# 7.1 WFC3/UVIS: Optimizing Image Alignment for Multiple Visits

**Introduction**

This example demonstrates the use of the **tweakreg** and **astrodrizzle** tasks to align and combine three UVIS images, each taken in different visits at different orientations.

There is a great deal of explanatory text in this example; users new to DrizzlePac are encouraged to start here.

## Summary of Steps

1. Description of the data
2. The images are aligned to the coordinate frame of a reference image using the **tweakreg** task. In this example, one of the three images has been selected as the reference image.
3. After the three images have been aligned to a common WCS, they're drizzle-combined using AstroDrizzle.

## 7.1.1 Description of the Data

In this example, three single full-frame WFC3/UVIS images of the globular cluster Omega Centauri (WFC3/CAL Program 11911), each with exposure time of 40 seconds, taken in F606W, will be combined using **astrodrizzle**, after the images are aligned using **tweakreg**. These observations were acquired over a 6-month period in three separate visits at different orientations and offsets.

**Table 7.2: Summary of Images in this Example**

| Image Name | Association ID | Proposal ID | Visit & Line Number | Pointing[1] (degrees) | PA_V3 Orientation (degrees) | Observation Date | Exposure Time (sec.) |
|---|---|---|---|---|---|---|---|
| ibc301qrq_flt.fits | NONE | 11911 | 01.001 | 201.6821267139 -47.46661075341 | 105.0002 | 2010-01-14 | 40.00 |
| ibc304v3q_flt.fits | NONE | 11911 | 04.001 | 201.7121069398 -47.47292818852 | 199.9971 | 2010-04-29 | 40.00 |
| ibc307qyq_flt.fits | NONE | 11911 | 07.001 | 201.7051092234 -47.49082103744 | 279.9995 | 2010-07-04 | 40.00 |

1. Original CRVAL1, CRVAL2, in degrees, at the reference pixel in [sci,1]

### 7.1.2 Aligning Images with tweakreg

In order to correctly align and combine multiple images into a distortion-free final drizzled product, AstroDrizzle relies on the World Coordinate System (WCS) information stored in the header of each image.

WCS information for an image is tied to the positions of guide stars used for the observation. As a result, if these three images were aligned based on the original WCS information in their headers, they would still have small offsets due to the uncertainty in guide star positions that can be as large as 0.5 arcseconds (12 pixels).

The TweakReg software can align a set of images to better than subpixel accuracy (< 0.1 pixel). For each image, it computes the geometric transformation (shift, rotation, and scale) after correcting for geometric distortion relative to a reference image based on positions of stars common to both images. That transformation is used to update the WCS information for the image.

Like other **drizzlepac** tasks, **tweakreg** can be run several different ways: using Python syntax in either the PyRAF or Python environments; setting parameter values using a TEAL GUI[1]; using a preset parameter configuration file (with suffix `.cfg`) that can be loaded into the TEAL GUI, or run using Python directly to call the **tweakreg** module in the **drizzlepac** package.

In the example below, **tweakreg** is run as a Python command in PyRAF. First, the **drizzlepac** package is imported into the Python environment, followed by importing the **tweakreg** task from the **drizzlepac** package. (**imagefindpars** does not need to be imported because it is automatically called from within the **tweakreg** task.) **tweakreg** and **imagefindpars** are then reset to default parameters (in case non-default parameters were used in a previous run of the tasks). Finally, **tweakreg** is run with two non-default parameters values: ***threshold*** and ***peakmax***.

```
--> import drizzlepac
--> from drizzlepac import tweakreg
--> unlearn tweakreg
--> unlearn imagefindpars
--> tweakreg.TweakReg('*flt.fits',updatehdr=False,conv_width=3.5,\
threshold=2000,peakmax=50000)
```

**tweakreg** is typically run several times until the best fit is found for the image offsets. When the user is satisfied with the results, **tweakreg** is run a final time with ***updatehdr=yes*** so WCS information in the images can be updated using the calculated transformations required to align the images.

Listed below are some **tweakreg** parameters, as well as parameters from the **imagefindpars** task called by **tweakreg**, that are directly related to this example,

1. Running **tweakreg** from the TEAL GUI is helpful for several reasons: (1) task parameters, with descriptions are displayed. (2) Quick access to help files. (3) Task settings can be saved to uniquely-named configuration files (suffix `.cfg`) for later use or for record-keeping purposes. (4) Previously-created configuration files can be loaded for execution.

To set parameter values and run **tweakreg** in TEAL, type "`epar tweakreg`" in PyRAF. Don't forget to click the "ImageFindPars" button to open the **imagefindpars** TEAL window, and after setting those parameters, click "Save and Quit" to return to the **tweakreg** window to continue setting those parameters.

including those with non-default values. (Those not mentioned use default values.) The parameters are broadly grouped into three categories: logistics, source detection, and source matching.

### Logistics Parameters in tweakreg

***input***: images that need to be aligned. For this example, input files are expressed as "`*flt.fits`".

***refimage***: name of the reference image. Input images will be aligned with respect to the reference image. For this example, the default setting is used, which means that the reference image is the first image in the set.

***updatewcs***: the default setting, ***no***, is retained because the **updatewcs** task was performed in the pipeline.

***writecat***: the default value, ***writecat=yes***, is kept so that source coordinate lists are created for each image using an algorithm similar to IRAF's **daofind**.

***updatehdr:*** this switch determines if the geometric transformation solution for aligning each image, computed by **tweakreg**, should be used to update WCS information in each image header. It's highly recommended that this parameter be set to its default value, ***no***, until an optimal alignment solution is found for all images. This way, **tweakreg** can be run several times to optimize image alignment without any undue modification of header WCS information. When the best alignment solution is found, rerun **tweakreg** with this parameter set to ***yes***.

***headerlet***: when set to ***yes***, the newly-updated WCS information is written to a headerlet. More information on headerlets is available in Section 3.4. For the purposes of this example, the default value, ***no***, is used.

***shiftfile***: this shift file is in the same format as that used by **multidrizzle**, but it is only generated for informational purposes because offsets determined by **tweakreg** are used to directly update the WCS of each input image that will then be combined using **astrodrizzle**. The default setting, ***no***, is used for this example.

### Source Detection Parameters in imagefindpars

Parameters for source detection are in a task called by **tweakreg**, called **imagefindpars**. Optimal parameters may vary on a case by case basis. In general, the best way to determine the best combination of parameters is to explore the parameter space of these values until an adequate number of sources are found in each image. A few hundred sources (or more) is ideal, but accurate solutions may also be obtained using only a few dozen sources matched between images.

***threshold***: the threshold above the background for source detection, in units of sigma (standard deviation). The default value is 4. But for this example, previous tests determined that a value of ***2000*** yielded good results.

***peakmin*** and ***peakmax***: these cutoffs allow the user to reject sources that are too faint or too bright, based on the peak value of the source. ***peakmin*** and ***peakmax*** can be used to reject saturated sources, and bright and/or saturated cosmic rays. For the purposes of this example, ***peakmin*** is left blank (default setting), but ***peakmax*** is set to ***50000***.

***conv_width***: the convolution kernel width in scale units is used to identify potential sources in the image which look closest to a gaussian with the FWHM equal to this

value. The value is typically twice the FWHM of the point source. For this example, the default value, ***3,5*** is used.

***skyigma***: standard deviation of background in counts. Since ***computesig*** was set to ***yes***, this parameter is not used.

***computesig***

Automatically compute sigma for all inputs?

The default value, ***yes***, is used for this example. **tweakreg** automatically computes a value for sigma for each image, according to the following formula:

```
1.5 * imstatistics(image,nclip=3,fields='stddev')
```

(where **imstatistics** is an IRAF task, although a Python version of **imstatistics** actually gets used, not the IRAF task).

The computed sigma will override any user-specified value of sigma in the parameter ***skysigma***.

***fluxmin* and *fluxmax*** minimum and maximum total source flux.

These are cutoffs that allow the user to reject sources that are too faint or too bright based on the integrated flux in a circular region. The radius of that circular aperture is the value of the ***conv_width*** parameter. For this example, it was not necessary to set values for these parameters.

**Source Matching Parameters in tweakreg**

***searchrad***

The search radius for a match.

**tweakreg** uses this search radius for finding matched sources common to an image and its reference image. For this example, the value ***1*** (arcseconds), the default, is used. If the value is too small, an optimal fit solution will not be found because the residual offset between the images is larger than the search range. If the value is too large, **tweakreg** will take longer than needed to find an optimal fit solution. A larger search space also includes more sources, so there's an increased risk of finding false matches which can affect the final fit solution.

***searchunits***

Units for search radius.

The choice is pixels or arcseconds, the latter being the default that's also used in this example. (Note, the WFC3/UVIS plate scale is 0.04 arcsec./pixel, so a search radius of one arcsecond (the default) translates to 25 pixels.)

***use2dhist*** and ***see2dplot***

Use and display a two-dimensional histogram to find initial offsets for each image.

In this example, the default values, ***yes***, are used for both parameters. This creates a plot of the initial offset, as seen in Figure 7.1

***fitgeometry***

Fitting geometry.

There are two choices: ***shift*** and ***rscale***. The default, ***rscale***, is used here. It fits for *x* and *y* shifts, as well as rotation and scale. The ***shift*** option only fits for *x* and *y* offsets, with rotation assumed to be zero and scale factor assumed to be one.

***residplot***

Plot residuals from fit?

For the default parameter, ***Both***, a residual plot and vector plot are displayed for each image. The vector plot, which shows source magnitude with direction of residuals plotted as a function of location, is useful for spotting localized systematic deviations in image alignment. A good image alignment results in a vector plot that appears as small randomly-oriented vectors with no clear organized flows or structures. The residuals plot, which plots the *x* and *y* components of the residuals vs. *x*- and *y*-axis position, is useful for spotting subtle and/or larger scale and alignment issues. Good residual plots are flat with typical RMS values less than 0.1 pixels.

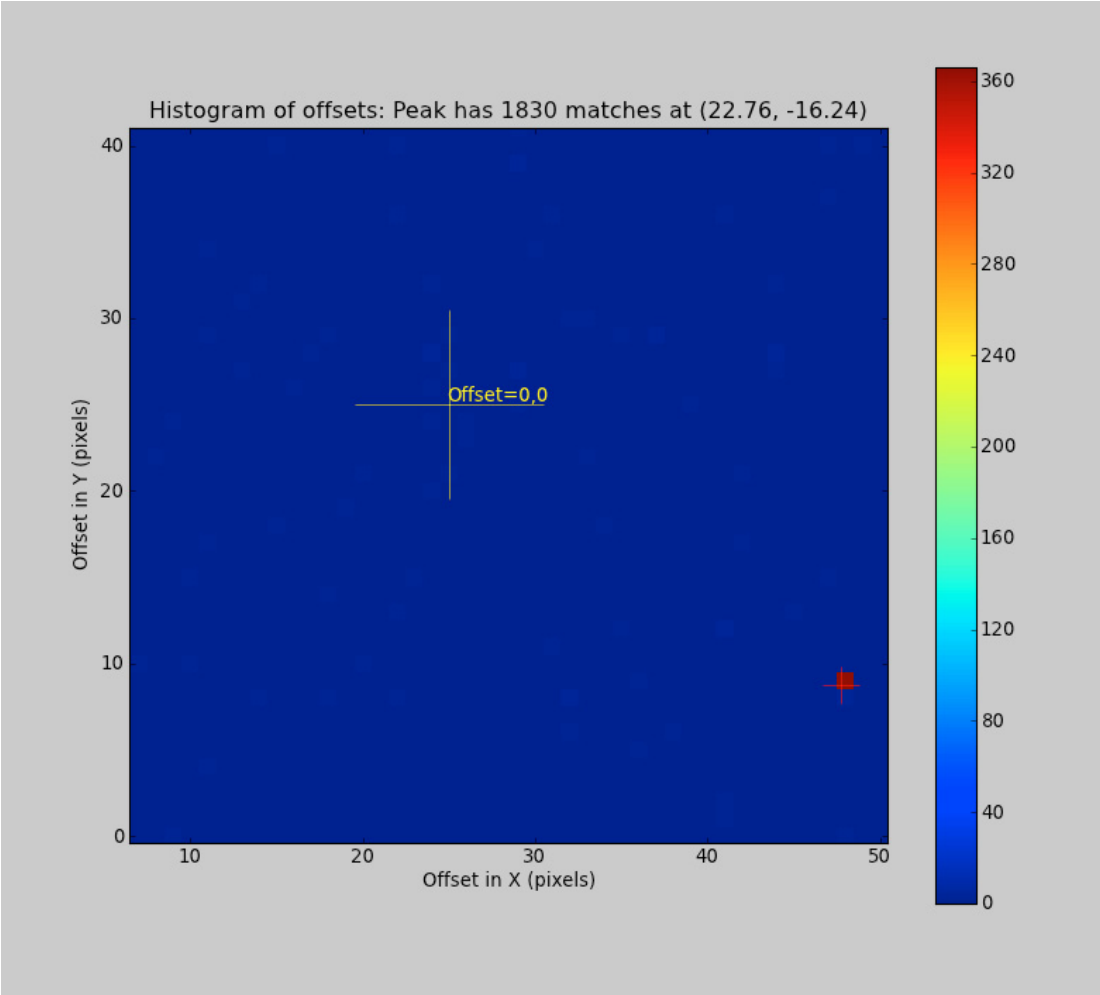
When **tweakreg** is executed, it computes the geometric transformation between the non-reference images and the reference image. In this example, `ibc301qrq_flt.fits` is the first image on the list and is selected as the reference image. Offsets are computed between it and each of the two remaining images in the set, `ibc304v3q_flt.fits` and `ibc307qyq_flt.fits`.

For each image-reference image pair, **tweakreg** will first display the two-dimensional histogram (Figure 7.1) for the initial shift and wait for the user to confirm that the guess is acceptable. After confirmation, it will display the residual plots (Figure 7.2) and the vector plot (Figure 7.3), as well as the fit solution for the first pair of images. The user will be prompted to either proceed to the next set of plots (by hitting the return key), or quit running **tweakreg**. If the user elects to continue, the same plots will be displayed for the next image-reference image pair in the image set.

While using **tweakreg** with different parameter settings to find the optimum fit for image offsets, the parameter ***updatehdr*** is set to ***no***, meaning that the fit solutions computed by **tweakreg** are not applied to the WCS information in the image headers.

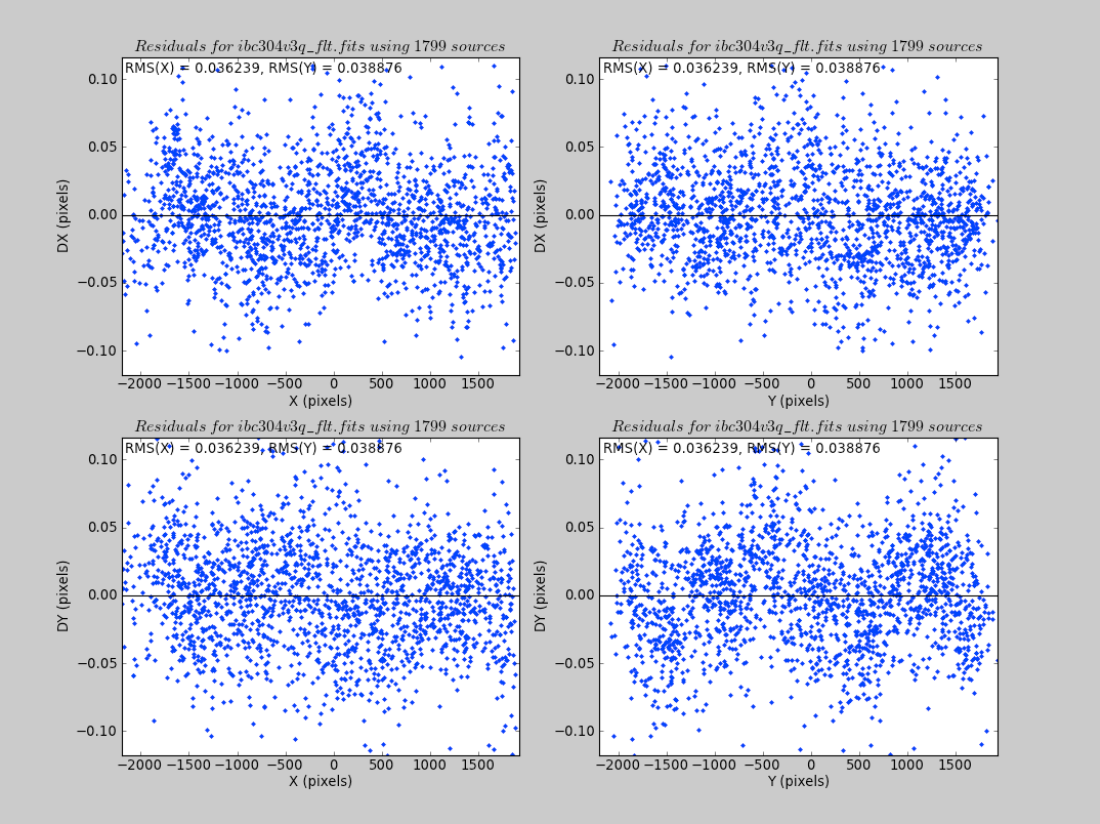
When a good fit is found using parameter values that work best for this example, **tweakreg** gets run one last time with the ***updatehdr*** parameter set to ***yes***. This not only gives the user one last chance to verify that the parameters produce acceptable residuals, but applies the determined fit to the headers of each of the images. The updated headers then get used by **astrodrizzle** in the next step to combine the now properly aligned images.

**Figure 7.1: Two-dimensional Histogram Plot Generated by tweakreg.**



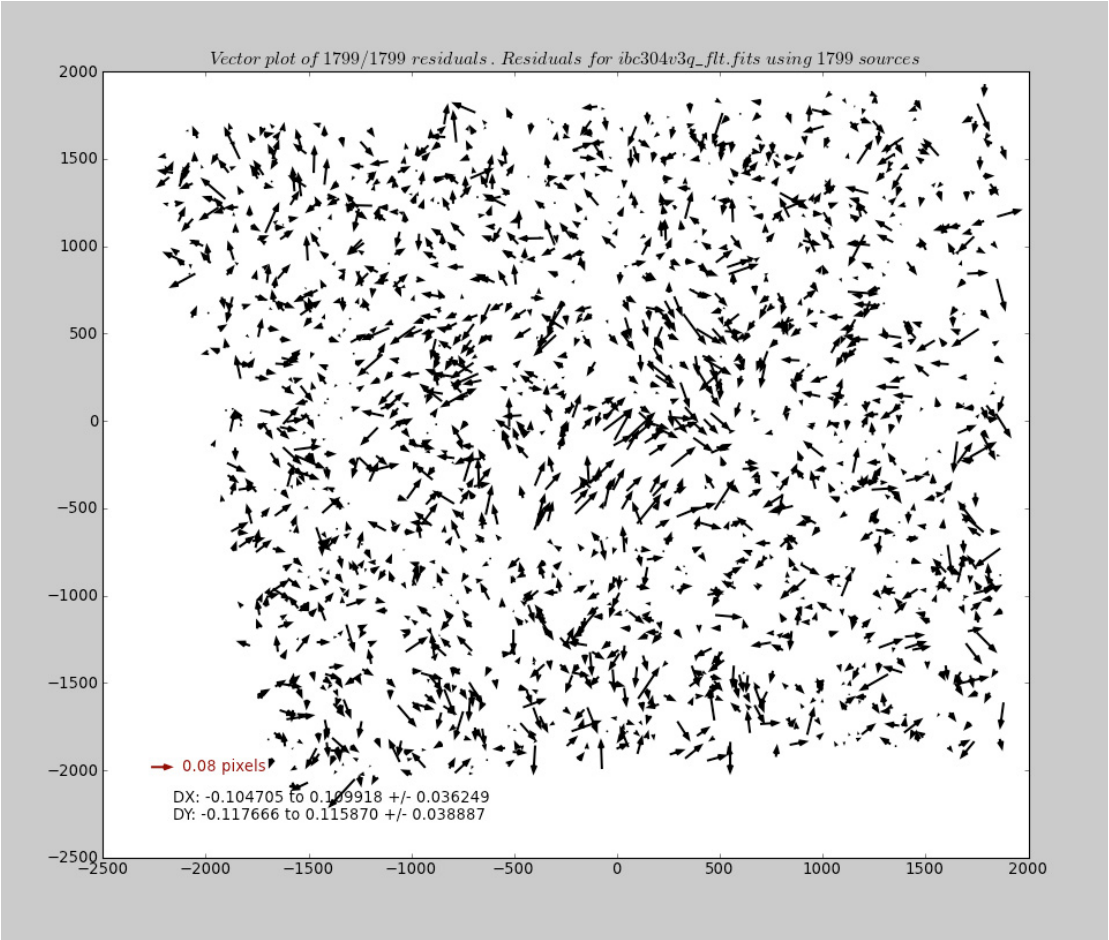

**Two-dimensional histogram generated during the alignment of** `ibc304v3q_flt.fits` **with the reference image** `ibc301qrq_flt.fits`

**Figure 7.2: Residual Plots Generated by tweakreg**




**Residual plots from the alignment of `ibc304v3q_flt.fits` with reference image `ibc301qrq_flt.fits`. Wavy patterns seen in these plots are caused by lithography patterns (artifacts of the CCD chip manufacturing process) of the detector. The detector lithography patterns are not yet fully accounted for in the distortion correction solution. This demonstrates that the fit solution computed by tweakreg is so good that it is actually being limited not by any factor associated with tweakreg itself, but by the accuracy of the distortion solution for the detector.**

**Figure 7.3: Vector Plot Generated by tweakreg**




**Vector plot generated during the alignment of `ibc304v3q_flt.fits` with reference image `ibc301qrq_flt.fits`.**

### 7.1.3 Combining the Images using AstroDrizzle

Now that the images are aligned to a common WCS, they are ready for combination with AstroDrizzle. In a nutshell, AstroDrizzle processes a set of pipeline standard calibrated files (with suffix `flt.fits`), and also CTE-corrected calibrated files (with suffix `flc.fits`). CTE-calibrated files are currently available for ACS, and will be eventually available for WFC3 data as well.

The **astrodrizzle** steps are summarized below. A more detailed description is available at Section 4.2.

- Each input image is aligned, based on WCS information in its header, to a reference image.
- A static pixel mask is created to flag bad detector pixels.
- Sky subtraction is performed on working images (not the original images)

- Each image is individually drizzled, with geometric distortion corrections, to a common reference frame.
- The distortion-free drizzled images are combined to create a median image.
- The median image is blotted, or reverse-drizzled, back to the frame of each input image.
- By comparing each `flt.fits` image with its counterpart blotted median image, the software locates bad pixels in each of the original frames and creates bad pixel masks (typically cosmic rays and faulty pixels in the detector).
- In the final step, `flt.fits` images with updated WCS information from **tweakreg** are drizzled together, using the mask files, onto a single output image.

Running **astrodrizzle** using default parameter values is not recommended, as these defaults may not provide optimal science products. Users are strongly advised to inspect the quality of the sky subtraction and cosmic ray rejection, and to experiment with the output ***final_scale*** and ***final_pixfrac*** parameters in the final drizzle step. Information on evaluating the final products is available in Chapter 8.

The following sections will walk the user through each of the seven AstroDrizzle processing steps.

Note that the software may be run interactively by

- manually changing parameters in the TEAL GUI
- loading a custom configuration file (with suffix `.cfg`) into the task from TEAL or the command-line interface
- using command-line syntax executed in PyRAF or Python (as shown below)

The entire example may be reproduced with the three commands listed below. Since the task has been reset to default values, or "unlearned," it is only necessary to specify non-defaults parameters from the command-line.

```
--> from drizzlepac import astrodrizzle
--> unlearn astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits',output='f606w',\
driz_sep_bits='64,32',driz_cr_corr=yes,final_bits='64,32')
```

Here's a detailed description of what's done by executing the commands:

### Creating the Static Mask

When the **astrodrizzle** parameter ***static*** is set to ***yes*** or ***true*** (the default setting): for each input image, the software calculates the RMS value for each chip, and identifies pixels that are below the median value by more than the "***static_sig*** value times the RMS". This step is used for identifying pixels that may have high values in the dark frame but not in the corresponding image pixels. When the science image gets subtracted by the dark during calibration, this results in anomalously large negative values in the image. Such pixels are not always flagged in the `flt.fits` DQ array, so this step allows them to be identified. Sometimes, such pixels fall on bright objects

so instead of being negative, they would be positive but lower than surrounding pixels. If the images are dithered, those abnormal pixels should land on blank sky at least some of the time, in which case they will appear negative and will be flagged.

### Removing the Sky

When ***skysub*** is set to ***ye***s or ***True***, **astrodrizzle** will subtract the sky from each drizzled exposure. Sky subtraction is recommended for effective cosmic ray flagging and removal, but only if sufficient blank sky is available to perform an accurate determination. For more details, refer to Section 6.3.1.

Many astronomical fields of view cover parts of the sky devoid of any large objects. As a result, the default **astrodrizzle** sky subtraction parameters are usually sufficient, as was the case for the data in this example.

AstroDrizzle computes the sky background independently for each chip and the lowest value is taken to represent the true sky value for both chips. This value is then written to the header keyword `MDRIZSKY` in the `flt.fits` image header in both science extensions (`[sci,1]` and `[sci,2]`). Note that the sky is not subtracted from the science array pixel values in the `flt.fits` image, only in the drizzled science products.

### Separately Drizzling Each Input Image onto a Common Output Frame

When ***driz_separate*** is set to ***ye***s or ***True***, each input image is drizzled to create a single-drizzled science image with suffix `single_sci.fits` and a corresponding weight image with suffix `single_wht.fits` for each input frame. These output images are used in the creation of the image median in the next step, and for the flagging of cosmic rays in subsequent steps.

An important parameter to consider in this step is ***driz_sep_bits***. Data quality flags in the `flt.fits` file, which were set during calibration, can be used as bit masks when drizzling, and the user may specify which bit values should actually be considered "good" and included in image combination. In **astrodrizzle**, this parameter may be given as the sum of those DQ flags or as a comma-separated list, as shown in this example. In this example, ***driz_sep_bits*** setting of "***64,32***" is recommended, so that warm pixels, and CTE tails in the superdark reference file, are treated as valid input pixels.

### Creating the Median image

When ***median*** is ***yes***, **astrodrizzle** creates a median image from the sky-subtracted, separately-drizzled input images, allowing a variety of combination and rejection schemes. In this example, the default image combination method, ***combine_type***, is set to ***minmed***. For this computation, each pixel in the output combined image will either be the median or the minimum of the input pixel values, depending on whether the median is above the minimum by more than *n* times ***sigma***. An estimate of the "true" counts is obtained from the median image while the minimum is actually the minimum unmasked ("good") pixel. This algorithm is designed to perform optimally in the case of combining only a few images (three or four), where triple-incidence cosmic rays often pose a serious problem for more simplified median-combination strategies. The ***minmed*** algorithm performs the following steps:

1. Create a median image, rejecting the highest pixel and applying masks.
2. Use this median to estimate the true counts, and thus derive an RMS.

3. If the median is above the lowest pixel value by less than the first value in ***combine_nsigma***, then use the median value, otherwise use the lowest value.
4. If ***combine_grow*** > 0, repeat the three steps above for all pixels around those that have already been chosen as the minimum, this time using a lower significance threshold as specified as the second value in ***combine_nsigma***.

The last step is very successful at flagging the lower signal-to-noise "halos" around bright cosmic rays which were flagged in the first pass.

When the parameter ***median_newmasks=yes*** or ***True*** (default value), the single-drizzled weight maps (with suffix `single_wht.fits`) are used to create pixel masks for each image based on the pixel flag information present in the `flt.fits` DQ arrays. These masks are used when creating the median image to prevent bad pixels from adversely affecting the median.

Selecting the best parameters for the median step can be an iterative process and should always involve examination of the clean, combined median image to verify its quality:

- The majority of cosmic rays and other artifacts should have been successfully removed
- The width and shape of PSFs should be inspected over the field of view to confirm the quality of the image registration. A PSF which is "round" and "narrow" is a sign that the alignment was successful
- To confirm that no residual offsets or rotations remain, the positions of stars and the geometric transformations may be examined by running **tweakreg** on the separately-drizzled images.

The median combination may require some fine-tuning of the rejection parameters to produce a clean result. To avoid overwriting the median image when fine-tuning the parameter values, the user may wish to copy this product to a unique filename before rerunning AstroDrizzle.

### Computing Cosmic Ray Masks

For optimal cosmic ray flagging, sky subtraction should be performed when the background is more than a few electrons.

When ***blot=yes*** or ***True*** (default value), the median image is transformed back ("reverse drizzled") to the reference frame of each original input image. This is done by backing out the shifts and rotations which were applied in the "single drizzle" step, and by applying inverse distortion corrections. The median image is resampled to the pixel scale of the original images and is trimmed to match the dimensions of each input image. The blotted frames, created for each chip, are named `_sci?_blt.fits` ("?" refers to the extension number). If input images are blinked with their blotted counterparts, the blotted images should align perfectly with their respective input images and should be similar in appearance, except for being cleaned of cosmic rays and other defects.

When ***driz_cr=yes*** or ***True*** (default value), **astrodrizzle** uses the original input images, the blotted median image, and the *derivative* of the blotted image (created using the *deriv* algorithm) to create a cosmic ray mask for each input image.

The deriv algorithm uses the blotted median image to compute the absolute value of the difference between each pixel and its four surrounding neighbors; for each

pixel, the largest of these four values is then used by the *driz_cr* algorithm to flag cosmic rays and other blemishes, such as satellite trails. Where the difference is larger than can be explained by noise statistics, the flattening effect of taking the median, or an error in the shift (the latter two effects are estimated using the image derivative), the suspect pixel is masked.

Cosmic rays are flagged using the following rule:

$$|data\_image - blotted\_image| > scale \times deriv + SNR \times noise$$

Here, *scale* is the multiplicative factor applied to the derivative, *deriv*. This expression is used to determine if the difference between the data image and the blotted image is large enough to require masking. *noise* is calculated using a combination of the detector read noise and the poisson noise of the blotted median image, plus the sky background.

The user must specify two cut-off signal-to-noise (*SNR*) values for determining whether a pixel should be masked: the first for detecting the primary cosmic ray, and the second for masking lower-level bad pixels adjacent to those found in the first pass. Since cosmic rays often extend across several pixels, the adjacent pixels make use of a slightly lower *SNR* threshold. If desired, a third-pass cosmic ray rejection can be carried out by "growing" the cosmic rays via the ***driz_cr_grow*** parameter.

Images obtained from the Archive have already been run through AstroDrizzle using a set of default parameters. For most[2] detectors, pixels identified as cosmic rays during pipeline processing are flagged with the value of 4096 in the data quality (DQ) array of the `flt.fits` images. Because pipeline parameter settings may not provide optimal flagging for every dataset, users are encouraged to fine-tune the cosmic ray rejection parameters ***driz_cr_snr*** and ***driz_cr_scale***.

When **astrodrizzle** is run, cosmic rays flagged during pipeline processing or previous **astrodrizzle** runs are reset in the `flt.fits` DQ arrays via the parameter ***resetbits=4096*** (default value) which changes the bit value 4096 to 0. That way, cosmic ray detection can be performed from scratch by **astrodrizzle**.

When ***driz_cr_corr*** is set to ***yes*** or ***True***, the task will create both a cosmic ray mask image (suffix `sci?_crmask.fits` where "?" is the extension number) and a clean version of the original input images (suffix `crclean.fits`), where flagged pixels are replaced by pixels from the blotted median.

It is strongly recommended that the quality of the cosmic ray masks be verified by blinking the original `flt.fits` input image with both the cosmic ray-cleaned image (`cr_clean.fits`) and the cosmic-ray mask (`crmask.fits`) as shown below.

```
disp ibc301qrq_flt.fits[sci,1] 1 zs- zr- z1=0 z2=250
disp ibc301qrq_crclean.fits[sci,1] 2 zs- zr- z1=0 z2=250
disp ibc301qrq_sci1_crmask.fits 3 zs- zr- z1=0 z2=1
```

Figure 7.4 shows the original input image `ibc301qrq_flt.fits[sci,1]` with its corresponding cosmic ray mask in Figure 7.5.

2. For WFC3/IR images, the flag value for cosmic rays identified by AstroDrizzle are 8192, to distinguish them from cosmic rays detected in **calwfc3** with flag 4096.

**Figure 7.4: Chip 2 of Input Image `ibc301qrq_flt.fits`**

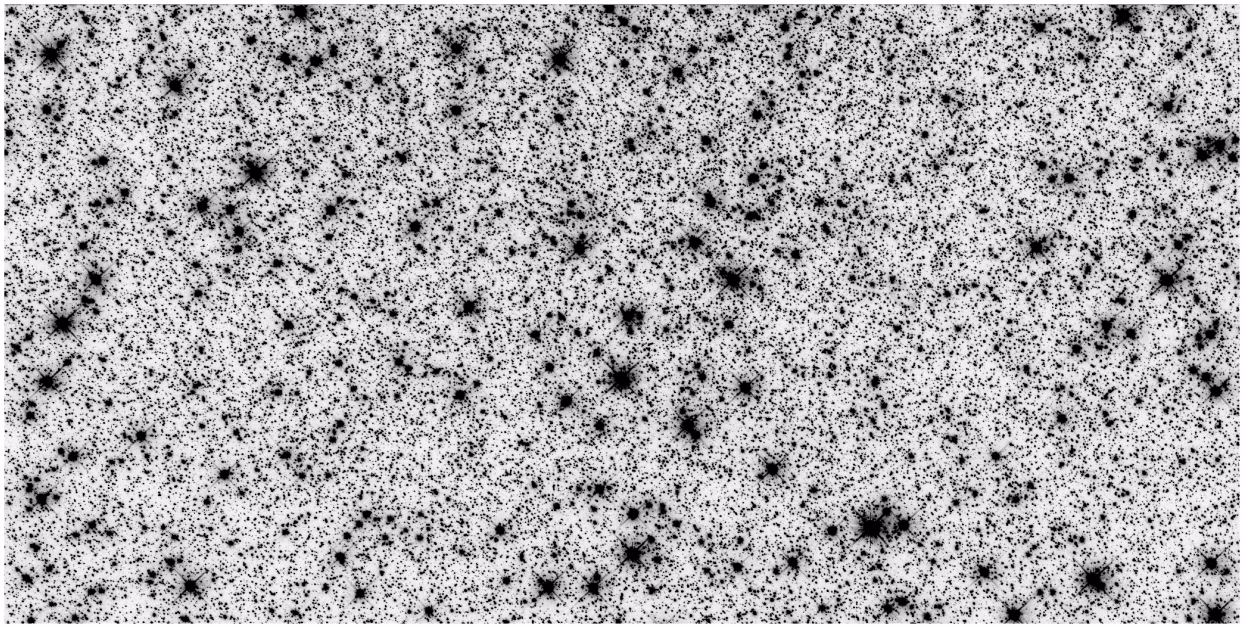
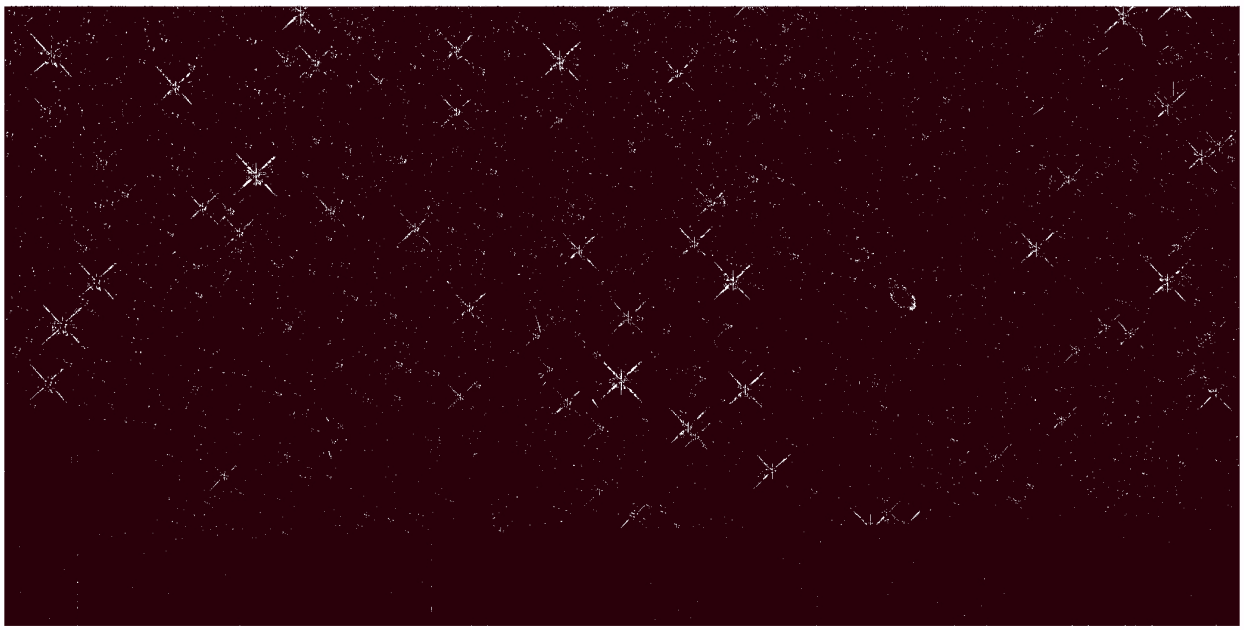


**Figure 7.5: The Cosmic Ray Mask for `ibc301qrq_flt.fits[1]` (Chip 2)**



**A mask value of 1 (shown in black) corresponds to good pixels, while a mask value of 0 (white) corresponds to pixels flagged as cosmic rays. The lack of flagging in the lower portion of the figure is because there were no overlapping images in that area.**

### Final Drizzle Combination

When ***driz_combine*=*yes*** or ***True*** (default), **astrodrizzle** takes the original input images, together with the final masks, and drizzles them onto a single output image. The final masks are a combination of the static masks, the cosmic ray masks, and the DQ values assigned as "bad" by the user via the ***final_bits*** parameter.

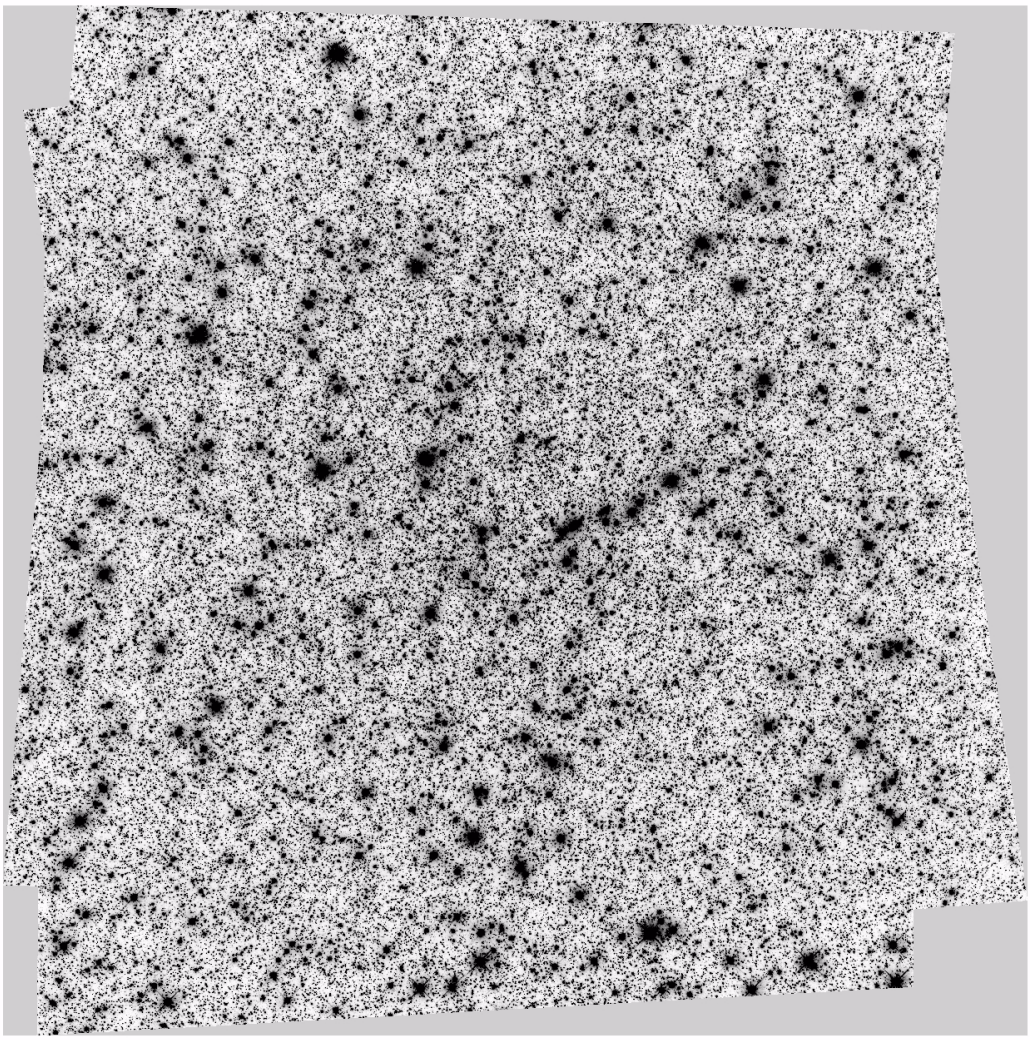
By default, UVIS images are drizzled to a scale of 0.0396 arcseconds/pixel, but the user is encouraged to experiment with other options (e.g., shrinking the ***final_scale*** and ***final_pixfrac*** value to yield a sharper output PSF.) For more information, refer to Section 6.3.3 and the example at Section 7.2.

In this example, the final image is weighted by the exposure time of each image, so ***final_wht_type=EXP***. This weighting is a good approximation in the regime where the noise is dominated by photon counts from the sources, while the contributions from sky background, read noise, and dark current are negligible. For a discussion about photometry and weights in AstroDrizzle, please refer to Section 4.2.12.

When the parameter ***build=yes*** or ***True*** (a non-default value), AstroDrizzle generates a single multi-extension FITS file named `<output>_drz.fits`, where the ***output*** parameter defines the string to name the final product. For this example, the image is named `f606w_drz.fits` (so ***output*='*f606w*'**).

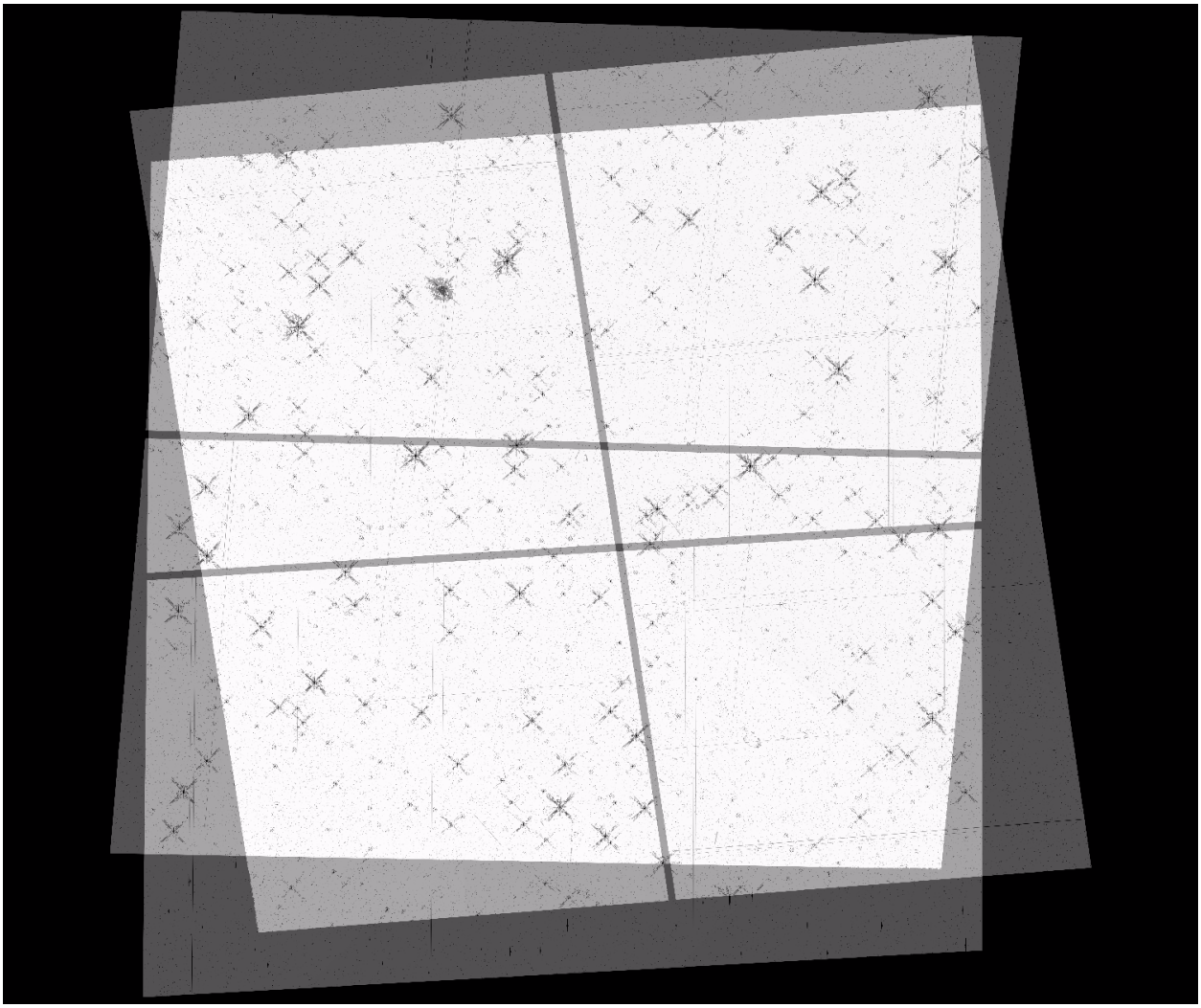
The first extension of the drizzled product, shown in Figure 7.6, represents the WCS-aligned combination of science extensions of the three input images. Each pixel now covers an equal area on the sky and has an equal photometric normalization across the field of view. Thus, the drizzled science image is photometrically and astrometrically accurate for point sources and extended sources. The dimensions of the output image are computed on-the-fly by AstroDrizzle. The user also has the option to define a custom size and orientation for the final output image, by setting parameter values in the ***final_wcs*** section of the **astrodrizzle** parameters.

**Figure 7.6: Extension #1 of `f606w_drz.fits`, the Final Combined Science Image**



The second extension of the drizzled image contains the weight image (Figure 7.7). When ***final_wht_type*** is set to ***EXP***, the weight image can be considered an effective exposure time map. The weight image, in Figure 7.7, shows lighter areas that have higher weight. The chip gaps are clearly visible, as are column defects and cosmic ray features. The majority of the image is white, with a value of ~120 which is the combined exposure time of the three 40-second exposures. Regions that have less than a full contribution from all three observations are colored various shades of gray. Examples of this can been seen around the edges of the weight image, where there was little image overlap, in the three intersecting chip gaps, and in the saturated cores and diffraction spikes surrounding bright stars.

**Figure 7.7: Extension #2 of `f606w_drz.fits`, the Final Weight Image**



The third extension of the drizzled image (shown in Figure 7.8) contains the *context* information. Here, each input image chip is identified by a bit (power of 2) in a 32-bit integer. For instance, image 1/chip 1 = $2^0$ (1), image 1/chip 2 = $2^1$ (2), image 2/chip 1 = $2^2$ (4), image 2/chip 2 = $2^3$ (8), image 3/chip 1 = $2^4$ (16), image 3/chip 2 =$2^5$ (32), and so on[3]. Each context pixel value is an additive combination of these chip bits, depending on which images contributed to the corresponding pixel in the drizzle-combined image. For example, if the pixel (200,100) in the drizzle-combined image has contributions from image 1 / chip 1, image 2/chip1, and image 3/chip 2, the value for pixel (200,100) in the context image would be the sum of the image bit values, 1+4+32=37.

3. Should more than 32 chips be combined into a final output image, this context array will be "expanded" by creating another 32-bit integer array of the same size, layered on top of the original to create a 3D array with 2 layers. The new layer would then be used to keep track of the next 32 chips, with additional layers being created for more sets of 32 chips, as needed.

For optimal cosmic ray rejection, the drizzled science product in this example has been sky-subtracted. Some photometry packages, however, require that the sky be added back in to the final product prior to analysis. For example, the DAOPHOT Reference Guide recommends that

> "the sky background should NOT be subtracted from imaging prior to photometry. DAOPHOT fitting routines use an optimal weighting scheme which depends on the read noise, gain, and true counts in each pixel. If the mean sky has been subtracted, the computed weights will be incorrect."

Therefore, the user can add the average sky value that was determined from the individual input frames. This average sky can be determined by querying the `MDRIZSKY` keyword in the *merged header table extension* `HDRTAB`, described in Section 4.4.6. Additionally, **daophot** assumes that the final image is in units of electrons, so if the image was drizzled with the parameter ***final_units*** set to ***cps***, it may be necessary to convert the image back to units of electrons by multiplying the image by the exposure time prior to performing photometry.

**Figure 7.8: Extension #3 of f606w_drz.fits, the Final Context Image**