.hhh img2_bl.hhh xsh=-5.049  ysh=-4.947
blot dr_med.hhh img3_bl.hhh xsh=5.099  ysh=4.939
cd ..
```

The output blotted images are named "img*_bl.hhh".

3.3.7.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 *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf2
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..

cd wf3
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..
```

```
cd wf4
!ls *bl.hhh > blot.list
deriv @blot.list
cd ..
```

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

3.3.7.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 ..

cd wf2
driz_cr ../img*.hhh 2
cd ..

cd wf3
driz_cr ../img*.hhh 3
cd ..

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".

3.3.8. Create "Master" Mask Files.

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

3.3.9. Construct the Final Drizzled Image.

Each input image is drizzled, applying the shifts and the mask files created in Section 3.3.7.

```
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
drizzle ../img1.hhh[1] final.hhh outweig=final_w.hhh in_mask=img1_cr.pl \
xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[1] final.hhh outweig=final_w.hhh in_mask=img2_cr.pl \
xsh=-10.878 ysh=10.971
drizzle ../img3.hhh[1] final.hhh outweig=final_w.hhh in_mask=img3_cr.pl \
xsh=10.861 ysh=-11.081
cd ..

cd wf2
drizzle ../img1.hhh[2] final.hhh outweig=final_w.hhh in_mask=img1_cr.pl \
xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[2] final.hhh outweig=final_w.hhh in_mask=img2_cr.pl \
xsh=4.974  ysh=5.022
drizzle ../img3.hhh[2] final.hhh outweig=final_w.hhh in_mask=img3_cr.pl \
xsh=-5.024 ysh=-5.015
cd ..

cd wf3
drizzle ../img1.hhh[3] final.hhh outweig=final_w.hhh in_mask=img1_cr.pl \
xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[3] final.hhh outweig=final_w.hhh in_mask=img2_cr.pl \
xsh=4.995  ysh=-5.001
drizzle ../img3.hhh[3] final.hhh outweig=final_w.hhh in_mask=img3_cr.pl \
xsh=-4.987 ysh=5.051
cd ..

cd wf4
drizzle ../img1.hhh[4] final.hhh outweig=final_w.hhh in_mask=img1_cr.pl \
xsh=0.0000 ysh=0.0000
drizzle ../img2.hhh[4] final.hhh outweig=final_w.hhh in_mask=img2_cr.pl \
xsh=-5.049 ysh=-4.947
drizzle ../img3.hhh[4] final.hhh outweig=final_w.hhh in_mask=img3_cr.pl \
xsh=5.099  ysh=4.939
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
```

3..4. Example #4: 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.

Note: For this example, we will be recalibrating the images with the best available dark reference file. This will allow us to obtain information about warm pixels in the image and flag them in the static pixel mask file. Therefore, before proceeding, recalibrate the images using the best available reference files. (For more information on recalibration, please refer to the "HST Data Handbook" or "WFPC2 Data Analysis: A Tutorial" available at *http://www.stsci.edu/ftp/instrument_news/WFPC2/wfpc2_advisory.html* . If you prefer not to recalibrate the images yourself, you may retrieve the data for working through this example from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html*. This site also has two scripts containing all commands used in this example.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.).

3.4.1. A little "Housekeeping" to keep track of the Files.

Copy calibrated images (PC only) to working images, then store them in a directory 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_dir
!mv u3*.c0? orig_dir/
```

3.4.2. Subtract the Sky from the Images.

Before proceeding, the value for the *sky.width* parameter should be determined, using the task *imhistogram*. Run *imhist* on all the images, to get the sky width.

```
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
imhist img2.hhh | sgraph wl=-10 wr=10
imhist img3.hhh | sgraph wl=-10 wr=10
imhist img4.hhh | sgraph wl=-10 wr=10
imhist img5.hhh | sgraph wl=-10 wr=10
```

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 in the "BACKGRND" header keyword.

3.4.3. 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.

3.4.4. 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.

```
fourier
```

Create an input list and output list.

```
print("img1_obj.hhh", >> "inlist")
print("img2_obj.hhh", >> "inlist")
print("img3_obj.hhh", >> "inlist")
print("img4_obj.hhh", >> "inlist")
print("img5_obj.hhh", >> "inlist")

print("cross1x1.hhh", >> "outlist")
print("cross1x2.hhh", >> "outlist")
print("cross1x3.hhh", >> "outlist")
print("cross1x4.hhh", >> "outlist")
print("cross1x5.hhh", >> "outlist")

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 @inlist img1_obj.hhh @outlist
```

where "img1_obj.hhh" is the reference image.

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.xcenter = INDEF
shiftfind.ycenter = INDEF
shiftfind.boxsize = INDEF
shiftfind.fwhm = 7
shiftfind.ellip = 0.05
shiftfind.pa = 45
shiftfind.fitbox = 7
shiftfind.kellip = yes
shiftfind @outlist shifts.txt
```

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

3.4.5. Determine the Average Shifts

This step is omitted because only the PC images are being processed.

3.4.6. 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.

<u>Note:</u> 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 u3lr0205r.c1? orig_dir/
```

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"
```

3.4.7. Make Cosmic Ray Masks.

3.4.7.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

drizzle img1.hhh img1_dr.hhh outweig=img1_drw.hhh  xsh=0.0000 ysh=0.0000
drizzle img2.hhh img2_dr.hhh outweig=img2_drw.hhh  xsh=11.008 ysh=5.4022
drizzle img3.hhh img3_dr.hhh outweig=img3_drw.hhh  xsh=16.476 ysh=16.371
drizzle img4.hhh img4_dr.hhh outweig=img4_drw.hhh  xsh=5.5907 ysh=11.009
drizzle img5.hhh img5_dr.hhh outweig=img5_drw.hhh  xsh=0.0974 ysh=0.1534
```

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

3.4.7.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.

```
unlearn mask_head
!ls *dr.hhh > dr.list
!ls *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 = 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.

3.4.7.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.

```
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 expout=120 xsh=0.0000 ysh=0.0000
blot dr_med.hhh img2_bl.hhh expout=120 xsh=11.008 ysh=5.4022
blot dr_med.hhh img3_bl.hhh expout=120 xsh=16.476 ysh=16.371
blot dr_med.hhh img4_bl.hhh expout=120 xsh=5.5907 ysh=11.009
```

```
blot dr_med.hhh img5_bl.hhh expout=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”.

3.4.7.d. Create Derivative Images.

This step estimates effects caused by blurring in the medianed 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”.

3.4.7.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”.

3.4.8. Create a "Master" Mask for each Image.

The static mask file created in Section 3.4.6 is multiplied with each cosmic ray mask file created in Section 3.4.7to 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
```

3.4.9. Construct the Final Drizzled Image.

Each input image is drizzled, applying the shifts and the mask files created in Section 3.4.8.

```
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.
drizzle img1.hhh final.hhh outweig=final_w.hhh in_mask=img1_cr.pl \
xsh=0.0000 ysh=0.0000
drizzle img2.hhh final.hhh outweig=final_w.hhh in_mask=img2_cr.pl \
xsh=11.008 ysh=5.4022
drizzle img3.hhh final.hhh outweig=final_w.hhh in_mask=img3_cr.pl \
xsh=16.476 ysh=16.371
drizzle img4.hhh final.hhh outweig=final_w.hhh in_mask=img4_cr.pl \
xsh=5.5907 ysh=11.009
drizzle img5.hhh final.hhh outweig=final_w.hhh in_mask=img5_cr.pl \
xsh=0.0974 ysh=0.1534
```

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
```

3.5. Example #5: STIS/CCD Images of the HDF-N Field.

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. (<u>Note:</u> this proposal uses a base32 numbering scheme for visits after 100--these values have been converted to numbers in the LINENUM header keyword.)

Because of a processing anomaly in the pipeline at the time these observations were taken, these images had to be recalibrated. If you wish to try the example in this section, you may download it from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html* instead of retrieving and recalibrating the images from scratch. This site also has two scripts containing all commands used in this example.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.).

3.5.1. A little Organization to keep track of Files.

Some of the tasks in *dither* and *ditherII* call *imcalc*. Unfortunately, in the current version of IRAF and STSDAS (versions 2.11.1 and 2.0.2 respectively), *imcalc* cannot operate on FITS files. This will be fixed in the next IRAF release; for now, please convert the FITS images to the GEIS format.

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      # prop 7781, visit 7w, exp 10
imcopy o46p7y010_crj.fits[dq]  img01_m.hhh

imcopy o46p81010_crj.fits[sci] img02.hhh      # prop 7781, visit 81, exp 10
imcopy o46p81010_crj.fits[dq]  img02_m.hhh
imcopy o46p82010_crj.fits[sci] img03.hhh      # prop 7781, visit 82, exp 10
imcopy o46p82010_crj.fits[dq]  img03_m.hhh

imcopy o46p83010_crj.fits[sci] img04.hhh      # prop 7781, visit 83, exp 10
imcopy o46p83010_crj.fits[dq]  img04_m.hhh
imcopy o46p84010_crj.fits[sci] img05.hhh      # prop 7781, visit 84, exp 10
imcopy o46p84010_crj.fits[dq]  img05_m.hhh

imcopy o46p8c010_crj.fits[sci] img06.hhh     # prop 7781, visit 8a, exp 10
imcopy o46p8c010_crj.fits[dq]  img06_m.hhh
imcopy o46p8d010_crj.fits[sci] img07.hhh      # prop 7781, visit 8b, exp 10
imcopy o46p8d010_crj.fits[dq]  img07_m.hhh
```

```
imcopy o46p8e010_crj.fits[sci] img08.hhh      # prop 7781, visit 8c, exp 10
imcopy o46p8e010_crj.fits[dq]  img08_m.hhh
imcopy o46p8f010_crj.fits[sci] img09.hhh      # prop 7781, visit 8d, exp 10
imcopy o46p8f010_crj.fits[dq]  img09_m.hhh

imcopy o46p8g010_crj.fits[sci] img10.hhh      # prop 7781, visit 8e, exp 10
imcopy o46p8g010_crj.fits[dq]  img10_m.hhh
imcopy o46p8h010_crj.fits[sci] img11.hhh      # prop 7781, visit 8f, exp 10
imcopy o46p8h010_crj.fits[dq]  img11_m.hhh

imcopy o46p8k010_crj.fits[sci] img12.hhh      # prop 7781, visit 8g, exp 10
imcopy o46p8k010_crj.fits[dq]  img12_m.hhh
imcopy o46p8l010_crj.fits[sci] img13.hhh      # prop 7781, visit 8h, exp 10
imcopy o46p8l010_crj.fits[dq]  img13_m.hhh

imcopy o46p8m010_crj.fits[sci] img14.hhh      # prop 7781, visit 8i, exp 10
imcopy o46p8m010_crj.fits[dq]  img14_m.hhh
imcopy o46p8n010_crj.fits[sci] img15.hhh     # prop 7781, visit 8j, exp 10
imcopy o46p8n010_crj.fits[dq]  img15_m.hhh

imcopy o46p8o010_crj.fits[sci] img16.hhh      # prop 7781, visit 8k, exp 10
imcopy o46p8o010_crj.fits[dq]  img16_m.hhh
imcopy o46p8p010_crj.fits[sci] img17.hhh      # prop 7781, visit 8l, exp 10
imcopy o46p8p010_crj.fits[dq]  img17_m.hhh

imcopy o46p97010_crj.fits[sci] img18.hhh      # prop 7781, visit 97, exp 10
imcopy o46p97010_crj.fits[dq]  img18_m.hhh
```

Store original images for safe-keeping.

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

Move mask files to a separate working directory.

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

You will also need the the STIS geometric distortion correction coefficients file in your working directory. This file is part of the *dither* package in STSDAS and can be found in the directory where the package is installed.
(Example: <your STSDAS directory location>/stsdas/pkg/analysis/dither/drizzle/coeffs/)

```
copy drizzle$coeffs/stis-ccd .
```

3.5.2. 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 "BACK-GRND" header keyword.

3.5.3. Prepare the Images for Cross-correlation.

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

3.5.4. 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.

```
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"  # the STIS/CCD distortion coefficients file
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.xcenter = INDEF
shiftfind.ycenter = 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*
```

3.5.5 Determine the Average Shifts.

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

3.5.6. 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 the WFPC2 example in Section 3.4. However, if your image has 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 ..
```

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

3.5.7.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 d*rizzle.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.

3.5.7.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 *_dr.hhh > dr.list
ls *_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".

3.5.7.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"  # exposure time of the original input images
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”.

3.5.7.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”.

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

```
unlearn driz_cr
driz_cr.gain = 1.    # the A-TO-D gain of the image (keyword: ATODGAIN)
driz_cr.rn = 4.      # read-noise of the image (keyword: READNSE)
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.

3.5.8. Create a "Master" Mask for each Image.

The static mask file created in Section 3.5.6 is multiplied with each bad pixel mask file created in Section 3.5.7. 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 ..
```

3.5.9. Construct the Final Drizzled Image.

Each input image is drizzled, applying the shifts and the mask files created in Section 3.5.8.

```
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 2400x2400 pixels of the drizzled image.

```
set stdimage = imt3200
```

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.

3.6. Example #6: NICMOS Camera 2 Images of the 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.

You will need the geometric distortion correction coefficients file for drizzling; before proceeding any further, please request this file from the NICMOS group by sending e-mail to *help@stsci.edu*. Be sure to specify which camera you're using. Or you can download the images and the distortion correction coefficients file from *http://www.stsci.edu/ftp/instrument_news/WFPC2/Wfpc2_driz/wfpc2_driz.html*. This site also has two scripts containing all commands used in this example.

Before running this example, please make sure that you have loaded the packages and run the commands listed on pages 16 and 17 (Section 3.0. A-E.).

3.6.1. A little Organization to keep track of Files.

Some of the tasks in *dither* and *ditherII* call *imcalc*. Unfortunately, in the current version of IRAF and STSDAS (versions 2.11.1 and 2.0.2 respectively), *imcalc* cannot operate on FITS files. This will be fixed in the next IRAF release; for now, please convert the FITS images to the GEIS format.

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. (Note: a future version of *driz_cr* will be able to work with countrates.)

```
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
```

3.6.2. 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 see *http://www.stsci.edu/ftp/instrument_news/NICMOS/NICMOS_updates/ped_10August98.ascii* Please also refer to the NICMOS image anomaly page at *http://www.stsci.edu/ftp/instrument_news/NICMOS/nicmos_anomalies.html* for more information.

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.

3.6.3. Prepare the Images for Cross-correlation.

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

3.6.4. 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.

```
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.xcenter = INDEF
shiftfind.ycenter = 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*
```

3.6.5 Determine the Average Shifts.

This step is omitted because the NIC2 images in this example are single-group images.

3.6.6. 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
```

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

3.6.7.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.

3.6.7.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 img?_dr.hhh > dr.list
!ls 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”.

3.6.7.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  # exposure 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”.

3.6.7.d. Create Derivative Images.

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

```
!ls *bl.hhh > blot.list
deriv @blot.list
```

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

3.6.7.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.

3.6.8. Create a "Master" Mask for each Image.

The static mask file in Section 3.6.6 is multiplied with each bad pixel mask file created in Section 3.6.7. 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
```

3.6.9. Construct the Final Drizzled Image.

Each input image is drizzled onto a single output image, applying the shifts and the mask files created in Section 3.6.8.

```
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
```

4.0 Troubleshooting.

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

A. 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 time. *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.

B. 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.

C. Check the *imcombine* parameters. If your input images have unequal exposure time, and you're using the parameter *imcombine.reject=minmax*, make sure that the *imcombine.scale* parameter is set to "*exposure*." Otherwise, the median image will be incorrectly scaled.

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

A. 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.

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

A. Check the *blot* parameters, especially the *blot.expout* parameter. See Section 1A.

B. 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. (<u>Note:</u> 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.)

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

A. 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.

5. *Why does my final drizzled image look so noisy?*

A. Check the *blot* parameters--were they properly set? See Section 1A.

B. 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 i*mstat.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 Data Analysis Help Desk (*help@stsci.edu*) for assistance in determining the reason for the unequal background.

6. *Why can't I run* ***mask_head****,* ***deriv****, and* ***driz_cr****?*

A. These tasks are new additions to the *dither* suite of tasks, and they are not available in the latest STSDAS release. You can download them from the Dither webpage at *http://www.stsci.edu/~fruchter/dither/#DitherII* as the *ditherII* package. Please be sure to register when your download this package; this way, Andy Fruchter will be able to notify you as changes are made to the software.

B. Be sure to load the *ditherII* package by typing "ditherII" in *iraf*.

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

A. Check to see if you've run out of disk space.

B. Check the image, it may have been corrupted.

C. It could be an IRAF or STSDAS problem. Flush the remaining processes by typing "flpr" (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, reinstall the *ditherII* package. If that does not work, run "mkiraf" and start over. If that does not work, contact your local IRAF guru or the STScI help desk.

If you encounter any other problems while using the *dither* and *ditherII* packages, please be sure to contact the STScI Help Desk (*help@stsci.edu*). If you have comments about this document, please contact Shireen Gonzaga (*shireen@stsci.edu*) or John Biretta (*biretta@stsci.edu*).

ACKNOWLEDGEMENTS:

We wish to thank Andy Fruchter, Chris Hanley, Richard Hook, Rubina Kotak, and Max Mutchler for their helpful discussions and comments.

References:

A Method for the Linear Reconstruction of Undersampled Images
A. S. Fruchter, R. N. Hook
submitted PASP

Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration
A. S. Fruchter, M. Mutchler
http://www.stsci.edu/~fruchter/dither/#DitherII

The Hubble Deep Field Webpage
A. S. Fruchter
http://www.stsci.edu/ftp/science/hdf/hdf.html

WFPC2 Instrument Handbook, version 4.0
ed. John Biretta

WFPC2 Tutorial, version 2.0
WFPC2 Group
http://www.stsci.edu/ftp/instrument_news/WFPC2/wfpc2_advisory.html