

## Multidrizzle: Automated Image Combination and Cosmic-Ray Identification Software

Robert Jedrzejewski, Warren Hack, Chris Hanley, Ivo Busko, Anton M. Koekemoer

*Space Telescope Science Institute, Baltimore, MD 21218*

**Abstract.** MultiDrizzle, a task which automatically combines images taken at different pointings and removes cosmic rays and other image defects, is now in use in the Advanced Camera for Surveys calibration pipeline at STScI. This poster describes the motivation for the development of this capability, details the steps taken in MultiDrizzle processing and describes the design decisions made in implementing MultiDrizzle in the automatic HST calibration pipeline. Interested users can go to <http://stdsdas.stsci.edu/multidrizzle> for more information.

### 1. Motivation

Since its introduction in March 2002, the ACS calibration pipeline has combined images taken at slightly different pointings using the PyDrizzle<sup>1</sup> software. This Python application automates the process of combining ACS images taken at different dither positions. While this has simplified the task of running the Drizzle task (see Fruchter & Hook 2002), it was not designed to do anything more than combine images that were already assumed to have been cleaned of cosmic rays and other image defects. However, many ACS users started specifying observing programs that took only one image at each dither position, assuming that the cosmic ray identification and removal could be done in post-pipeline processing. While this is true (see the Dither Handbook<sup>2</sup> (Koekemoer et al. 2002) for details), the process is time-consuming and presents a bewildering set of calibration options to the unwary user.

### 2. History

The development effort for MultiDrizzle is led by Anton Koekemoer, who wrote the prototype version (Koekemoer et al. 2002) in consultation with A. Fruchter and other members of the Dither Working Group at STScI. It was conceptually based on earlier tasks written individually by Koekemoer, Fruchter and others, all of which aimed to simplify the interface to the tasks needed for processing dithered data.

---

<sup>1</sup>[http://www.stsci.edu/resources/software\\_hardware/pydrizzle](http://www.stsci.edu/resources/software_hardware/pydrizzle)

<sup>2</sup>[http://www.stsci.edu/instruments/wfpc2/Wfpc2\\_driz/dither\\_handbook.html](http://www.stsci.edu/instruments/wfpc2/Wfpc2_driz/dither_handbook.html)

The fundamental goal behind the design of MultiDrizzle was to provide a seamless “one touch” interface to the variety of dither-related routines, while also providing powerful flexibility through a range of parameters. The prototype was written in Python, acting as an interface to the various IRAF cl scripts and tasks set out in the Dither Handbook (Koekemoer et al. 2002):

- Remove sky background from the individual images
- Remove static defects from individual images
- Determine output WCS by examining individual image headers
- Drizzle each image onto individual aligned outputs
- Median-filter images to generate cleaned reference image
- Blot median image back to geometry of each input image
- Compare each input image with blotted reference, masking significantly discrepant pixels
- Drizzle each masked input image onto a final output product

The prototype version was released as part of STSDAS 3.2 in March 2004. This version then served as a baseline for a completely redesigned version by the Science Software Branch at STScI, aiming to retain the fundamental functionality, but making the code more robust and streamlined. The goal of this version was to run in the automatic HST calibration pipeline for ACS images. As such, the requirements were:

- Reduce (or remove) dependence on IRAF tasks
- Perform most image manipulations in memory rather than reading to and writing from disk
- Provide detector-specific reduction parameters
- Limit total amount of memory required for image combination
- No significant speed reduction compared to prototype code
- Robust enough to handle all anticipated associations

### 3. Implementation

MultiDrizzle for the pipeline is implemented in Python, with some C extensions for performance-critical sections. Most of the calls to IRAF tasks in the prototype version of MultiDrizzle involved either array manipulation (for example, determination of the sky background by iteratively clipping the image histogram, and combining the images using either the median task or some concatenation of imcalc calls) or header keyword getting/setting. This was handled in the redesigned version by using `numarray` objects to represent the data and bad pixel masks, and by using `PyFITS` to handle file I/O and header keyword manipulation. This also has the benefit of allowing image operations to be done in memory, so that numerous read/write operations are unnecessary and makes the redesigned version significantly faster than the prototype.

Some image operations could not be handled efficiently by using pure Python code operating on `numarray` objects; for these operations (image statistics, image combination), C extensions were written. The total memory usage was limited by using an image iterator that operates on only part of an image at once, so that, for example, in the image combination step, it is not necessary to have all of the images in memory at once. This means that the total memory usage is determined by the size of the output image, not by the number of images that

are to be combined (which can grow to be very large in the case of programs like the Ultra-Deep Field).

The detector-specific section of the code allows different behavior depending on whether the detector is the Wide Field Channel, High Resolution Channel or Solar Blind Channel. Using standard object-oriented methods, the detector-specific code could be isolated, making it relatively straightforward to add the capability of processing other instruments than ACS. Other detector-specific behavior includes choosing different reduction parameters depending on the channel in use; this is handled using a reference file that lists the MultiDrizzle parameters for the observational configuration selected. In this way, for example, the sky subtraction could be handled differently for narrow-band images since most data of this type has a negligible background and so sky subtraction is not necessary.

An example of the product that the new MultiDrizzle generates in the HST ACS calibration pipeline is shown in Figure 1.

Since most of the code is either pure Python or C extensions, it can be ported to platforms other than the standard Unix that IRAF runs on. We have successfully built and run MultiDrizzle on a system running Microsoft Windows.

MultiDrizzle was installed in the ACS calibration on September 21, 2004. The acquisition of a new multi-processor system for running the HST operational pipeline and On-the-Fly Reprocessing means that there is enough CPU capacity to support this compute-intensive application.

MultiDrizzle is included in STSDAS 3.3, which was released in November 2004. As well as providing support for ACS images, it also processes WFPC2, STIS imaging and NICMOS data.

## References

- Fruchter, A. S. & Hook, R. N. 2002, *PASP*, 114, 144
- Koekemoer, A. M. et al. 2002, *HST Dither Handbook V2.0*, Baltimore: STScI.
- Koekemoer, A. M., Fruchter, A. S., Hook, R. N., Hack, W. 2002, *HST Calibration Workshop* (eds. S. Arribas, A. Koekemoer & B. Whitmore, STScI:Baltimore).

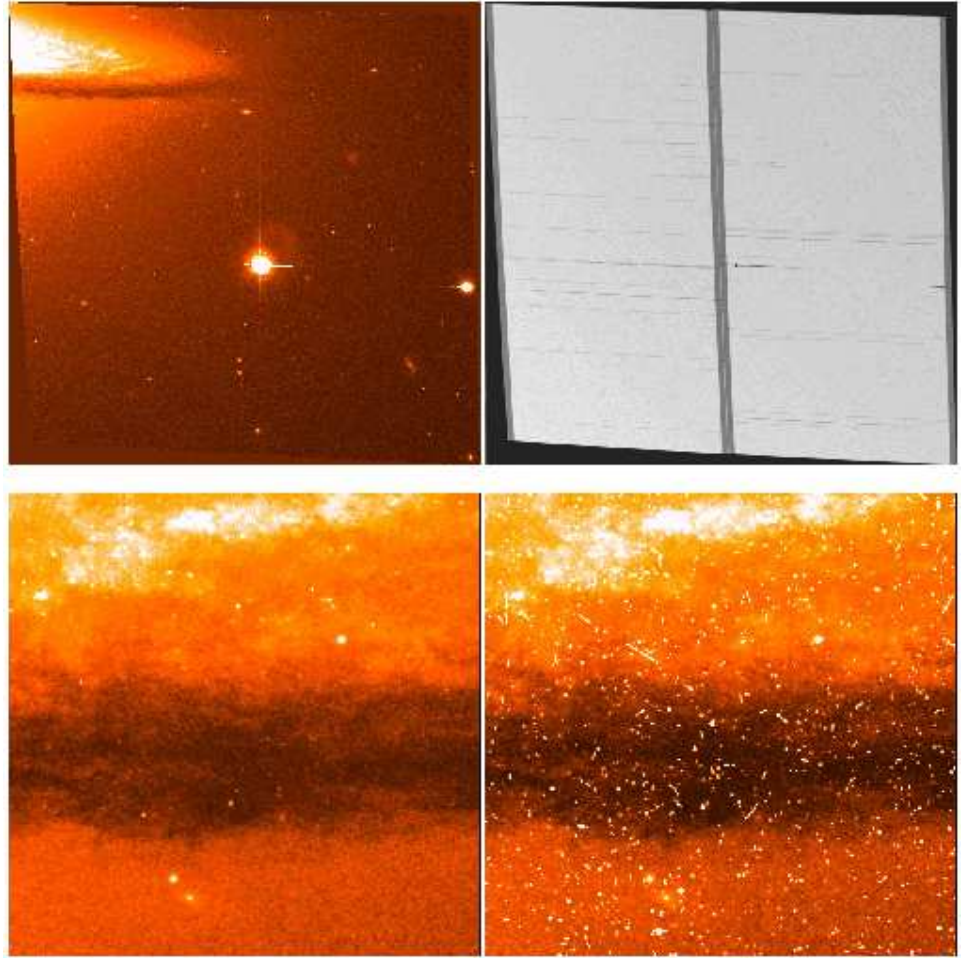


Figure 1. An example of the cleaned MultiDrizzle product from the ACS Archive Pipeline, for a dithered 4-exposure observation of the nearby galaxy NGC 4594. The top panels show the final full-frame drizzled and weight images (left and right, respectively). The bottom left panel shows a close-up of the output image from MultiDrizzle, while the bottom right panel shows the sum of all the accumulated cosmic rays originally present in the exposures