1997 HST Calibration Workshop
Space Telescope Science Institute, 1997
S. Casertano, et al., eds.

NICRED: Reduction of NICMOS MULTIACCUM Data with IRAF

Brian A. McLeod
Harvard-Smithsonian Center for Astrophysics, 60 Garden St., Cambridge, MA 02138

Abstract. I describe a set of algorithms and associated IRAF package, NICRED, for reducing NICMOS MULTIACCUM data. In particular, this package addresses the problems of cosmic ray rejection, and the bias problems seen in early NICMOS data.

1. Introduction

NICMOS MULTIACCUM mode is a powerful data-taking technique that can substantially increase the dynamic range of an image, and can provide substantial cosmic-ray rejection capability without significant loss of data. The down-side is that the data are somewhat more difficult to reduce than previous HST data. The NICRED package was developed to allow reduction of the data from within the IRAF environment and independently of the STScI pipeline software, CALNIC, described elsewhere in this volume. NICRED requires the IRAF FITS kernel, which is built into IRAFv2.11, but is easily installed into version 2.10. All of the tasks operate directly on multiextension FITS files.

2. Dark Subtraction

Dark subtraction, done with `darksub`, is a straightforward procedure consisting of subtracting from each readout the corresponding readout from a dark file with the same sample sequence. Example raw and dark-subtracted images are shown in Figures 1 and 2.

Dark current and the sky background may also be subtracted together by constructing a sky frame from dithered observations, much like the traditional ground based approach. For those users who have enough dither positions to construct such a frame, the task `multicomb` will be of use to construct the sky frame. It combines each of the readouts from several dither positions, producing as output another multi-readout file that can be used for sky-subtraction.

3. MULTIACCUM Fitting and Cosmic Ray Rejection

The non-destructive readout mode of NICMOS allows for rejecting cosmic rays from exposures with a minimum loss of information. In the absence of cosmic-rays, the count-rate would be determined by doing a linear least-squares fit to the signal as a function of time. The slope of the fit is the count-rate. In the presence of cosmic rays, we also fit a slope, but we allow for a jump in the intercept where the cosmic ray occurs. The program `fullfitbam` performs this function, assuming that there is only one or zero cosmic-ray hits per pixel in a given image. This is a reasonably good assumption for all but the longest exposures. Since we do not know *a priori* whether there is a cosmic ray hit in a pixel, or when it occurred, we must make this determination. For each pixel with N readouts, we consider the N possible times that a cosmic ray could have hit, and fit a slope and two intercepts, a and b. Then

Figure 1. A raw image.

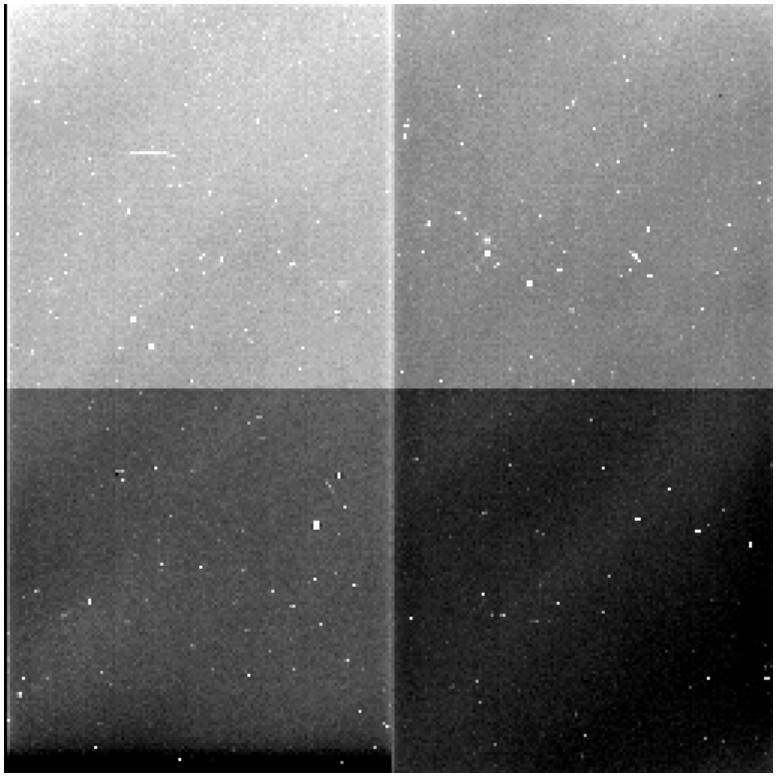


Figure 2. A dark-subtracted image.

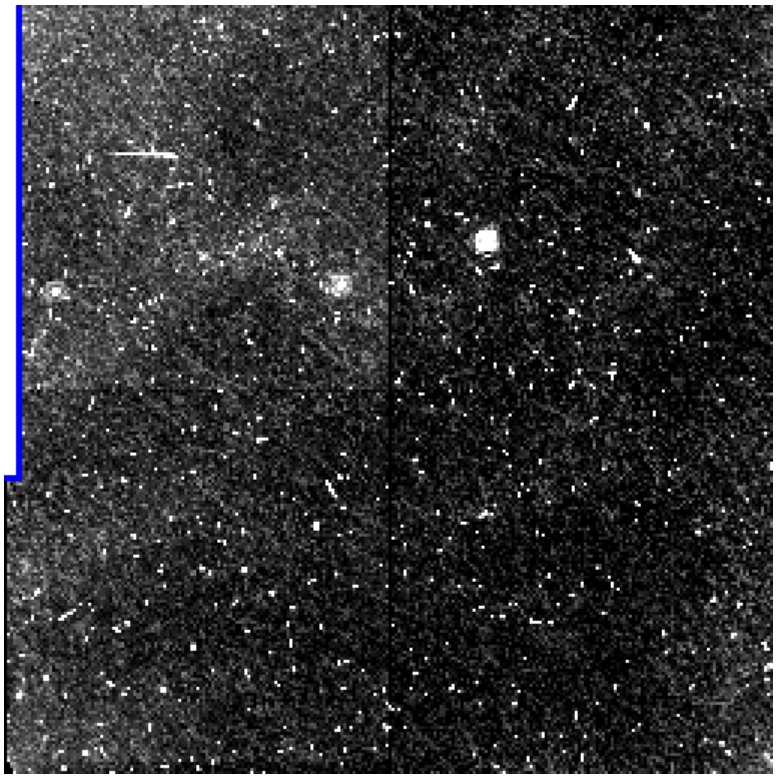
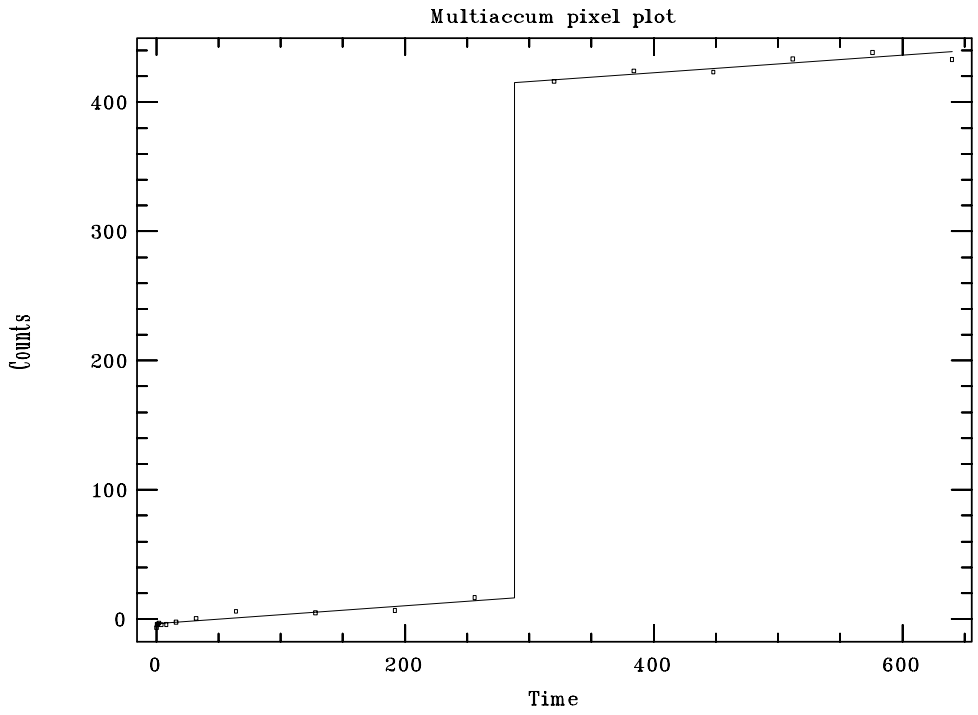


Figure 3. A cosmic ray hit and the fit to the data



we compute the residuals of the fit and compare with the difference in the two intercepts. If $a-b$ is large compared with the uncertainty in $a-b$, as determined from the residuals, it is likely that a cosmic ray hit occurred between the readouts. Note that even in the event of a cosmic ray hit *all data readouts are used* unless the pixel is saturated!

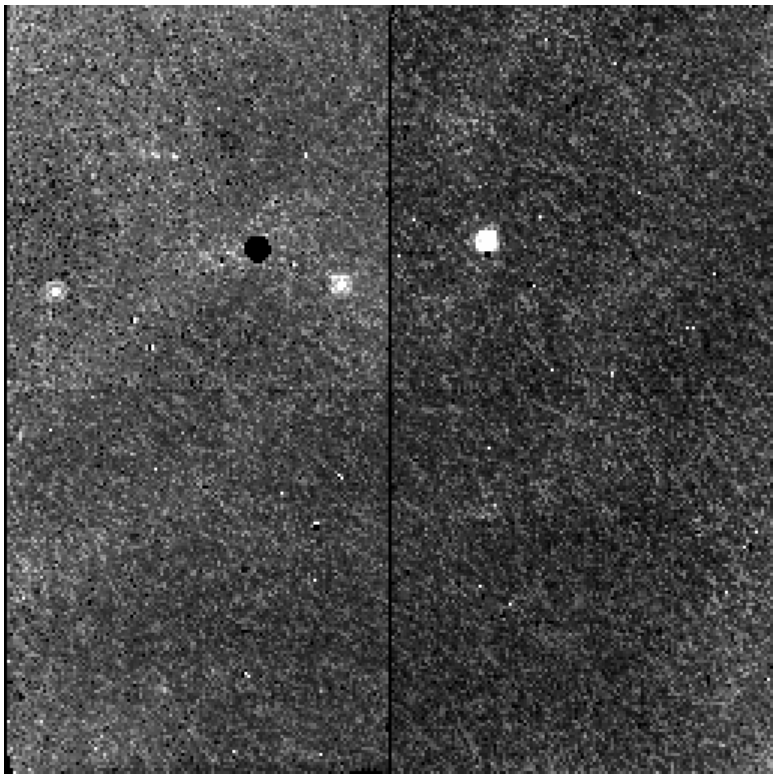
Lets make this process more explicit. For a trial cosmic ray hit occurring between readout times, t_j and t_{j+1}, we fit the constants m, a, and b, in the expression,

$$y = \begin{cases} mt + a & t < t_j, \\ mt + b & t > t_j \end{cases} .$$

Then, define the merit function, $M = (a-b)/\sigma_{a-b} \approx (a-b)/\sqrt{\sigma_a^2/j + \sigma_b^2/(n-j)}$, where σ_a and σ_b, are the residuals over the points with $t < t_j$ and $t > t_j$, respectively. We compute the merit function over each possible j, and choose the largest value. This is where the cosmic ray occurred. If the largest value of M doesn't exceed a user-defined threshold, no cosmic ray is flagged and we do a standard fit with one intercept. The program also enforces the condition that $\sigma_a, \sigma_b >=$ readnoise. When N is small, this prevents statistical flukes of fits with unusually small sigmas from being flagged as cosmic ray hits. I have typically been using a threshold of $M = 3$–5, and readnoise=10ADU.

To assist in tuning up the parameters for the fit, there is a utility program, `multiplot`, which plots the data values vs. time, and shows the fit. The pixel to plot is selected with the image cursor. Figure 3 shows a pixel with a cosmic ray hit and the associated fit. Figure 4 shows a slope-fitted, cosmic-ray rejected image.

Figure 4. Result of fitting cosmic rays



4. Bias Problems

Early observations with NICMOS have suffered from a drifting bias level, also known as the "pedestal effect" (NICMOS STScI Analysis News #4, August 1997). The result is that each readout contains the true signal plus an additional value constant over the array, but varying with readout number. This problem is propagated by the fitting process so that the final image also suffers from an additional constant. After applying the flat, the constant component is "unflattened" so that the resulting image shows the flat-field image in reverse (see Figure 5).

Another effect of the wandering bias is to make the MULTIACCUM data ramps non-linear. Since it is constant over the array, the slopes are affected equally, but the cosmic ray rejection algorithm gets confused because it detects false excursions from the expected linear ramp. This can result is a speckled-looking image when some pixels have false cosmic rays detected in the presence of the non-linear ramp. NICRED solves both of these problems.

The output of `fullfitbam` can be flattened using `nicflatten`. First, the user specifies threshold values, between which it is assumed the image contains blank sky. If the image is extremely crowded, this program may not work. It is assumed that the spatial variations of the blank sky on the detector are identical to that of the flat. The program then minimizes the quantity $\sum_i (S_i - aF_i - b)^2$, where a and b are the fitted constants, and S_i and F_i are the pixels in the slope image and the flat. The fit is only done over those image pixels meeting the threshold requirements. In other words, the flat and an additive constant are fitted to the sky regions of the image. The residual of the fit is the sky-subtracted, bias-corrected image. This procedure is done independently for each of the four quadrants of the image, with the result shown in Figure 6.

Figure 5. Un-bias corrected, flattened image. Residuals from the flat are apparent.

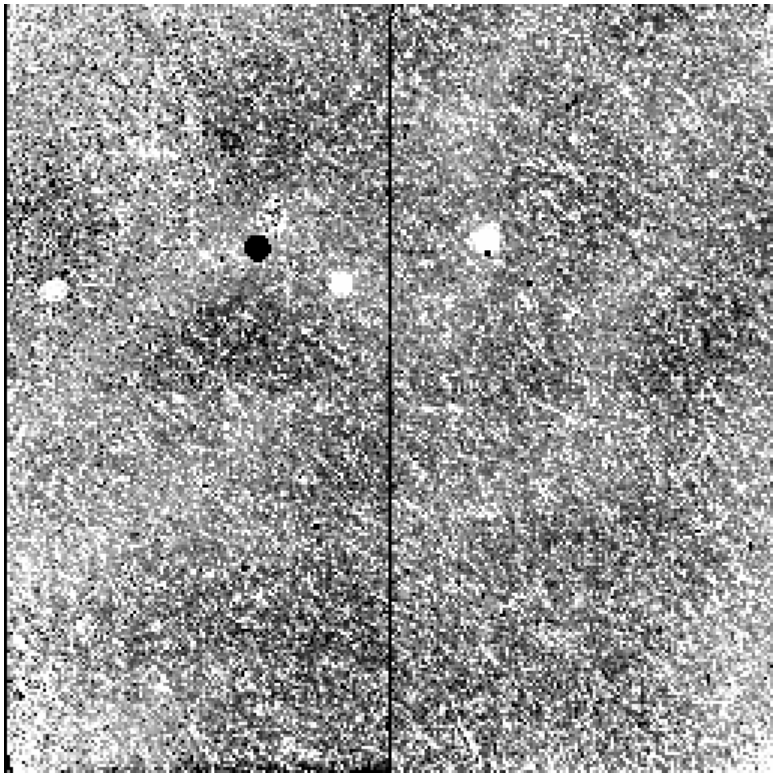
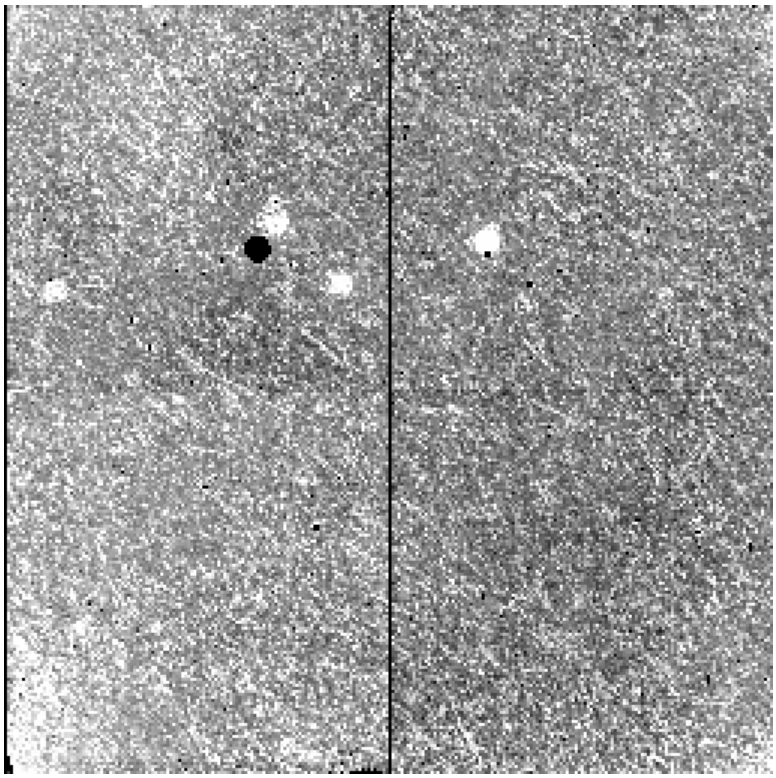


Figure 6. Sky-subtracted, bias-corrected, flattened image.



Given this initial estimate of the cumulative bias drift, we can eliminate the bias drift in the individual readouts. The bias error for frame j is $E_j = Median_i(I_{ji} - t_j(S_i - b))$, where t_j is the sample time for frame j, I_{ji} is the ith pixel in the jth frame, and as before S_i is the fitted slope (counts/sec) for pixel i. The bias is removed from the slope image with `biasfix`, and the individual biases are removed with the IRAF script `nicbias`. Once the individual bias errors are removed, `fullfitbam` is run again to refit the slopes and cosmic rays, and `nicflatten` is rerun to flatten and sky subtract.

5. Summary of the Processing Steps

- `darksub` – Subtract the dark current from each frame.
- `fullfitbam` – Linearize, check for saturations, fit the slopes, and remove the cosmic rays.
- `biasfix` – Remove bias error from initial slope image.
- `nicbias` – Remove bias errors from MULTIACCUM frames.
- `fullfitbam` – Recompute slopes and remove cosmic rays.
- `nicflatten` – Compute flattened, sky-subtracted image.

6. Obtaining NICRED

The NICRED package is available on the World Wide Web at `http://cfa-www.harvard.edu/\~bmcleod/Nicred`, or by contacting the author at `bmcleod@cfa.harvard.edu`.

Acknowledgments. I would like to thank Marcia Rieke for providing her MULTIACCUM fitting program `fullfit`, which mutated into the program described here. My collaborators Emilio Falco and Joseph Lehár tested the programs and found various bugs. John Roll provided the FITS library `fitsy`.