Space Telescope Science Institute, July 28, 1998

Drizzling Singly-Dithered Hubble Space Telescope Images: A Demonstration

Andrew S. Fruchter and Max Mutchler
Space Telescope Science Institute
3700 San Martin Drive, Baltimore, Maryland 21218
fruchter@stsci.edu, mutchler@stsci.edu

Abstract. The development of the IRAF/STSDAS package **dither** and its primary task, **drizzle** (Fruchter & Hook 1997), now give the HST user powerful tools for combining dithered images; however, these tasks do not provide a simple means for locating and removing cosmic rays, or other blemishes, from dithered images. Here we discuss several additional IRAF scripts that one of us (ASF) has written to simplify the removal of cosmic rays from dithered data. These procedures are particularly useful when only a single image has been taken at a given dither position. We demonstrate these tasks in a cookbook-style by combining a set of singly-dithered WFPC2 images, and give instructions for retrieving these new tasks from the WWW. These tasks are presently in development, and the tasks, and this document, should be considered a "beta" release.

1. Introduction

The Hubble Deep Field (HDF) (Williams et al. 1996) made clear the advantages that dithering can confer by allowing the user to regain much of the spatial information hidden by undersampling. While undersampling is most evident in WFPC2 data, it is, in fact, an issue that affects all HST instruments in at least some of their modes. The substantial amount of time devoted to the Hubble Deep Field meant that a number of images could be taken at each dither position, and cosmic rays removed from these stacks before the stacks were drizzled onto a common output image. For typical HST users with modest allocations of telescope time, it is not practical to take many images at each of several dither positions. Here we display the use of several new tasks designed to help users combine singly-dithered images, that is dithered data where only a single image has been taken at each dither position. These tasks will also be helpful in the case where perhaps only a couple of images have been taken at a given dither position – indeed this may be the optimal way to obtain dithered data in the presence of cosmic rays. Then the two or three images at a given dither position can be combined to provide a first-pass cosmic-ray removal before shift determination, after which *all* of the data can be used to locate blemishes and cosmic rays in each of the individual images.

This document is intended as an introduction to using **drizzle** to remove cosmic rays from dithered data. Before reading this document, one should be familiar with the concepts behind **drizzle** and preferably with some of the tasks in the IRAF **dither** package. Fruchter and Hook (1997) provides a good introduction to **drizzle**. An overview of both the **drizzle** task and the **dither** package can be found in (Fruchter et al. 1998).

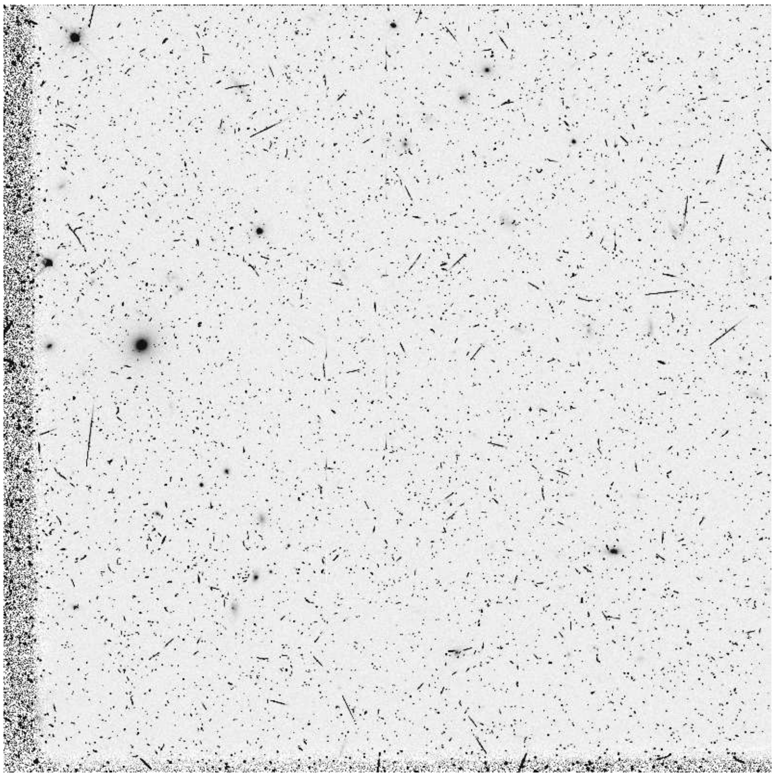

Figure 1. Group 2 from one of the 12 WFPC2 images of SA22. We will process only the WF2 chip in this example.

2. The images

Our sample dataset consists of twelve singly-dithered F814W WFPC2 images of the Hawaii Deep Survey field SSA22, from HST program 5399 with Principal Investigator Lennox Cowie (Cowie, Hu & Songaila 1995). The archival dataset rootnames are u2h90101t through 106t, and u2h90201t through 206t.

The images were obtained from the CADC HST archives (http://cadcwww.dao.nrc.ca). The CADC (and the ST-ECF, http://ecf.hq.eso.org) presently provide calibration-on-the-fly: images are recalibrated with the best currently available calibration files. This option will be available on WFPC2 data from the STScI archives later in 1998. The fits files were read into IRAF using **strfits**, and the new IRAF name "SA22_" was used, causing strfits to create the files SA22_001 to SA22_012. We generally find using such a numbering system more convenient than keeping the original dataset names.

3. Measuring shifts between dithered images

The drizzling process is sensitive to the accuracy of the shifts used, so we begin by carefully measuring the shifts between the dithered exposures.

3.1. Subtract the sky background from the input images

Although removing the sky background is not strictly necessary for determining shift offsets, it is useful for proper combination of the images later – because drizzling interlaces pixels from different images, their skies must be identical, or drizzling will create additional noise.

Furthermore, the task that we will use to do a first pass removal of cosmic rays before shift determination will need to subtract the sky if we do not do it now; therefore we might as well remove the sky before starting. We subtract skies from all of the images using the task **sky**. The value of the sky in each image is written by the task to the image header. Before running **sky**, one may wish to use **imhistogram** on an image to estimate the FWHM (sky.width) input parameter required by **sky**. Note that **dpar** lists the input parameters for a given task, in this case **sky**. Throughout this write-up, we use **dpar** to list parameters, but will sometimes not show the full output of dpar, but rather only those parameters which are particularly relevant to the problem at hand. More information on all of the tasks used in this example can be found in their IRAF help files.

```
di> dpar sky
sky.input = "SA22_*.hhh"
sky.subsky = yes
sky.width = 8
sky.stat = "mean"
sky.skyname = "BACKGRND"
sky.skyvalue = 0.

# SA22_001.hhh
# Group    Rough sky   Delta sky    Final sky
#
   1       6.212685    -0.023009    6.1896756
   2       28.89058    0.0766959    28.967276
   3       28.90152    0.0498477    28.951368
   4       28.25609    0.0426755    28.298765
```

The task **sky** obtains the sky in two steps, first using the peak in the image histogram with one unit bins to estimate a sky and then using the statistics about this peak to obtain a more accurate sky. In WFPC2 images with low sky a difference of a significant fraction of a pixel may occasionally be seen.

3.2. Preparing images for cross-correlation

Finding the peak of the cross-correlation between two images is an excellent means of determining their relative offset, provided the noise in the images is nearly Gaussian (see Fruchter *et al.* 1998 for a discussion of this point). Of course cosmic rays are highly non-Gaussian, and therefore for an accurate determination of a shift using cross-correlation, we must first remove much of the power from the cosmic rays in the images.

To do this we use the **dither** task **precor**. **precor** is well-suited to relatively sparse images (such as deep images of background fields). **precor** looks for astrophysical objects by determining regions whose median is significantly different from the sky (which should be removed previously or in **precor** itself). The size of the region and the level of significance are set by the user. **precor** then sets regions without significant power from astrophyical objects to zero. Cosmic rays on top of astrophysical objects are left untouched, however. In cases where the user is imaging a large extended object, a better approach might be to take a local median of the input images and cross-correlate these medianed images. The median will blur the image a bit, but in very extended objects much of the power in the image is on large scales and will not be harmed. In cases where a preliminary removal of the cosmic rays is particularly difficult, the user may wish to use the IRAF **immatch** task, which determines relative shifts of images based on the positions of objects in a list provided by the user. While this technique does not take full advantage of the information in an image, it nonetheless can be quite effective. Once a preliminary shift is determined, the iterative techniques described below can be used to determine a more accurate shift, should that be desired.

The cleaned images produced by **precor** are named *obj.hhh. These should be used only for creating the cross-correlation images and measuring shifts, and NOT used as the input images in later stages.

We now present the inputs to **precor** and the output of the program. Note that we set the `do_sky` parameter to "no" because we have previously removed the sky from the images and there is no reason to repeat the calculation. If the sky has not been previously removed `do_sky` should be set to "yes".

```
di> dpar precor
precor.inlist = "SA22_*.hhh"
precor.box_siz = "5"
precor.min_pix = 16
precor.min_val = 6.
precor.ngroup = 4
precor.do_sky = no
precor.sig2n = yes

Working on SA22_009.hhh...
SA22_009.hhh -> SA22_009_obj.hhh
add SA22_009_obj.hhh,ORG_TITL = SA22_009.hhh
SA22_009_obj.hhh updated
Percent done: 10 20 30 40 50 60 70 80 90 100
```

Note that many tasks in **dither** use the tmp directory for scratch. It is best to set this to a real directory, such as the present directory.

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

In Solaris, tmp is frequently assigned to memory, and therefore the operating system may attempt to store all of the intermediate images in swap space, if tmp is not redefined.

3.3. Cross-correlate the cleaned images

We next use the **offsets** task to perform the cross-correlation. **offsets** cross-correlates all of the four WFPC2 groups per image in one go. We will use the first image as our reference image, and cross-correlate against all input images. We use the output filename prefix "SA22_1x" to indicate that we have cross-correlated against the first image. Note that by using the "*obj.hhh" as the input file list, we will cross-correlate SA22_001 against itself. However, this is both a check of method and a good way to insure that the numbering of the output cross-correlation images is the same as that of the input SA22_ names.

```
di> dpar offset
offsets.input = "*obj.hhh"
offsets.refimage = "SA22_001_obj.hhh"
offsets.output = "SA22_1x"
offsets.coeffs = "header"
```

Note that the paramter `coeff` is set to header. This tells drizzle to examine the image header and then use the geometric distortion parameters appropriate for the file. In the case of shifts of many pixels, it is vital to use the geometric distortion coefficients when drizzling WFPC2 images in order to obtain the best possible alignment of the images.

If we were measuring shifts for only one of the four WFPC2 groups, the **crossdriz** task in **dither** could be used instead of **offsets**.

Finally a note on a subtle but occasionally important parameter, `pad`, which is found in both **offsets** and **crossdriz** (but is hidden in the `cdriz` parameter set in the case of **offsets**). `pad` determines whether the image is padded with zeros (and quadrupled in area) before cross-correlation. This parameter is normally set to "no" to avoid the excess cpu time. However, this means that sources near the edge can wrap around and cross-correlate with objects on the other side. If you have a complicated field, or a substantial number of remaining cosmic-rays, and the method described in the next section is finding shifts way off the mark, try turning on the padding.

3.4. Measure the shifts using the cross-correlated images

The task **shiftfind** determines the shifts between images by locating the peak in the cross-correlation images. The output is written, in this case, to the file "SA22.shifts".

```
di> dpar shiftfind
shiftfind.cclist = "SA22_1x*.hhh"
shiftfind.outfile = "SA22.shifts"
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

                     x shift          y shift
SA22_1x_02_g1.hhh  231.9379 0.1365  -83.8882 0.1006
SA22_1x_02_g2.hhh   -0.1066 0.0306   50.2241 0.0242
SA22_1x_02_g3.hhh   50.1089 0.0282   -0.0090 0.0229
SA22_1x_02_g4.hhh   -0.3521 0.0243  -50.2614 0.0203
```

Although it is known that the chips of the WFPC2 drift in relative position on the order of a pixel per year, if the images are all taken at similar times, then we would expect the shifts in all four chips to be the same. We can therefore improve on the estimate of the shifts by averaging all four chips together. To do this, we use the task **avshift**. **avshift** takes into account the small relative rotations of the chips and allows one to assign relative weights to the different chips. In the case of the SA22 data, we give group 1 (PC1) a weight of 0, and the other chips weights of 1, due to the fact that the PC has far less image power, yet a similar amount of noise (primarily due to cosmic rays) as the WF images.

```
di> dpar avshift
avshift.table = "SA22.shifts"
avshift.angle = 0.
avshift.weight = "0. 1. 1. 1."

cl> avshift SA22.shifts angle=0 > SA22.avshift
```

Below we show the output from **avshift** for one of the 12 cross-correlations. In this case the cross-correlation is with image "002" (though **avshift** only presently handles up to 99 images at a time, thus the "02"). The observed, estimated best shift based on the weighted average of all chips, and the difference between the latter two is output.

```
# Image        xsh_in   ysh_in best_xsh best_ysh tot_sh_in delta_xsh delta_ysh
SA22_1x_02_g1  231.938  -83.888 -109.726    0.434  109.727  341.664   -84.323
SA22_1x_02_g2   -0.107   50.224   -0.257   50.198   50.198    0.150     0.026
SA22_1x_02_g3   50.109   -0.009   50.198   -0.015   50.198   -0.090     0.006
SA22_1x_02_g4   -0.352  -50.261   -0.496  -50.196   50.198    0.144    -0.065
# RMS error (all chips on WF scale)):  80.4980
# RMS error (WF only):     0.1372
# weighted RMS error (all chips on WF scale)):   0.1372
# weighted RMS error (WF only):     0.1372
```

The avhsift output gives the input shifts, `xsh_in` and `ysh_in`), the best esimate of the shifts for a particular group based on the average of the four groups, `best_xsh` and `best_ysh`, the total shift found for a particular group, `tot_sh_in`, and the difference between the input and best shifts, `delta_xsh` and `delta_ysh`. In the example here, we will work only on group 2 (WF2). The best shifts found for group 2 are shown in the following table.

Group 2 Shifts in SA22 Images

Image	X	Y
1	0.00	0.00
2	-0.26	50.20
3	-50.48	50.00
4	-50.17	-0.17
5	-49.99	-50.36
6	0.16	-50.12
7	0.41	0.33
8	0.17	50.59
9	-50.09	50.34
10	-49.82	0.12
11	-49.62	-50.17
12	0.37	-49.83

3.5. Image shifts from object lists

Cross-correlation has the advantage that it does not depend on objects having a particular size or shape, and properly weights and adds the contributions of different objects. However, when the template is noisy (for instance when a single image is being used) and when the objects are all easily modelled, it may be simpler and more accurate to use a task, such as **imcentroid**, which determines the positions of individual objects on a list and uses these to determine the image offsets. This procedure is also often quite robust in the presence of cosmic rays, since it uses human intelligence to separate out true objects from blemishes.

Finally, the user should note that s/he has more than one chance to accurately determine the image shifts. Later in this article we discuss the iterative use of drizzling to improve the shift determination.

4. Image combination

Now that we have measured accurate shifts, we can begin to combine the images. The steps in image combination are:

- Create a static mask.
- Drizzle images onto separate output images, taking into account shifts and static mask.
- Obtain the median of the drizzled images.
- Blot (map) the median image back to the positions on the sky of the input images.
- Create cosmic-ray masks by comparing the original images to the blotted images.
- Drizzle input images for a final time.

We generally prefer to work with the original data in a main directory while combining each of the 4 WFPC2 image groups in separate subdirectories.

4.1. Make a static mask

WFPC2 images suffer from the presence of hot pixels. While there is a task in IRAF for the removal of these pesky pixels, a very effective and instructive means of eliminating them in cases such as this one, where one has multiply dithered images is to use the images themselves to remove the hot pixels. Although we are going to create individual masks for each of the images, we can more effectively identify low-level static defects in the images if we use all of the images, rather than separately identifying the same defects in the individual images.

First we use **gcombine** to create a median image of all four image groups. Here all four images have equal exposure times, so we do not have to scale to exposure time:

```
di> dpar gcombine
gcombine.input = "SA22_*.hhh"
gcombine.output = "med"
gcombine.groups = "*"
gcombine.combine = "median"
```

After the use of **gcombine**, the image is largely blank with the exception of hot pixels and chip defects. Had the shifts not been so large, we would have had to mask the sources in the images before taking the median, or else power from the sources would also remain. In such cases the user may wish to defer the creation of the static mask until later in the process, when we have created our first cosmic-ray free image, so that the creation of object masks is simplified.

We next use **imcalc** to create a file which will mask pixels that are significantly different from 0:

```
cl> imcalc med mask "if (abs(im1) .gt. 5) then 0 else 1"
```

and then copy the appropiate group of the mask to the new directory and set the outer bad regions of the chip to 0. Users wishing to preserve as much information as possible near the edges of the chip may wish to reduce the regions zeroed, and possibly create a weight file using the appropriate flat that reflects the lessening signa-to-noise of the image near the edge.

```
cl> imcopy mask[2] group2/mask

cl> cd group2
cl> imrep mask [1:50,1:800] 0.
cl> imrep mask [1:800,1:30] 0.
cl> imrep mask [799:800,1:800] 0.
cl> imrep mask [1:800,799:800] 0.
```

4.2. Drizzle the input images

In order to align the images on the sky, we drizzle them onto a finer 2048x2048 grid. The output drizzled images are named *p1.hhh here to indicate that they were drizzled with `pixfrac=1.0`.

```
cl> cl < rundriz1.cl
```

where rundriz1.cl contains:

```
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="mask"
drizzle  ../SA22_001.hhh[2] SA22_001_dr outweig=SA22_001_drw
xsh=0.0 ysh=0.0 rot=0.0
drizzle  ../SA22_002.hhh[2] SA22_002_dr outweig=SA22_002_drw
xsh=-0.257 ysh=50.198 rot=0.0
```

etc ...

Remember to `set stdimage=imt2048` to display the drizzled images on their new finer grid.

5. Create a median image largely cleaned of cosmic rays

In order to produce an image largely devoid of cosmic rays, we will take a modified median of the drizzled input images. However, due to the shifts and the presence of a mask, the output images do not fully cover the same areas. We must therefore be careful to only use an image in the median where it has valid data. To do this we employ the IRAF task **imcombine** (we use **imcombine** because this task is better suited than **gcombine** for processing STIS and NICMOS data, and we would like the method we are using here to be easily extended to other HST instruments).

Before running **imcombine** we use **mask_head** to convert the output drizzle weight images into 1's and 0's, and to update each image header giving the name of the associated mask, as required by **imcombine**. **mask_head** is one of the tasks available on the web in the new package **ditherII**. We expect all the tasks in **ditherII**, or their successors, to eventually be incorporated into **dither**.

```
di> dpar mask_head
mask_head.inlist = "@dr.list"
mask_head.masklist = "@drw.list"
mask_head.invert = no
```

In the above `dr.list` is a file containing a list of all the output drizzled images, and `drw.list` is similarly a list of all output weight images. In both cases a single image is listed per line. The task **mask_head** can be set to invert the masks – that is map positive values to zero and zero to one.

Next we use **imcombine** to create a median of the output images.

```
di> dpar imcombine
imcombine.input = "*dr.hhh"
imcombine.output = "dr_med"
imcombine.plfile = ""
imcombine.sigma = ""
imcombine.combine = "median"
imcombine.reject = "minmax"
imcombine.project = no
imcombine.outtype = "real"
imcombine.offsets = "none"
imcombine.maskvalue = 0.
imcombine.masktype = "badvalue"
imcombine.nlow = 1
imcombine.nhigh = 4
imcombine.nkeep = 1
```

Two parameters of particular note are `nlow` and `nhigh`. These give the number of high and low pixels rejected outright from the stack. In these images, where 5% of all pixels are affected by cosmic rays, a straight median of the images would frequently be affected by cosmic rays – especially in outlying areas where there is less overlap of images. Finally note that we have told **imcombine** the type and value of the mask using the parameters `maskvalue` and `masktype`. Had we instead used `mask_head.invert = yes` in the previous step, the bad value in imcombine would be 1 rather than 0.

6. Compare images and create a cosmic-ray mask

6.1. Blot the median image onto the input planes

We now map the median image back to the position of the inputs (including the effects of geometric distortion) using the task **blot**. These blotted images will act as estimates of cosmic-ray-free images to be compared to the input data. The output blotted images will be scaled to the input image exposure time and named *bl.hhh:

```
cl> cl<runblot.cl
```

where the file `runblot.cl` is:

```
blot.scale=0.5
blot.outnx=800
blot.outny=800
blot.expout=2400
blot.coeffs="header"
blot.rot=0.0
blot  dr_med SA22_001_bl xsh=0.000   ysh=0.000
blot  dr_med SA22_002_bl xsh=-0.257  ysh=50.198
blot  dr_med SA22_003_bl xsh=-50.481 ysh=49.998
blot  dr_med SA22_004_bl xsh=-50.171 ysh=-0.169
```

etc....

In this case all input images had the same exposure time, so we could set `blot.expout` in a line before the blot commands. In general, however, different exposures will have different exposure times and `expout` should then be included as a parameter in the individual blot calls, in the fashion of `xsh` and `ysh`, for instance.

6.2. Take the derivative of the blotted images

When, in the next step, we directly compare the blotted images with the original data sets, we must take into account the blurring of the image caused by the median process as well as by possible errors in the input shifts. To help estimate these errors, we take the absolute value of the difference of each pixel with each of its four major neighbors, and save the largest value associated with each pixel. The task that performs this is called **deriv**. The only input for this task is the list of blotted images. The output images will have names of the form `*_bl_deriv`.

7. Combine cosmic-ray masks and remove bad pixels

We now use a new task from **ditherII**, **driz_cr**, to compare the original images to their blotted counterparts. Any pixels that show a significant difference are flagged as bad pixels in the mask it creates. This task iterates to find bad pixels adjacent to others it has already found, since most cosmic-ray hits involve many adjacent pixels.

This task has a number of user-controlled parameters. The user must first give the group of the original image, since this task allows one to compare the estimated image with the original data in the four-group WFPC2 data file. In order to estimate the signal-to-noise ratio (SNR) of a given pixel, the task requires the image gain and read noise (in electrons). The user must also supply the SNR required before an outlier is masked. Two SNRs are actually needed, one for the first pass detection of outliers, and the second to mask pixels adjacent to pixels zapped on the first pass. The second SNR ratio is usually set to a smaller, and thus more rigorous, value than the first.

```
cl> dpar driz_cr
driz_cr.inlist = "../SA22_???.hhh"
driz_cr.group = "2"
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"
```

If the $|\text{data\_image} - \text{blotted\_image} - \texttt{scale} * \text{deriv\_image}| > \texttt{SNR} * \textit{noise}$, where `scale` is defined by the user and *noise* is calculated on a per pixel basis from the gain and read noise, then the pixel is masked, otherwise it is considered good.

driz_cr creates a mask file in the pixel list format (.pl) with _cr appended to the basenames, *e. g.* SA22_001_cr.pl. The pixel list format is a very compact format for files which have largely repetitive values.

The user should now visually compare the cosmic-ray masks with the data images and the blot images. If the scale parameters given to **driz_cr** were too large then cosmic rays on stars will be missed; conversely, if the scale parameters are too small, stars (typically the central pixel) will be unnecessarily masked. The scale parameter should be adjusted and **driz_cr** re-run until the results are satisfactory.

We next multiply each cosmic-ray mask by the static mask.

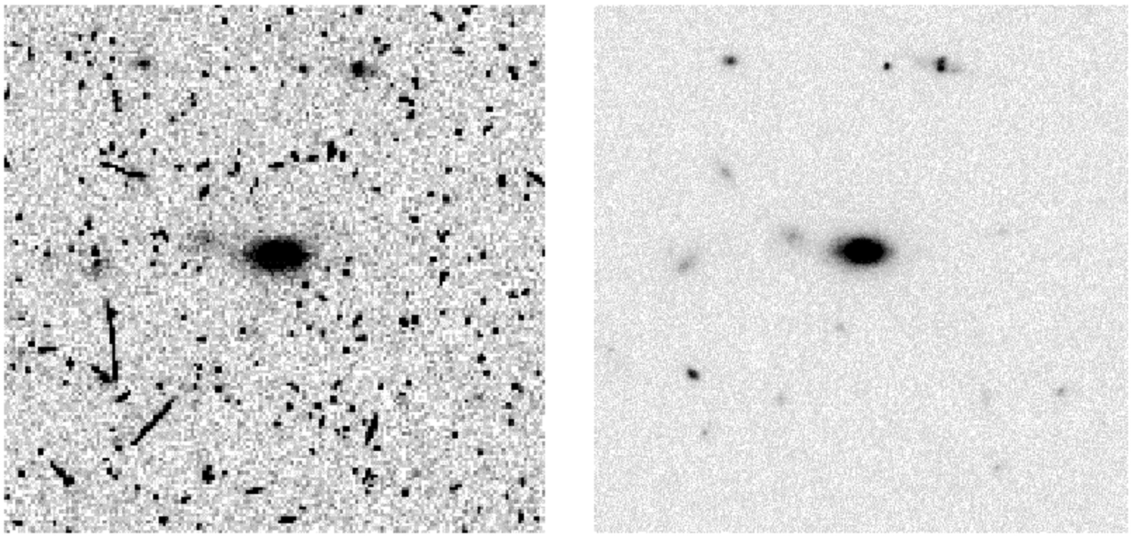

Figure 2. A section of one of the three dithered F814W images of the Hawaii Deep Survey field SA22 with cosmic rays (left), and the final drizzled image with cosmic rays removed (right).

```
imarith mask * SA22_001_cr.pl SA22_001_cr.pl
imarith mask * SA22_002_cr.pl SA22_002_cr.pl
imarith mask * SA22_003_cr.pl SA22_003_cr.pl
```

etc.

8. The Final Combination

Finally, we drizzle the input images onto a single output using the cosmic-ray masks, using the `cl` command to run a file such as:

```
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=0.0
drizzle  ../SA22_001.hhh[2] sum_g2 outweig=sum_g2w in_mask=SA22_001_cr
xsh=0.0 ysh=0.0
drizzle  ../SA22_002.hhh[2] sum_g2 outweig=sum_g2w in_mask=SA22_002_cr
xsh=-0.257 ysh=50.198
drizzle  ../SA22_003.hhh[2] sum_g2 outweig=sum_g2w in_mask=SA22_003_cr
xsh=-50.481 ysh=49.998
drizzle  ../SA22_004.hhh[2] sum_g2 outweig=sum_g2w in_mask=SA22_004_cr
xsh=-50.171 ysh=-0.169
```

etc...

The final output image is then sum_g2.hhh. The process described here can now be repeated for each CCD.

9. Improving shifts and cosmic-ray rejection through iteration

Although a single pass through the data frequently provides excellent results, there are a number of cases where the user may wish to iterate to improve upon either estimates of the shifts or the cosmic-ray rejection.

Let us first consider the case for using iteration to improve the estimated shifts. The combined data provides a far superior template for cross-correlation compared to the single image used above. Furthermore, the cosmic-ray masks allow one to "correct" pixels hit by cosmic rays, by replacing these pixels (in a temporary image) with the corresponding pixels from the blotted image. Therefore one now has both better templates and individual images with which to work. In order to find improved shifts, then, one can cross-correlate the zero-shift blotted image with each of these temporary images (including the one associated with the nominal position). Now all images are on an equal footing, and we would expect even the image originally used as the template to show an (albeit small) offset from the new blotted template. However, if the errors in the shifts from the first determination are already of order one-tenth the full width at half maximum of the point spread function or less, this procedure is unlikely to produce a significant gain in science.

Iteration can also improve cosmic ray rejection. The scale parameter used in **driz_cr** was very liberal. This was required by our taking the median of images drizzled with `pixfrac = 1.0`. However, now that we have masks that remove the overwhelming majority of cosmic rays the user may wish to repeat the drizzle onto independent output images using a very small `pixfrac`, for instance `pixfrac = 0.25`, and then take the median of these images. This median image will produce sharper stellar images, and when blotted back onto the original image plane will have far better agreement with the orginal image, allowing a substantially smaller `scale` to be used in **driz_cr**. The user might also wish to use the smaller `pixfrac` when performing the final drizzle with new cosmic-ray masks. In the case of the WF cameras of WFPC2, this can slightly reduce the width of the final PSF as well as decrease the correlation of the noise in the final image. However, in order to insure both photometric accuracy and a pleasing PSF, `pixfrac` should be kept large enough to so that the standard deviation of the final output weight image is no larger than about $\sim 30\%$ of its mean.

10. Making a mosaic

Although it is quite possible to drizzle all four chips onto a single output mosaic, doing so, is in actuality a complicated affair, because of the need to add the large relative offsets of the individual chips with the small dithers between images in the proper reference frame. As part of the further development of the **dither** package, we expect a task will be created to drastically simplify this process. However, in the short term, if the user is happy with the original resolution of the HST images for the purpose of the mosaic, a simple solution exists. One can blot the final drizzled images onto a common reference frame (the nominal position zero, for instance). Then one can copy each of these blotted images individually into a single four group geiss file, and then use the task **wmosaic** to comibine these into a single mosaiced image. In the following example we create a temporary file, `wf_mos`, copy the appropriate blotted images into it, and then after loading the necessary packages, use wmosaic to create the final mosaiced image, `MOS`.

```
di> gcopy SA22_001 wf_mos groups="ALL"
```

```
SA22_001 -> wf_mos
di> gcopy group1/sum_g1 wf_mos[1]
group1/temp_g1 -> wf_mos[1]
di> gcopy group2/sum_g2 wf_mos[2]
group2/temp_g2 -> wf_mos[2]
di> gcopy group3/sum_g3 wf_mos[3]
group2/temp_g3 -> wf_mos[3]
di> gcopy group4/sum_g4 wf_mos[4]
group2/temp_g4 -> wf_mos[4]
di> hst_calib
hs> wfpc
wf> wmosaic wf_mos MOS
```

11. Obtaining the software

More information on dithering, drizzling, and the removal of cosmic rays can be found at "Andy's Dithering Page" (http:www.stsci.edu/ fruchter/dither). A link for retrieval of the **ditherII** package will be available the week of 21 June 1998. Potential users of this software should understand that it is a "beta" release of scripts written by a scientist (not a computer scientist) to test algorithms. These programs are therefore slow. Faster versions will become available, once we assure ourselves that these algorithms will provide the large majority of users with good results.

References

Cowie, L. L., Hu, E. M., & Songaila, A. 1995, Astron. J., 110, 1576

Fruchter, A. S., & Hook, R. N. 1997, in Applications of Digital Image Processing XX, Proc. SPIE, Vol. 3164, ed. A. Tescher (SPIE), 120

Fruchter, A. S., Hook, R. N., Busko, I. C., & Mutchler, M. 1998, in 1997 HST Calibration Workshop, ed. S. Casertano, R. Jedrzejewsky, T. Keyes, & M. Stevens (STScI)

Williams, R. E., et al. 1996, Astron. J., 112, 1335