

Using *MultiDrizzle* to combine Dithered WFPC2 Images

Gabriel Brammer, Anton Koekemoer, and Bulent Kiziltan

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218

Abstract. This poster presents a guide to using the new *MultiDrizzle* Pyraf script to combine sets of dithered WFPC2 images. The *MultiDrizzle* script has condensed the steps of drizzling multiple images, as shown in the *HST Dither Handbook* (Koekemoer, et al. 2002), into a single Pyraf command with a number of parameters governing its behavior. This is aimed at greatly improving the ease with which images can be registered, cleaned of cosmic rays, and combined together using *drizzle* and related tasks. Images that have been produced using *MultiDrizzle* to combine WFPC2 datasets from the *Dither Handbook* examples are presented, and the results from a variety of parameter settings are explained and compared.

1. Introduction

Previously, a set of tools were provided in the IRAF/STSDAS *dither* package to analyze dithered data obtained with *HST*. These tools address a variety of issues, such as registration, cosmic ray cleaning, and combination. The *dither* tasks are extremely flexible, but are also extremely complex with a large number of parameters.

A new technique is now available—*MultiDrizzle* (Koekemoer et al. 2003, this volume, p. ??)—which automatically calls the *dither* package scripts along with the *drizzle* program (Fruchter & Hook 2002) and the *PyDrizzle* script (Hack & Jedrzejewsky 2002), using default parameters designed to work for a wide range of images (while still allowing the parameters to be changed as necessary). *MultiDrizzle* is run as a single command in Pyraf, following standard IRAF syntax.

2. Some *MultiDrizzle* Parameters

The initial steps carried out by the *MultiDrizzle* script consist of the identification of bad pixels, subtraction of the sky background, and running *drizzle* to transform each of the individual input images onto a set of output images that are registered on a common frame. The script then combines these registered images to create a clean median image, which is subsequently transformed back to the original frame of each input image using the *blot* task.

The next step involves the creation of the cosmic ray mask file. This is done by comparing each original input image with its counterpart “blotted” clean image, together with a third image that represents the spatial derivative of the “blotted” image. This comparison is carried out by the task *driz_cr*, and it uses the following algorithm:

$$|original\ image - blotted\ image| > scale \times derivative + snr \times rms . \quad (1)$$

The two important parameters in the step are:

- *driz_cr_scale*—this takes into account the possibility that slight offsets in the shifts may be present, which could cause inappropriate rejection of valid pixels, such as bright stellar cores. Increasing the value of *scale* will help ensure that such misidentifications occur less often; good values to use for *scale* typically range from 1–2.

- *driz_cr_snr*—this is simply a multiplicative scaling of the rms, which has been calculated by taking into account the sky background (stored in the header), together with the readnoise and gain, as well as the observed flux in the pixel. Typical values for *snr* that will yield good results are in the range 3–5; higher values will lead to fewer cosmic rays being rejected, while lower values will cause more frequent rejection of pixels that are not necessarily cosmic rays.

After creating the cosmic ray masks, the final step is to drizzle all the input images onto a single output image, using the information from the masks to exclude pixels in each input image that have been affected by cosmic rays. The task *drizzle* has several parameters, but two of the most important are:

- *final_scale*—the size of output pixels relative to the input pixels. If the sub-pixel space is reasonably well sampled by more than a few dithers, then it is acceptable to consider setting the value of *scale* small enough to provide critical sampling of the PSF. Typically, this means that values down to ~ 0.4 – 0.5 can be considered for *scale*.
- *final_pixfrac*—the size by which input pixels are “shrunk” before being mapped onto the output grid. The values for this range between 0 and 1: if *pixfrac* = 0, then this is equivalent to interlacing (each input pixel will only ever contribute to a single output pixel), while at the other extreme, *pixfrac* = 1 corresponds to “shift-and-add.” In this case the output image is convolved by the full size of the input pixels. For a typical case of ~ 4 sub-pixel dithers and *scale* = 0.5, reasonable values for *pixfrac* would be in the range 0.6–0.8. This allows some sharpening of the PSF relative to *pixfrac* = 1, while at the same time still retaining reasonably uniform coverage of the output pixel grid.

For more information on the many *MultiDrizzle* parameters, as well as a detailed description of the script’s intermediate actions and products, please refer to the paper by Koekemoer et al. (2003, this volume) and to the *MultiDrizzle* web page, located at:

<http://www.stsci.edu/~koekemoe/multidrizzle/>

3. Executing *MultiDrizzle* on a Set of WFPC2 Images

This section goes through a step-by-step explanation of how to run *MultiDrizzle* on a set of 3 dithered exposures of the edge-on spiral galaxy NGC 4565 (ID 6092, PI: Keith Ashman) used in Example 2 of the *HST Dither Handbook* (Koekemoer et al. 2002). Note that *MultiDrizzle* products require substantial free disk space. The GEIS-formatted images for this example can be downloaded from:

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

MultiDrizzle is run in the Pyraf command environment, loaded with *pyraf* at the unix command prompt. More information about Pyraf can be obtained at the STScI Pyraf home page:

<http://pyraf.stsci.edu>

Move to the directory containing the images for this demonstration, create an input image list, and set up *MultiDrizzle*:

```
$ cd /data/mydir/
$ ls -l *.c0h > files.list
$ pyraf
  pyexecute('/data/wallaby1/anton/multidrizzle/multidrizzle_iraf.py',
--> import multidrizzle
--> unlearn multidrizzle
```

files.list should contain:

```
u31s0101t.c0h
u31s0102t.c0h
u31s0103t.c0h
```

Run *MultiDrizzle* for the WFPC2 images with shifts calculated from the image headers:

```
--> multidrizzle output='final' filelist='files.list' inst='WFPC2'
```

The drizzled output images are 4-chip mosaics with the pixel scale of the PC (0.05"/pixel). Since the WF chips have twice the pixel scale of the PC (0.1"/pixel), the output images are approximately

$$\frac{2 \text{ chips} \times 800 \text{ pixels} \times 2 \text{ pixel scale}}{\text{final.scale} = 1} = 3200 \quad (2)$$

pixels on a side. Display the drizzled science and weight images (see Figure 1), and compare to an individual input frame:

```
--> display final_sci.fits 1 zr- zs- z1=-.1 z2=.5
--> display final_wht.fits 2 zr- zs- z1=0 z2=500
--> display u31s0101t.c0h[3] 3 zr- zs- z1=0 z2=100
```

4. *MultiDrizzle* Products

The output science images have units of counts s^{-1} , and the weight image is a map of the effective exposure time per pixel. Note how the final science image stitches together the four WFPC2 chips and how the cosmic rays seen in the input image are eliminated. The slight level differences between the chips, (e.g., between WF2 and WF3 at upper and lower left, respectively) arise from difficulties in determining the sky background level for images of an extended source that entirely fills the camera's field of view.

The weight map shows how pixels affected by cosmic ray events on one or more input exposures have lower effective exposure times than those unaffected by cosmic rays. In this example with three exposures of 160 s each, a pixel affected by cosmic rays in all three images would have an effective exposure time (or weight) of zero, while a clean pixel in all three exposures would have an effective exposure time of 480 s. The PC and WF chips scale differently in the weight image because of the difference in pixel scales discussed above. *MultiDrizzle* corrects for the geometric distortions of the WFPC2 chips, and this correction is also visible in the weight images. From the distortion, the area of a single pixel projected onto the sky decreases towards the corners of the chips, and this decrease is incorporated in the flatfield file. Thus, the weight of the pixel needs to be increased correspondingly in order to preserve total flux.

5. Conclusions

MultiDrizzle produces clean, registered drizzled images of dithered exposures through an interface that greatly simplifies the dither process. For the example described above, the process of producing properly drizzled products that required more than one hundred commands using the IRAF *dither* package has been reduced to a single Pyraf command.

References

- Fruchter, A. & Hook, R. 2002, *Drizzle: a method for the linear reconstruction of undersampled images*, PASP, 114, 144.
- Hack, W. & Jedrzejewsky, R., 2002, *Pydrizzle User's Manual*, <http://stsdas.stsci.edu/pydrizzle>

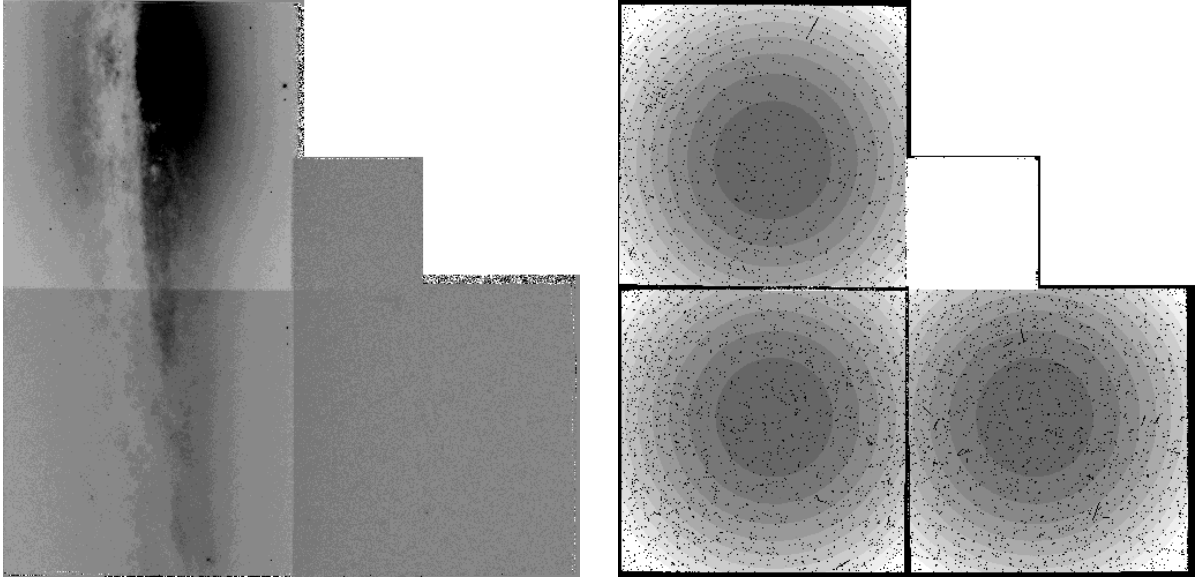


Figure 1. **(left)** *MultiDrizzle* output science image, `final_sci.fits`, with inverted color scale. **(right)** *MultiDrizzle* output weight image, `final_wht.fits`. The display of the weight image has been stretched to highlight the geometric distortion corrections in the WF chips. The gradient from the centers to the edges of the chips is about 3% of the modal WF pixel value. The PC shows a similar effect, but at a level that is outside the image stretch as a result of the difference in the pixel scales of the PC and WF chips.

Koekemoer, A. M., et al. 2002, *HST Dither Handbook*, Version 2.0 (Baltimore: STScI)

Koekemoer, A. M., Fruchter, A. S., Hook, R., & Hack, W. 2003, this volume, ??