Aligning and Drizzling WFPC2 Images

Introduction

The three Wide Field channels (WF2, 3, and 4) in WFPC2 produce images that are strongly undersampled. But spatial resolution could be improved by combining sub-pixel dithered images using AstroDrizzle. In this example, images of Messier 2, obtained by WFPC2 using a 4-pt dither box, followed by an offset with a dither line, are used to demonstrate WFPC2 image processing using TweakReg and AstroDrizzle for image alignment and image combination.

Towards the end of WFPC2's operation, images were recalibrated with the best-available reference files, then drizzled using the final version of PyDrizzle. These files are available as raw, calibrated, and drizzled products from the Archive (but are not processed "on-the-fly"). The drizzled images are simply mosaics of each `c0m.fits` file, for use as a "quick-look" images, that have been corrected for geometric distortion and have a plate scale of 0.1"/pixel.

Since WFPC2 images were archived in late 2009, HST FITS file structures have changed significantly to hold more astrometric information. This new FITS format is required for running AstroDrizzle; image headers now contain keywords that hold values for linear distortion corrections in the CD-Matrix, and values of coefficients for higher order polynomial distortion corrections. These distortion corrections are directly implemented in the WCS using the Simple Image Polynomial (SIP) convention. Other distortion information, such as non-polynomial and detector distortions, are stored as FITS extensions. Additional information is available in Chapter 3 of the DrizzlePac Handbook.

To process WFPC2 images using DrizzlePac tasks, the FITS images need to be made compatible with the software. This conversion is done when the images are run for the first time[1] in TweakReg or AstroDrizzle with the parameter ***updatewcs=True***.

This is the FITS file structure of an image fresh from the Archive.

```
--> catfits ua060502m_c0m.fits
EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT

0     ua060502m_c0m ua060502m_c0f.fits                       16
1       IMAGE       SCI                   1     800x800      -32
2       IMAGE       SCI                   2     800x800      -32
3       IMAGE       SCI                   3     800x800      -32
4       IMAGE       SCI                   4     800x800      -32
```

After running it through AstroDrizzle or TweakReg, the FITS file structure has changed. The D2IMARR extension contains the 34th row pixel width corrections. WCSCORR is a table containing all updates made to the image's WCS keywords values.

```
--> catfits ua060502m_c0m.fits
EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT

0     ua060502m_c0m ua060502m_c0f.fits                       16
1       IMAGE       SCI                   1     800x800      -32
2       IMAGE       SCI                   2     800x800      -32
3       IMAGE       SCI                   3     800x800      -32
4       IMAGE       SCI                   4     800x800      -32
5       IMAGE       D2IMARR               1     800x1        -32
6       BINTABLE    WCSCORR               1     24Fx28R
```

[1] The DrizzlePac task 'updateWCS' (not the same as the ***updatewcs*** parameter in AstroDrizzle and TweakReg), which is used for updating WCS information in FITS files, is currently (as of November 2013) not compatible with WFPC2 data. The WFPC2 FITS file structure for images fresh from the Archive can only be changed to the new FITS format using AstroDrizzle or TweakReg with ***updatewcs=True*** because they require extra processing steps that are not included in the task 'updateWCS.' The conversion from the old FITS format to new FITS format for WFPC2 images only needs to be done once for as long as that data is used.

A few notes about WFPC2 images:

I. When WFPC2 data from the Archive are processed in AstroDrizzle or TweakReg for the *first time*, set the parameter ***updatewcs*** to ***True***. With this setting, the software will configure WFPC2 data to the new FITS structure required for running DrizzlePac tasks. (Reminder: the reference file pointer "`uref`" should be set in the working C-shell window before starting PyRAF or Python;
i.e., `setenv uref /Users/jane/reffiles/`
since AstroDrizzle or TweakReg need to access the appropriate reference files when ***updatewcs=True***.)

The following changes are made:

i. AstroDrizzle reads the DGEOFILE[2] reference file that contains information about the 34th row pixel width corrections. The row width corrections are extracted and placed in a new reference file, named in the header keyword D2IMFILE. Then, corrections in the D2IMFILE reference file (with suffix `d2i.fits`) are inserted in the calibrated image (`c0m.fits`) as a FITS extension.

ii. The IDCTAB header keyword holds the name of the IDCTAB reference file (Instrument Distortion Coefficients TABle, with suffix `idc.fits`) that contains values for linear distortion corrections and polynomial distortion correction coefficients. This information is used to update CD-MATRIX and SIP coefficient values stored as image header keywords.

iii. The OFFTAB header keyword holds the name of a reference file containing time-dependent chip offsets. This information is extracted from the reference file and stored as keyword values in the image headers.

II. When all chips in WFPC2 are drizzle-combined, a 4-chip mosaic image is created. You'll notice that the PC section appears darker than the WF images. That's because the sky level is set by adopting the lowest sky value among the four chips, in units of arcseconds square. Since the PC1 FOV is a smaller area of sky projected on an 800x800 chip, background counts per pixel in the PC image are lower.

III. By default, AstroDrizzle and TweakReg adopt the plate scale of the first group in an image for subsequent processing. For WFPC2, the PC is the first chip handled by these tasks, and it has a plate scale 0.045"/pixel. If you prefer to drizzle the final image to the WF scale (0.1"/pixel), this has to be specified in AstroDrizzle with the parameters ***driz_sep_wcs=True, driz_sep_scale=0.1, final_wcs=True, and final_scale=0.1***.

IV. The data quality information for calibrated WFPC2 images are stored in a separate FITS file, with suffix `c1m.fits` or `c1f.fits`. These data quality files must be present in the working directory when WFPC2 image data (`c0m.fits`/`c0f.fits`)[3] are processed by AstroDrizzle.

V. The Archive only provides single-image drizzled WFPC2 data processed by PyDrizzle because WFPC2 does not have any associations and is not part of the OTFR system. Therefore, unlike ACS and WFC3/UVIS, WFPC2 data quality files (`c1m.fits`/`c1f.fits`) from the Archive do not have the cosmic ray bit flag (4096). However, when re-using WFPC2 input images (`c0m.fits`/`c1m.fits`) and their data quality files (`c1m.fits`/`c1f.fits`) that had previously been used in manual AstroDrizzle processing, cosmic ray DQ bits identified in that AstroDrizzle run are now stored in the `c1m.fits` data quality file. This does not require any change to the default cosmic ray parameters in AstroDrizzle, it's just something to know about the data.

VI. In the DrizzlePac handbook and online FAQs, users who wish to reprocess their data using AstroDrizzle are encouraged to use parameter values in the MDRIZTAB reference file as a starting point; these are the same parameter values used in the pipeline. WFPC2, however, does not have this reference file because it's not processed in OTFR. The parameters in this example may serve as a starting point for WFPC2 data, but keep in mind that every dataset is different and may require alternate settings.

VII. In raw images, saturation occurs at 4096 DN. However, in calibrated images, bias subtraction and flat-fielding

---

[2] For some other instruments, the DGEOFILE reference file used to also hold residual distortion corrections that are filter-dependent. In the new processing system, those residual distortion corrections are now placed in the NPOLFILE reference file (`npl.fits`). WFPC2, however, does not have any residual distortion corrections.

[3] `c0m.fits` are multi-extension FITS files, while `c0f.fits` are waivered FITS files. Both can be used as input to AstroDrizzle. Waivered FITS input files will be converted to multi-extension FITS files for further processing by the software.

reduces the saturation level value. In general, for calibrated data (`c0m.fits` or `c0f.fits`), pixel values above 3500 DN should be considered as saturated pixels and treated with caution.

VIII. The units for calibrated WFPC2 images (`c0m.fits`) is DN, not electrons. But the drizzled image is in electrons/sec, converted using the gain value of the PC. This will have some effect on photometric accuracy (see IX for more details).

IX. For best results, only perform point source photometric measurements on single-chip drizzled images or non-drizzled calibrated (`c0m.fits`) images multiplied by the pixel area map. In addition to the gain issue mentioned in (VIII), single-chip drizzled images are necessary because each chip has a slightly different detector sensitivity (PHOTFLAM value). In the drizzle-mosaiced image, only one sensitivity value is used, the average PHOTFLAM values of the four chips. Drizzle-mosaiced images, however, provide the best astrometric corrections for WFPC2 data, especially when images have to be realigned using TweakReg, as sources in all four chips are used to recalculate the WCS correction.

Summary of Steps

This example illustrates the use of AstroDrizzle and TweakReg on WFPC2 data, and are broken into the following steps:

1. A description of the data.
2. Run AstroDrizzle on WFPC2 images, using image header WCSs to align the images, and inspect data products.
3. Can image offsets be improved using TweakReg?
4. Transfer the new WCS, based on improved image offsets, to the calibrated images.
5. Re-run AstroDrizzle using images with updated WCSs, and try different ***final_scale*** and ***final_pixfrac*** settings.

1. A Description of Data

The target, globular cluster MESSIER 2 was observed on Aug 1, 2007, in two parts: (1) a 4-point dither; (2) a POS TARG shift with a 2-point dither. Six images were produced from these observations. The IRAF task **hselect** shows several useful keyword values: visit and line number, date and time (in UT) of the exposures, followed by RA and Dec at the reference pixel for the PC.

```
--> hselect *c0m.fits[1]  $I,linenum,date-obs,time-obs,crval1,crval2 yes
ua060502m_c0m.fits[1]  05.002  2007-08-01  13:51:16  323.3147138686913  -0.8710668436652544
ua060504m_c0m.fits[1]  05.002  2007-08-01  13:59:16  323.3149432083502  -0.8704559656029925
ua060506m_c0m.fits[1]  05.002  2007-08-01  14:07:16  323.3154170817679  -0.8700073543893925
ua060508m_c0m.fits[1]  05.002  2007-08-01  14:15:16  323.3151877409566  -0.8706182316527187
ua06050am_c0m.fits[1]  05.004  2007-08-01  14:23:16  323.3147625476145  -0.8709934880785906
ua06050cm_c0m.fits[1]  05.004  2007-08-01  14:31:16  323.3149918871733  -0.8703826100133287
```

Observations were taken in the F814W filter, each with a 10 second exposure time. This data was obtained from proposal 11100, Visit 5, lines 2 and 4. Three excerpts from the Phase II proposal are shown below. The first one shows how the exposures were defined in the exposure logsheet. The next two excerpts show how the dither patterns were specified in the Phase II proposal.

Excerpt from the Phase II proposal's exposure log:

```
Visit: 05
2  MESSIER-002  WFPC2  IMAGE  WFALL  F814W  CR-SPLIT=NO  1  10 S   PATTERN 1 1-2
4  MESSIER-002  WFPC2  IMAGE  WFALL  F814W  CR-SPLIT=NO  1  10 S   POS TARG 0.2241, 0.2241;
                                                                    PATTERN 2 3-4
```

(Exposures 1 and 3, referenced in the `PATTERN` special requirements, are not shown here.)

Excerpts from the Phase II proposal's dither pattern specification section:

```
Pattern_Number: 1
                     Primary_Pattern                   Secondary_Pattern
Pattern_Type         BOX
Pattern_Purpose      DITHER
Number_Of_Points     4
Point_Spacing        2.349
Line_Spacing         2.349
Coordinate_Frame     POS-TARG
Pattern_Orient       32.01
Angle_Between_Sides 154.01
Center_Pattern       NO
Pattern_Comments:     Big diamond, pattern 17

Pattern_Number: 2
                     Primary_Pattern                   Secondary_Pattern
Pattern_Type         LINE
Pattern_Purpose      DITHER
Number_Of_Points     2
Point_Spacing        2.349
Line_Spacing         <none>
Coordinate_Frame     POS-TARG
Pattern_Orient       32.01
Angle_Between_Sides <none>
Center_Pattern       NO
Pattern_Comments:     Half of big diamond
```

These datasets files were retrieved from the Archive. Unlike other instruments, WFPC2 image data and data quality arrays are in two separate files, with suffix `c0m.fits` and `c1m.fits`, respectively.

ua060502m_c0m.fits	ua060504m_c1m.fits	ua060508m_c0m.fits	ua06050am_c1m.fits
ua060502m_c1m.fits	ua060506m_c0m.fits	ua060508m_c1m.fits	ua06050cm_c0m.fits
ua060504m_c0m.fits	ua060506m_c1m.fits	ua06050am_c0m.fits	ua06050cm_c1m.fits

2. Run AstroDrizzle on WFPC2 images, using image header WCSs to align the images and inspect data products.

For images taken in a single visit, there is usually a high degree of accuracy between offsets specified in the Phase II proposal and actual offsets executed by the telescope. However, occasional drifts may occur due to a failure to acquire one of two guide stars, which introduces a slow roll centered on the single guide star. Therefore, users are encouraged to verify their image alignments using TweakReg (covered in step 3) before running AstroDrizzle.

Drizzle-combining data is usually an iterative process, using different parameter settings to improve the final product. For WFPC2, begin the process by using AstroDrizzle parameter values in this example.

Some parameter values for the first run:

- When input files are specified as `*c0m.fits` (or `*c0f.fits`), this means that all calibrated WFPC2 data in the working directory is processed. Be sure to check that each `c0m.fits/c0f.fits` file has an accompanying `c1m.fits/c1f.fits` data quality file.
- ***updatewcs=True*** is specified whenever WFPC2 data fresh from the Archive is processed for the first time by AstroDrizzle. This parameter reconfigures the old format FITS data to be compatible with DrizzlePac tasks. If the same data is later used in other DrizzlePac tasks, be sure to set ***updatewcs=no*** in those tasks (if the parameter exists) if you want to preserve WCS values that were obtained in a prior TweakReg run.
- ***driz_sep_bits*** and ***final_bits*** specify which data quality values are considered "good" pixels and should therefore be ignored when creating the bad pixel mask. The software treats pixels with a data quality flag of 0 as good pixels. Sometimes, it's necessary to include some non-zero flags as good pixels as well. For WFPC2, ***1024*** is the bit flag for repaired warm pixels -- it should definitely be considered "good." In this example, the bit flag value for saturated pixels, ***8***, is chosen as a "good" pixel (this selection is a user preference). In this case, it's being

used to better evaluate the quality of cosmic ray rejection, by avoiding confusion between cosmic rays and saturation flags in the data quality image display (more about this later in the example). Some users may also elect to retain saturated pixels because, under some circumstances, charge accumulation can remain linear beyond saturation (see Section 5.2.4 of the WFPC2 Data Handbook). For this example, ***driz_sep_bits*** and ***final_bits*** are set to ***'8,1024',*** telling AstroDrizzle to treat these bit values flagged in the data quality file (`c1m.fits`) as good pixels: 8 for saturated pixels, and 1024 for repaired warm pixels. All other non-zero bit flags in the data quality files are treated as bad pixels.

- ***driz_sep_fillval*** and ***final_fillval*** give the fill value for pixels in drizzled images with no data value. Here, it is set to a very high value, like ***99999***, so they stand out prominently as bad pixels. This will make it easier to statistically weed out bad data in the final drizzle-combined image when running automated photometry software like **daophot** and **SExtractor**.
- ***driz_cr_snr='5.5 3.5'*** and ***driz_cr_scale='2.0 1.5'*** are used together to adjust thresholds for cosmic ray rejection.

  If bright sources are flagged as cosmic rays (as was initially the case for this dataset when it was run with default settings), this could be due to (1) improper sky subtraction, (2) misalignment between input frames, and (3) cosmic ray rejection parameters need to be fine-tuned. For this set of data, the first two reasons seemed unlikely, leaving the third option.

  - ***driz_cr_snr*** is the signal-to-noise required for identifying cosmic rays in a pixel, and those adjacent to it. The default value, ***'3.5 3.0',*** was not used here because it was flagging the peaks of bright stars as cosmic rays in the final weight image. When this parameter is set correctly, the final weight image should show detector artifacts and cosmic rays, and none of the science sources (see Figure 1b.). After running several tests with different values of ***driz_cr_snr***, the values ***'5.5 3.5'*** provided satisfactory results in the final weight image. Different datasets have different characteristics, so it's a matter of trying different settings to hone in on the best results.
  - ***driz_cr_scale*** is a "fudge factor" that multiplies the local derivative of the blotted image. This is added to the calculated statistical noise in the pixel to create a new (larger) estimate of the noise, making it less likely that a pixel will be marked as bad. In this example, the values ***'2.0 1.5'*** were used, instead of the default ***'1.2 0.7'***, to provide optimal cosmic ray rejection. (Again, different values were tested and this set of values, along with the ***driz_cr_snr*** values, provided good cosmic ray rejection.) Using larger values for this parameter can also compensate for low levels of misalignments in the images which may be useful for some initial AstroDrizzle applications. But after the images have been realigned using TweakReg, this value should be decreased for subsequent AstroDrizzle run.
- ***driz_cr_corr=True*** creates cosmic ray-free versions of the input `c0m.fits` images (in this example, they've been given the suffix `c0m_crclean.fits`). These are versions of the original input images where bad pixels are replaced by pixels from the blotted median image. These cosmic ray-cleaned images, with suffix `c0m_crclean.fits`, will be later used in TweakReg to identify sources for image alignment. "CR-cleaned" images are not suitable for science analysis because not all artifacts have been removed at this stage.

DrizzlePac commands in this example use Python syntax. Even though they could be run in the Python environment, this example shows them in PyRAF because it handles both Python and IRAF commands. This makes it easier to switch back and forth between Python and IRAF.

Before starting PyRAF, the reference file pointer, `uref`, should be set so AstroDrizzle can access reference files named in the DGEOFILE, IDCTAB, and OFFTAB image header keywords. These reference files can be retrieved from the WFPC2 Reference Files Page, and placed in a reference file directory (i.e., `/Users/jane/reffiles/`).

Running AstroDrizzle

Set a pointer to the location of the reference files using the UNIX `setenv` command in the C-shell window. Then, go to the working directory and start PyRAF.

```
> setenv uref /Users/jane/reffiles/
> cd /Users/jane/mydata/
> pyraf
```

After loading PyRAF, the PyRAF prompt should appear: (`-->`). Next, load the DrizzlePac package, then load AstroDrizzle.

```
--> import drizzlepac
--> from drizzlepac import astrodrizzle
```

Before running AstroDrizzle, make a copy of the `c0m.fits` and `c1m.fits` files that came from the Archive (that have not been processed by AstroDrizzle and TweakReg) in a backup directory. This is just a precaution in case they're needed if you have to start over.

AstroDrizzle is run as a Python command by specifying the input files and parameter values. For WFPC2 images, files with the extension `c0m.fits` are specified as input files, but the data quality files, with suffix `c1m.fits`, must also be present in the same working directory.

Parameters not mentioned in the AstroDrizzle command use default values. However, in the AstroDrizzle command below, default values for some parameters have been included because they're important parameters that are useful for record-keeping purposes. (Note: AstroDrizzle may also be run from Teal[4] using the **epar** command in PyRAF.) To run the the AstroDrizzle[5] command below, simply cut and paste it in the working directory. The backslash is used to continue a command on the next line--when this happens, three dots (...) appear after hitting the return key, indicating that Python is expecting more parameters.

```
astrodrizzle.AstroDrizzle('*c0m.fits', driz_sep_fillval=99999, \
updatewcs='True', driz_sep_bits='8,1024',combine_type='minmed', \
driz_cr_snr='5.5 3.5', driz_cr_scale='2.0 1.5', final_wht_type='EXP', \
final_wt_scl='exptime', final_pixfrac=1.0, final_fillval=99999, \
driz_cr_corr= True', final_bits='8,1024')
```

Typically, AstroDrizzle is run several times to find the best match between parameter values and image quality. For instance, if there are bright sources in the image, blink the final drizzle-combined image with its weight image. Are there any pixels in the center of the stars? That may be due to AstroDrizzle mistaking them for cosmic rays.[6] Therefore, adjustments need to be made, using ***driz_cr_snr*** and ***driz_cr_scale*** to keep the stars intact. For this example, those tests were already done, resulting in the use of the non-default values mentioned earlier.

In this example, `combine_type='minmed'` was used. The "`minmed`" option produces an image that's the same as the median, except for cases where the median is significantly higher than the minimum good pixel value--when that is the case, "`minmed`" chooses the minimum value to be applied to the median image.

There are six images in this set, so AstroDrizzle could have also been run using `combine_type=median`. However, a stack of pixels from images being medianed (the separately-drizzled `single_sci.fits` images) may have bad pixel fill values set by the parameter `driz_sep_fillval=99999`. This could potentially create a bad median value. To avoid this scenario, an upper threshold for clipping input pixel values can be imposed, for instance, `combine_hthresh=90000`, to make sure the "fillval" pixels are not included in the median image.

---

[4] If you prefer to work in the Teal interface that is brought up using the **epar** command (i.e., "**epar astrodrizzle**") in PyRAF, be sure to click on the ***Default*** button at the top right of the Teal window before entering the non-default settings shown in the command line example above.

[5] To get help for AstroDrizzle, simply type one of these commands:

```
--> astrodrizzle.help()
--> help astrodrizzle
```

To save the help information to a text file, called 'adriz_help.txt'

```
--> astrodrizzle.help(file='help.txt')
```

[6] This is one of the reasons why the bit flag ***8***, the saturated pixel flag, was specified as a good pixel in ***driz_sep_bits***. If ***8*** had been removed, it would have appeared as a bad pixel in the mask file, making it difficult to check if the cosmic ray rejection algorithm was flagging those saturated pixels as cosmic rays.

Therefore, AstroDrizzle could also be run using these parameters:

```
astrodrizzle.AstroDrizzle('*c0m.fits', driz_sep_fillval=99999, \
driz_sep_bits='8,1024',combine_type='median', combine_hthresh=90000, \
driz_cr_snr='5.5 3.5', driz_cr_scale= '2.0 1.5', final_wht_type='EXP', \
final_wt_scl='exptime', final_pixfrac=1.0, final_fillval=99999, \
driz_cr_corr= True', updatewcs='True', final_bits='8,1024')
```

By default, the final combined products are written to three files:

- `final_drz_sci.fits` contains the science image
- `final_drz_wht.fits` is the weight image that shows the relative weight of the output pixels, and can be considered an effective exposure time map.
- `final_drz_ctx.fits`, the context extension, is a map of the output images and a record of which images contributed to each pixel.

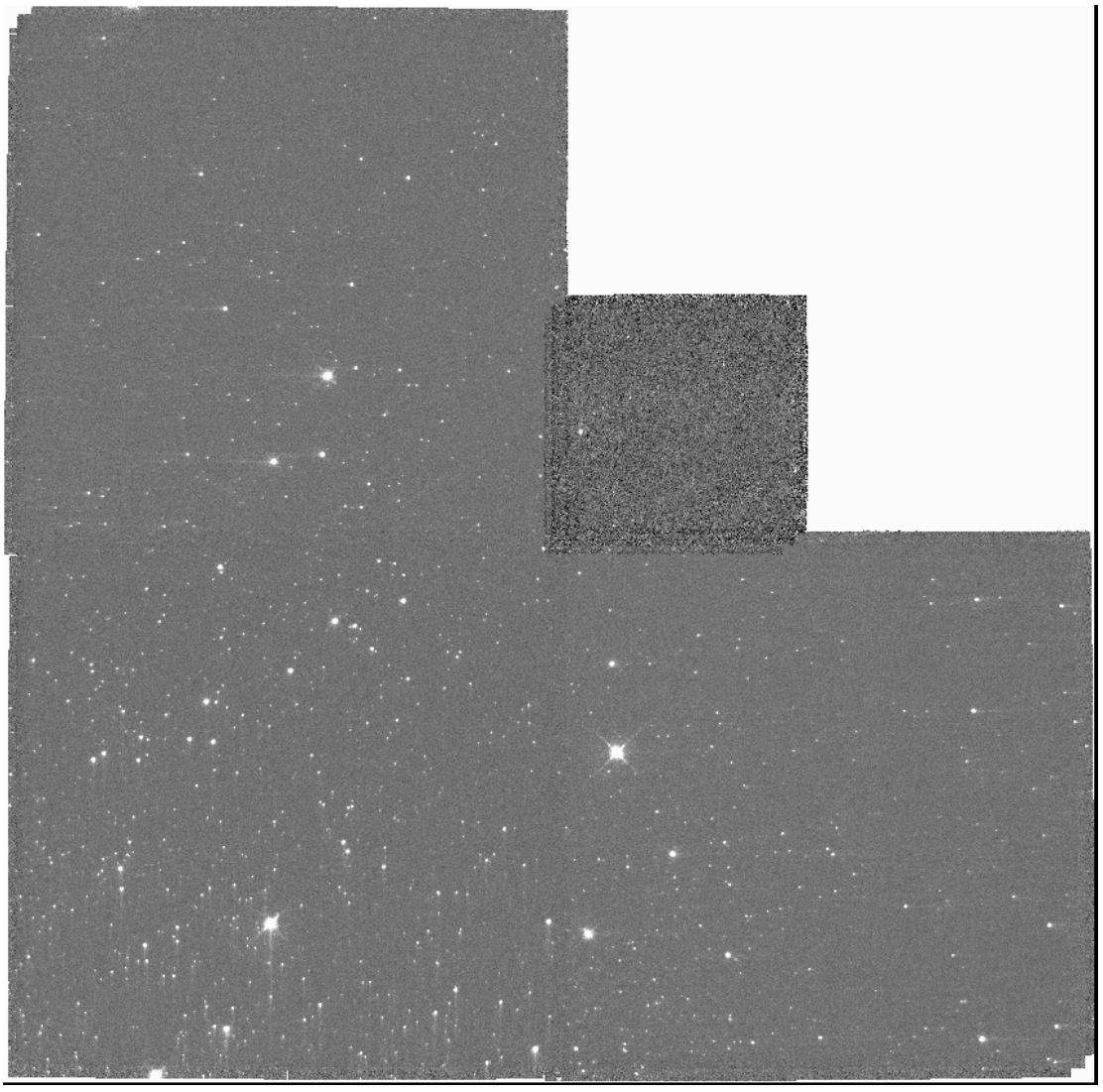


Figure 1a: The drizzle-combined mosaiced WFPC2 chips, in the PC plate scale.

Due to the aberration in the primary beam, light from sources near the pyramid edges is divided between adjacent chips. This effect is difficult to completely remove with just a few dither steps. The PC1 edges are further affected by the background level differences due to its larger focal length.

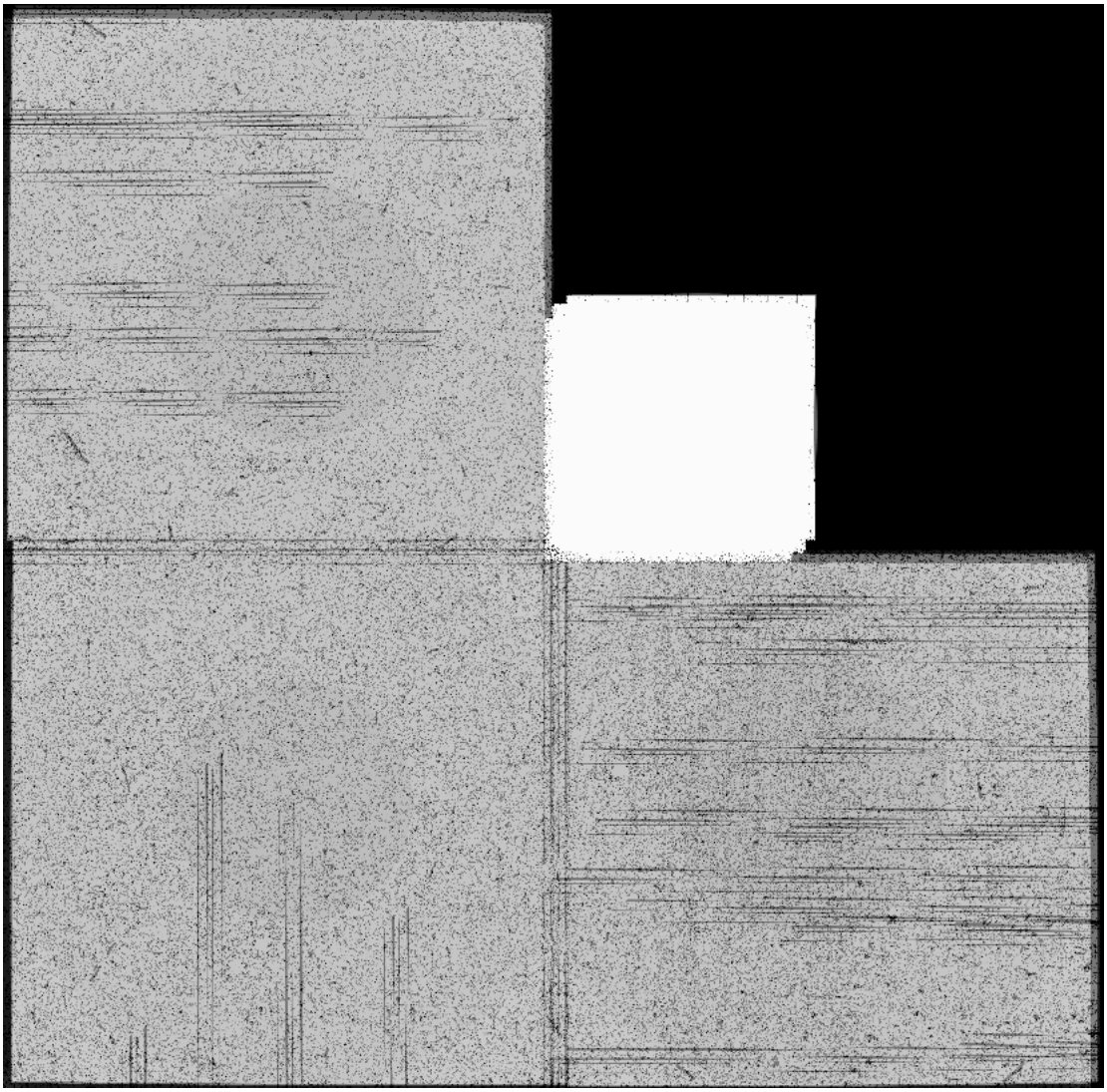


Figure 1b: The weight image, stretched to show details in the WF chips.

The dark features represent weighting from bad pixels flagged in the input data quality files (c1m.fits) such as bad columns and hot pixels. Adjustments to the AstroDrizzle parameters ***driz_cr_snr*** and ***driz_cr_scale*** were needed to keep the central peak of some bright stars from being flagged as cosmic rays.

3. Can image offsets be improved using TweakReg?

Source detection in WFPC2 images can be a bit tricky due to its undersampled PSF, detector artifacts like hot pixels, and cosmic rays in long exposures.

TweakReg's source identification routines may sometimes have trouble distinguishing between real sources and artifacts, especially if the field is sparse. Its object detection stage can be improved by taking some extra steps:

(a) Create rough approximations of input images with detector artifacts and cosmic rays removed. These images, with suffix `c0m_crclean.fits`, were created in the previous AstroDrizzle step by setting the parameter ***driz_cr_corr=True***.

(b) TweakReg has a tendency to pick up spurious objects along the chip pyramid edges. A simple Python script is used to set those edges to zero.

(c) Artifacts in diffraction spikes of bright sources sometimes get misidentified as real objects. The bright stars and diffraction spikes can be masked using a circular **ds9** regions file, referred to as "exclusion regions" in AstroDrizzle.

Mask Pyramid Edges in Chips

To set the pyramid edges to zero, the following steps are used:

- First, create a subdirectory for running TweakReg.

```
--> !mkdir tweak_dir
```

- Since this example is being run in the PyRAF environment, use the `!` symbol that escapes to the C shell to execute the UNIX copy command, `cp`. Copy images ending with `crclean.fits` to `tweak_dir`.

```
-->!cp ua060502m_c0m_crclean.fits tweak_dir/ua060502m_c0m_cleanmask.fits
-->!cp ua060504m_c0m_crclean.fits tweak_dir/ua060504m_c0m_cleanmask.fits
-->!cp ua060506m_c0m_crclean.fits tweak_dir/ua060506m_c0m_cleanmask.fits
-->!cp ua060508m_c0m_crclean.fits tweak_dir/ua060508m_c0m_cleanmask.fits
-->!cp ua06050am_c0m_crclean.fits tweak_dir/ua06050am_c0m_cleanmask.fits
-->!cp ua06050am_c0m_crclean.fits tweak_dir/ua06050cm_c0m_cleanmask.fits
```

Or to save time, just execute this simple awk statement, as done here in the PyRAF window.

```
--> !ls *crclean* | awk '{print "cp "$1" tweak_dir/"substr($1,1,14)"cleanmask.fits"}' | "sh"
```

- In `tweak_dir`, create a text file called `edges.py`, shown below. This is a short Python script to set the chip pyramid edges to zero for each image in the array `imglist`. These pyramid edge boundaries were previously determined by manually by measuring bad areas for each chip in an image.

```
--> cd tweak_dir
```

Contents of file `edges.py`:

```
import pyfits
imglist=['ua060502m','ua060504m','ua060506m','ua060508m','ua06050am','ua06050cm']
for root in imglist:
  twkimg = pyfits.open(root+'_c0m_cleanmask.fits',mode='update')
  twkimg[1].data[0:799,0:49] = twkimg[1].data[0:62,0:799] = 0.0
  twkimg[2].data[0:799,0:53] = twkimg[2].data[0:39,0:799] = 0.0
  twkimg[3].data[0:799,0:39] = twkimg[3].data[0:53,0:799] = 0.0
  twkimg[4].data[0:799,0:49] = twkimg[4].data[0:49,0:799] = 0.0
  twkimg.close()
```

In Python, note that indentations are important; the first three rows should be at the beginning of the line, and the following commands within the loop are indented by a couple of spaces.

Run this script using Python in the UNIX C-shell:

```
--> python edges.py
```

Or in PyRAF,

```
--> !python edges.py
```

Figure 2: Avoid spurious source detections by setting pyramid edges to zero.

The image on the left is the original form of ua060502m_c0m.fits[4]. The image on the right is ua060502m_c0m_cleanmask.fits[4], the cosmic ray-cleaned c0m_crclean.fits images from AstroDrizzle with the pyramid edges set to zero.

Running TweakReg

Use the `c0m_cleanmask.fits` images as input to TweakReg. A DrizzlePac task called ImageFindPars, containing parameters for specifying detection characteristics for sources, is used by TweakReg and its parameters can be incorporated into the TweakReg command, as shown below.

By default, the first image in the list is chosen as the reference image. In this example, it's `ua060502m_c0m_cleanmask.fits`. Each of the other images are aligned with respect to it using a source matching algorithm similar to IRAF's **xyxymatch** task.

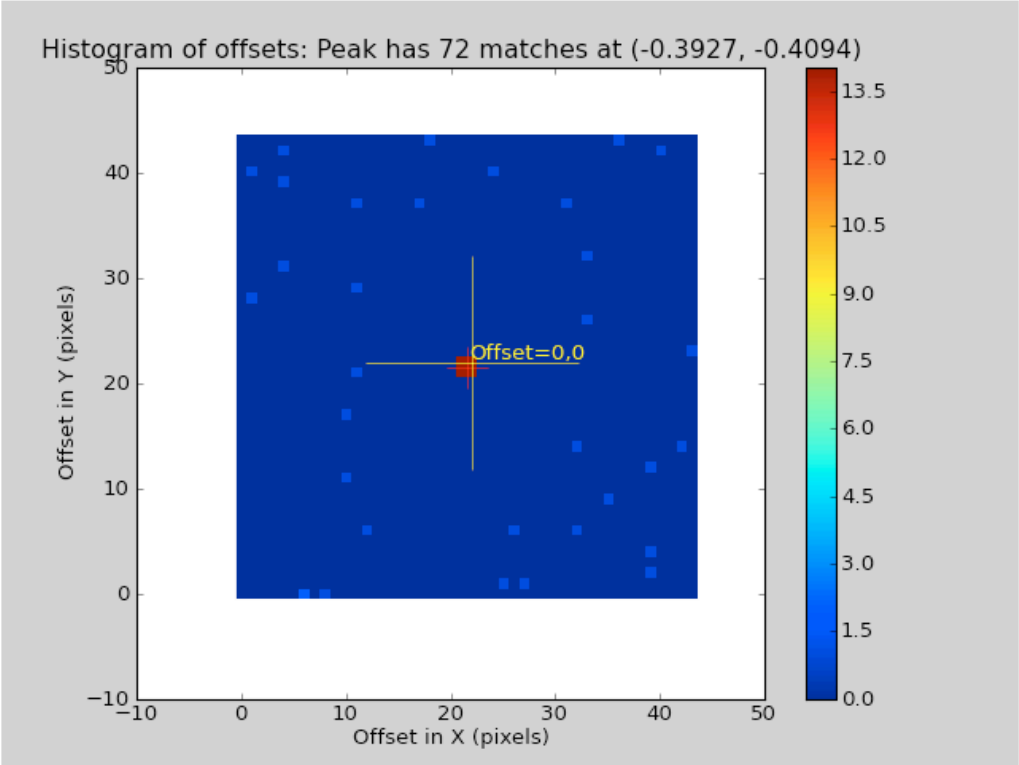
In practice, TweakReg should be run several times, adjusting the ImageFindPars parameters such as ***threshold***, ***peakmin***, and ***peakmax*** to avoid very faint objects that are hard to center. The ***conv_width*** parameter is set to twice the PSF FWHM in the chip being measured. Since most sources are in the WF, and the typical PSF FWHM in the WF is 1.5 pixels, a value of ***3.0*** is adopted in this example, A good alignment can be characterized as having an alignment fit RMS of 0.1 pixels and lower. (In this example, as you'll later see, the RMS was about 0.2 pixels in the PC scale but since most objects were in the WF chips, this RMS is equvalent to about 0.1 WF pixels. Even so, it's not always possible to achieve RMS values of about 0.1 pixels with undersampled data.)

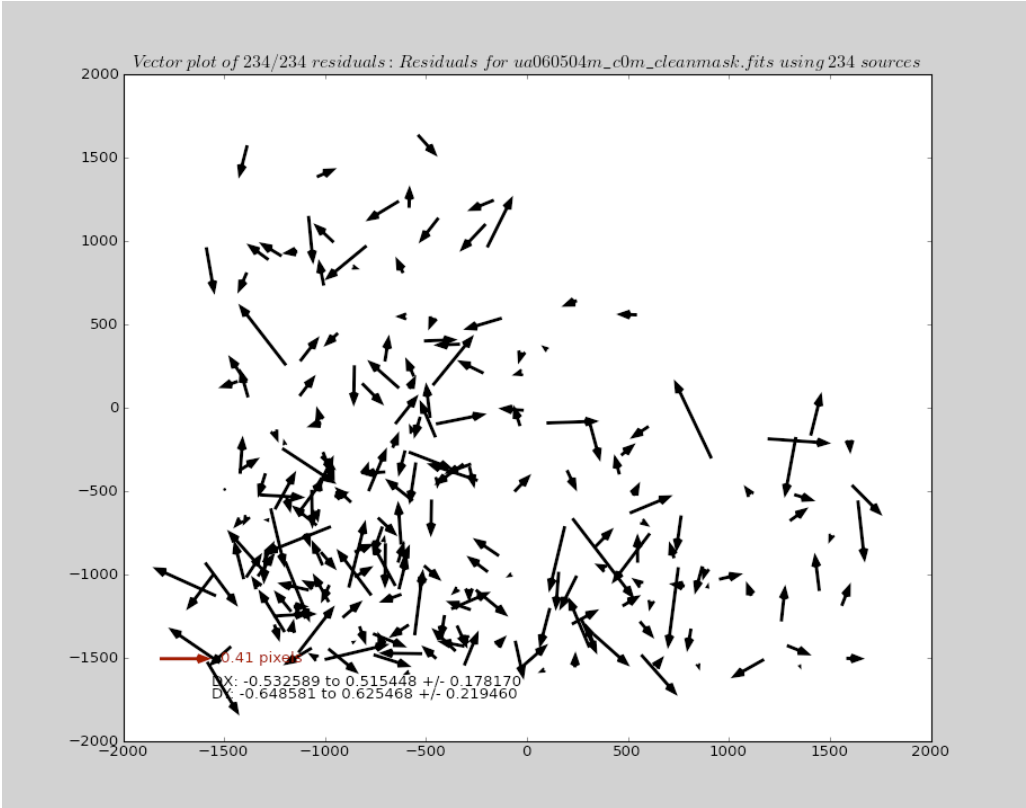
For this example, TweakReg was run several times; the parameters below provided an acceptable alignment fit. These are the commands typed at the PyRAF (or Python) interface[7]:

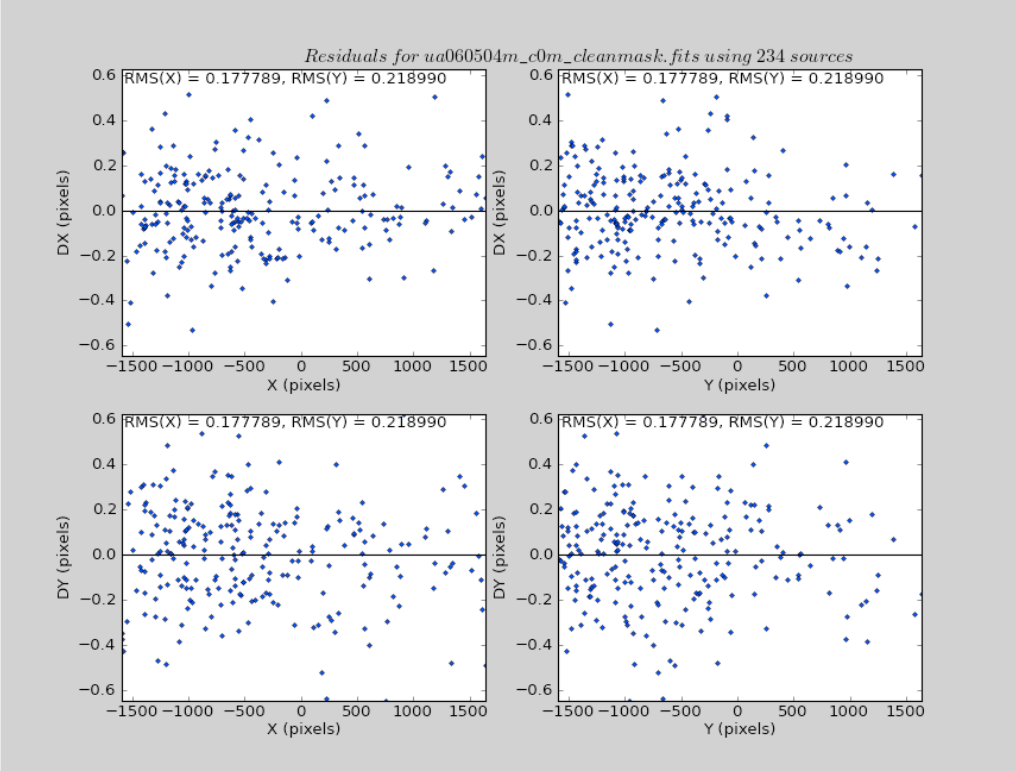
```
from drizzlepac import tweakreg
from drizzlepac import imagefindpars
tweakreg.TweakReg('*cleanmask.fits',updatewcs=False,conv_width=3.0,threshold=300.0, \
peakmin=100,peakmax=10000)
```

As TweakReg runs, it displays plots for each alignment fit, showing a probability distribution for the offset, and plots showing the x,y RMS for the alignment fit. To proceed from one plot to the next, hit the *return* key. To terminate processing, type "`q`". (If you do not wish to view the plots, add the parameter ***interactive=False***.)

[7] If you elect to run TweakReg using the TEAL interface, be sure to set TweakReg and ImageFindPars to default values before proceeding.


Histogram of offsets: Peak has 72 matches at (-0.3927, -0.4094)
Offset=0,0
Offset in X (pixels)
Offset in Y (pixels)
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50
13.5
12.0
10.5
9.0
7.5
6.0
4.5
3.0
1.5
0.0


Vector plot of 234/234 residuals: Residuals for ua060504m_c0m_cleanmask.fits using 234 sources
0.41 pixels
DX: -0.532589 to 0.515448 +/- 0.178170
DY: -0.648581 to 0.625468 +/- 0.219460
2000
1500
1000
500
0
−500
−1000
−1500
−2000
−2000
−1500
−1000
−500
0
500
1000
1500
2000



Figures 3a, 3b, and 3c above show the probability distribution for the offset, and plots showing the x,y RMS for an alignment fit.

The default value for the ***fitgeometry*** parameter in TweakReg is ***rscale***, which fits the offsets, rotations and scale changes from the matched object lists. These values for each image, with respect to the first (reference) image, can be viewed in files ending with `c0m_cleanmask_catalog_fit.match`. Here, the UNIX `grep` command is used for a quick look at the shifts and RMS, along with rotation and scale differences between each image and the reference image, respectively.

```
--> !grep shift *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0872806  0.0607778
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.16117    0.15522
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.154591   0.0678555
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0808349  -0.0396556
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0715184  -0.00362648

--> !grep rms *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.177789  0.21899
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.209588  0.202826
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.218422  0.181696
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.216939  0.210458
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.203379  0.188895

--> !grep rotation *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  0.000264901
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.999
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.998
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.998
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.998
```

```
--> !grep scale *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  1.00001   1.00001
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  1         1
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999956  0.999956
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999926  0.999926
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999926  0.999926
```

As mentioned earlier, the recommended fit RMS is less than 0.1 pixels. In this case, the RMS was ~0.2 pixels. However, this value is based on the PC plate scale which is less than half the size of the WF plate scale. Since most sources are in the WF, this RMS is equivalent to ~0.1 WF pixels, which is within the recommended[8] target RMS.

For these datasets, the computed shifts are significantly smaller than the RMS. In addition, the rotation and scale values of the images with respect to the reference image were negligible. This indicates that the images were already well-aligned based on their original WCSs alone. However, this is not always the case, which is why TweakReg is recommended for checking image alignments.

*Using a **ds9** Regions File as a Mask*

In cases where there are few sources for alignment, spurious sources could throw off the alignment fit RMS significantly. In such situations, TweakReg could be re-run using a mask to keep its object-finding algorithm away from those spurious sources.

Even though the images in this example are well-aligned, included here is a short demonstration of how to create a **ds9** regions file to mask bad sources, for use in TweakReg.

First, view each chip in each image; display it and plot the corresponding TweakReg image catalog on it. The image catalog for each image chip has the suffix `sci?_xy_catalog.coo`, where "`?`" is the chip number. The catalogs contain the positions of objects that were identified by TweakReg as sources.

In PyRAF:

```
--> disp ua06050cm_c0m_cleanmask.fits[4] 1 zr- zs- z1=-5 z2=70
--> tvmark 1 ua06050cm_c0m_cleanmask_sci4_xy_catalog.coo mark=circle radii=10 int-
```

[8] The RMS criteria was originally defined for ACS/WFC and WFC3/UVIS images, that have plate scales of 0.05" and 0.04", respectively.

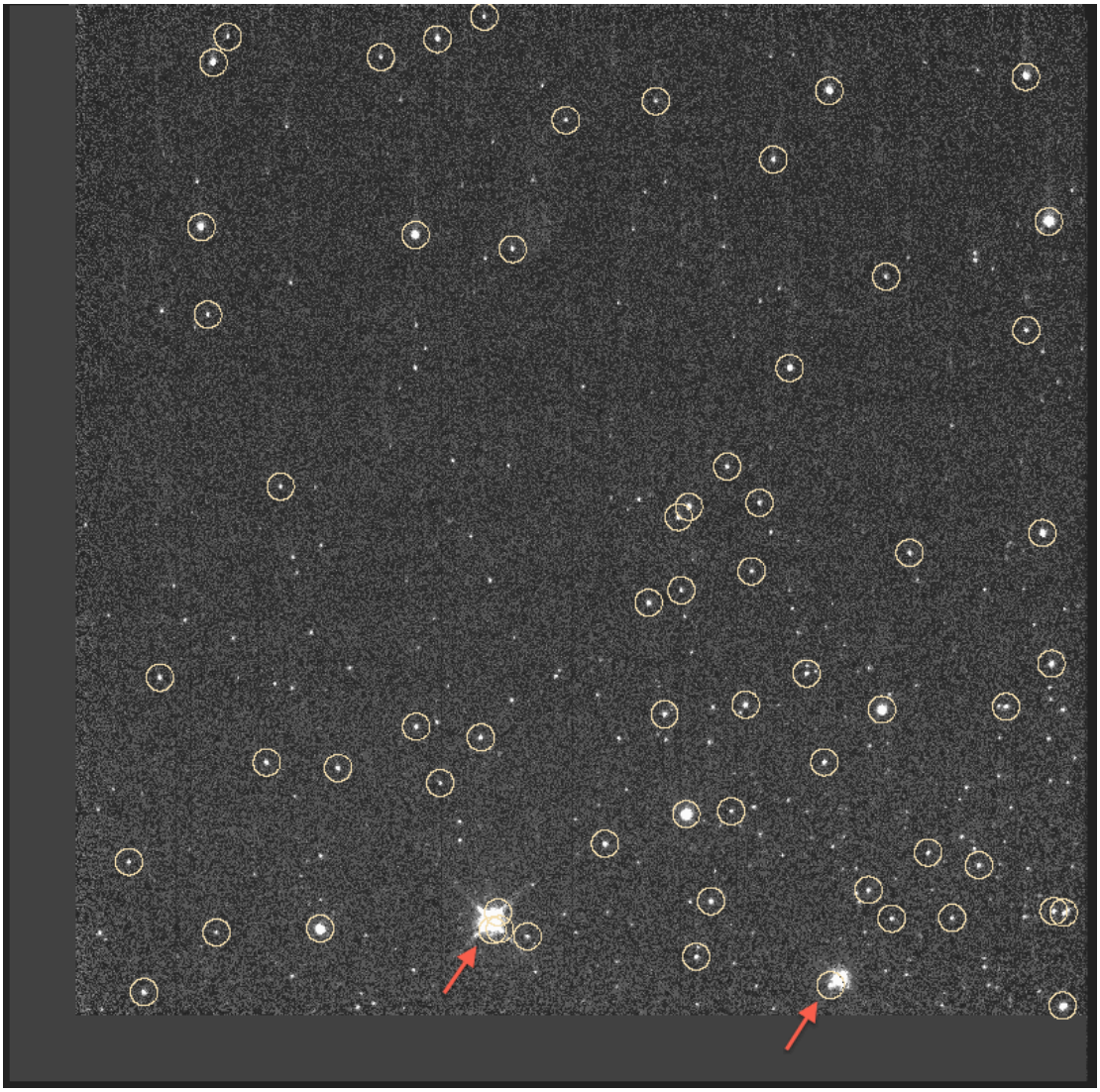


Figure 4: The TweakReg source catalog plot on a WF4 chip image.
Note the spurious sources around the two bright stars at the bottom.

In looking over all images and their corresponding catalogs, it appeared that the TweakReg object detection software kept identifying spurious source in the diffraction spikes of two bright stars in WF4, as shown in the image above.

The alignment fit can be improved by excluding those two bright stars in WF4, using a **ds9 "**regions file" to record the location around each of those bright stars; this is done by clicking on each object and drawing a circle that encompasses the star and most of its wings. (At this time, TweakReg only accepts circular regions, this may change in the future.) That information is saved to a regions file in the form of a position and radius, one line for each star. This regions file will be used in TweakReg to mask a circular area around each of the two bright stars.

To create a regions file:

- Load one of the images directly in **ds9**. *Do not use the IRAF* **display** *command.* At the **ds9** window, select ***File***, then select ***Open***. Find the image and open it in **ds9**. Be sure to add the group 4 extension, i.e., `ua060502m_c0m_cleanmask.fits[4]`
- In the ***Region*** pull-down menu**,** select ***Delete All Regions*** to remove any prior **ds9** markers that may be on the **ds9** display screen.
- Under ***Region,*** select ***shape,*** then select ***circle.***
- Use the cursor to draw a circle around each of the two bright stars*,* large enough to encompass sections of the diffraction spikes that generate spurious sources.
- When both stars have been marked with circles, save the position and radius of those circles to a file: under the ***Region*** menu, select ***Save regions.*** For this example, name it `wf4_exclude.reg.` Click ***OK.***
- A pop-up menu appears. For ***Format***, select ***ds9***; for ***Coordinate System***, select ***WCS***; click ***OK.*** Since the regions file was saved in the WCS coordinate system (RA and Dec of the source center, and a radius) this file can be used for all images in the next TweakReg run.

The format of the **ds9** regions file is shown below. The values of the center and radius will be slightly different depending on how you set it on your own machine.

```
--> !more wf4_exclude.reg
# Region file format: DS9 version 4.1
# Filename: /Users/shireen/ASTRODRIZZLE/ASTRODRIZ_EX_web/WFPC2_example_webpage/WFPC2/30Aug13/
tweak_dir/ua060502m_c0m_cleanmask.fits[SCI]
global color=green dashlist=8 3 width=1 font="helvetica 10 normal" select=1 highlite=1 dash=0
fixed=0 edit=1 move=1 delete=1 include=1 source=1
fk5
circle(323.32727,-0.86569003,3.0259204")
circle(323.33406,-0.86316979,1.660978")
```

Next, create a new file called `exclude.list`. This is an "exclusions" file, a list of images and the name(s) of regions file(s) associated with them. It instructs TweakReg to avoid searching for sources in the areas described by the regions file(s). The first column is the image name. The next three columns represent PC, WF2, and WF3, with "`none`" indicating there are no regions files for those groups. The last column, for WF4, names the regions file that was previously created. Shown below is the file "`exclude.list`:"

```
ua060502m_c0m_cleanmask.fits none none none wf4_exclude.reg
ua060504m_c0m_cleanmask.fits none none none wf4_exclude.reg
ua060506m_c0m_cleanmask.fits none none none wf4_exclude.reg
ua060508m_c0m_cleanmask.fits none none none wf4_exclude.reg
ua06050am_c0m_cleanmask.fits none none none wf4_exclude.reg
ua06050cm_c0m_cleanmask.fits none none none wf4_exclude.reg
```

Note that since `wf4_exclude.reg` has units in RA and Dec., the same file can be applied to all images.

Run TweakReg with the exclusions file.

```
tweakreg.TweakReg('*cleanmask.fits',updatewcs=False,exclusions='exclude.list', \
conv_width=3.0, threshold=300.0, peakmin=100, peakmax=10000, interactive=False)
```

Use **grep** to look at the shifts and their RMSs, as well as rotation and scale offsets, for alignments between the first image and each of the other images.

```
--> !grep shift *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0819565  0.0711967
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.159253   0.15542
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.149202   0.0644726
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0816095  -0.0385985
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y shift:  0.0719669  -0.004291
```

```
--> !grep rms *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.175647  0.208611
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.210049  0.203486
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.214002  0.181921
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.217729  0.210328
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y rms:  0.203854  0.189058

--> !grep rotation *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  360
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.999
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.999
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.998
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y rotation:  359.998

--> !grep scale *c0m_cleanmask_catalog_fit.match
ua060504m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  1.00001   1.00001
ua060506m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999999  0.999999
ua060508m_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999954  0.999954
ua06050am_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999927  0.999927
ua06050cm_c0m_cleanmask_catalog_fit.match:#  X and Y scale:  0.999926  0.999926
```

In comparing the offsets and their residuals, as well as scale and rotation offsets between this TweakReg run and the previous one, there's no significant change in values. But if the shifts had been significant, a new alignment fit would necessitate an update to the WCS of calibrated `c0m.fits` files that would then be reprocessed by AstroDrizzle. Assuming this is the case, the next step is to generate headerlets containing the new alignment solutions, and then transfer the new WCSs to `c0m.fits` files and run AstroDrizzle on them.

4. Transfer the new WCS, based on improved image offsets, to the calibrated images.

The new WCS derived in step 3 can be stored in headerlets, which can then be applied to `c0m.fits` files. Headerlets are small FITS files that can be created by TweakReg. They contain new WCS keywords values describing the new alignment fit. There is one headerlet for each corresponding `c0m.fits` image.

Information in the headerlets will be attached to their respective `c0m.fits` files, and the new WCS will become the primary WCS for each `c0m.fits` images, using the **stsci_tools** Python function **apply_headerlet_as_primary**. *Be sure to use a version of the* `c0m.fits` *and* `c1m.fits` *files that had previously been processed by AstroDrizzle because they have already been configured to the new FITS format* -- headerlet information can only be attached to images in the new FITS format. Then, run AstroDrizzle on the `c0m.fits` images that have updated WCS information in it.

Run TweakReg again to create headerlets

Run TweakReg as before, this time with ***headerlet=Yes***, ***Attach=No***.

```
tweakreg.TweakReg('*cleanmask.fits',updatewcs=False,headerlet=True, hdrname='test1', \
attach=False, exclusions='exclude.list',conv_width=3.0, threshold=300.0, peakmin=100, \
peakmax=10000,see2dplot=False,residplot= 'No Plot')
```

Headerlets will be generated using the default naming convention:

```
ua060502m_c0m_cleanmask_hlet.fits
ua060504m_c0m_cleanmask_hlet.fits
ua060506m_c0m_cleanmask_hlet.fits
ua060508m_c0m_cleanmask_hlet.fits
ua06050am_c0m_cleanmask_hlet.fits
ua06050cm_c0m_cleanmask_hlet.fits
```

These headerlets will be applied to corresponding `c0m.fits` images that were used in the initial AstroDrizzle run; that's because those `c0m.fits` and `c1m.fits` files have already been reconfigured by AstroDrizzle to be

compatible with DrizzlePac tasks. This update will be done using the Python function **headerlet.apply_as_primary**, that will apply the newly-derived WCS as the primary WCS in the `c0m.fits` files.

Apply the new WCSs to the c0m.fits images

Copy the `c0m.fits` and `c1m.fits` files from the first AstroDrizzle run, and headerlets in the `tweak_dir` directory, to a new working directory (i.e., `/Users/jane/mydata/adz_new_wcs/`), then run **apply_headerlet_as_primary**.

To run it on a batch of images, create a file with the following commands; in this example, it's called `run_apply_headlet.py`.

```
from stwcs.wcsutil import headerlet
imglist=['ua060502m','ua060504m','ua060506m','ua060508m','ua06050am','ua06050cm']
for root in imglist:
  headerlet.apply_headerlet_as_primary(root+'_c0m.fits',root+'_c0m_cleanmask_hlet.fits',force=True)
```

To run it in the PyRAF environment:

```
--> !python run_apply_headlet.py
```

To verify that it worked, check the `hdrname` keyword in the `c0m.fits` image, to make sure it is set to `test1`.

```
--> hsel *c0m.fits[1] $I,hdrname yes
ua060502m_c0m.fits[1]   test1
ua060504m_c0m.fits[1]   test1
ua060506m_c0m.fits[1]   test1
ua060508m_c0m.fits[1]   test1
ua06050am_c0m.fits[1]   test1
ua06050cm_c0m.fits[1]   test1
```

5. Re-run AstroDrizzle using images with updated WCSs, and try different *final_scale* and *final_pixfrac* settings.

Use these updated `c0m.fits` (and `c1m.fits`) files to run AstroDrizzle again, using the same syntax as shown at the beginning of the example, except leave out the parameters ***updatewcs*** and ***driz_cr_corr***.

In running AstroDrizzle with better image alignments, you're also going to be improving cosmic ray rejection. As a reminder, here's the command for ***final_pixfrac=1.0*** and ***final_scale*** is not mentioned so the default value of 0.046"/pixel is used.

```
astrodrizzle.AstroDrizzle('*c0m.fits', driz_sep_fillval=99999, \
driz_sep_bits='8,1024',combine_type='minmed', driz_cr_snr='5.5 3.5', \
driz_cr_scale='2.0 1.5', final_wht_type='EXP', final_wt_scl='exptime', \
final_pixfrac=1.0, final_fillval=99999, final_bits='8,1024')
```

Run it a few more times, with different ***final_scale*** and ***final_pixfrac*** settings. The main goal is to squeeze as much spatial resolution as possible from the images while making sure the final drizzle-combined image is well-sampled.

In choosing the ***final_scale*** and ***final_pixfrac*** settings, keep in mind that *the "drop size" (pixfrac) should always be larger than the scale so that some of the "drops" spill onto adjacent pixels*. Weight image statistics are an indication of how well the images are sampled to optimize signal-to-noise. The DrizzlePac Handbook recommends that the weight image RMS should be within 20% of the mean.

The table below shows a summary of measured FWHM values and weight image statistics for several ***final_scale*** and ***final_pixfrac*** settings. (One parameter was changed while the other stayed constant; users are encouraged to try changing both parameters at the same time to see how the PSF FWHM and weight image properties are affected.)

By default, AstroDrizzle uses the plate scale of the first chip as the ***final_scale*** value for the entire image. For a 4-group WFPC2 image, the first chip is the PC, with a plate scale of 0.046"/pixel.

For these tests, sources were picked in a baseline image (***final_pixfrac=1***, ***final_scale=0.046***) using the IRAF task **daofind**. The IRAF task **psfmeasure** was used to measure the PSF FWHM of the sources. To remove possible unresolved multiple stars, only objects with PSF FWHM values between 3.1 to 4 pixels were selected; these values should safely represent single stars and account for changes in the PSF across the FOV. A few stars were manually removed due to severe distortion caused by CTE or for being too close to the chip edges. Total number of sources in the PSF FWHM statistics is 390. Weight image statistics were obtained at the central 500x500 pixels in each chip, using the IRAF task **imexam**.

final_pixfrac & ***final_scale*** Settings	Median PSF FWHM in Arcsec (and PC Pixels)	Median and Std. Dev. in Central 500x500 Pixels of Weight Image (Median/Std.Dev. % in Parenthesis)	
Default values: final_pixfrac=1, final_scale=0.046	**0.1601 +/- 0.0112" (3.48 +/- 0.2428 pix)**	**PC: 56.26 +/- 5.339 (9.5%)** **WF3: 12.55 +/- 0.7687 (6.1%)**	**WF2: 12.55 +/- 0.7276 (5.8%)** **WF4: 12.53 +/- 0.9569 (7.6%)**
Adjust final_pixfrac. final_scale remains at 0.046"/pix.			
final_pixfrac=0.9, final_scale=0.046	0.1559 +/- 0.011" (3.39 +/- 0.2452 pix)	PC: 56.44 +/- 5.485 (9.7%) WF3: 12.27 +/- 0.9787 (8.0%)	WF2: 12.29 +/- 1.131 (9.2%) WF4: 12.11 +/- 1.17 (9.7%)
final_pixfrac=0.8, final_scale=0.046	0.1532 +/- 0.0114" (3.33 +/- 0.2487 pix)	PC1: 56.59 +/- 5.646 (10%) WF3: 11.98 +/- 1.35 (11.3%)	WF2: 12.12 +/- 1.926 (15.9%) WF4: 11.85 +/- 1.606 (13.5%)
final_pixfrac=0.7, final_scale=0.046	0.1500 +/- 0.012" (3.26 +/- 0.2616 pix)	PC1: 56.85 +/- 5.809 (10.2%) WF3: 11.86 +/- 1.813 (15.3%)	WF2: 11.99 +/- 2.866 (23.9%) WF4: 11.75 +/- 2.194 (18.7%)
Adjust final_scale, shown below in arcsec (fractional value in parenthesis). final_pixfrac remains at 1.0.			
final_pixfrac=1, final_scale=0.041 (0.9)	0.1619 +/- 0.0126" (3.91 +/- 0.3043 pix)	PC1: 45.72 +/- 4.604 (10.1%) WF3: 10.17 +/- 0.653 (6.4%)	WF2: 10.17 +/- 0.5905 (5.8%) WF4: 10.15 +/- 0.7797 (7.7%)
final_pixfrac=1, final_scale=0.037 (0.8)	0.1615 +/- 0.0107" (4.39 +/- 0.2919 pix)	PC1: 37.22 +/- 3.837 (10.3%) WF3: 8.281 +/- 0.5441 (6.6%)	WF2: 8.277 +/- 0.4957 (6.0%) WF4: 8.269 +/- 0.633 (7.7%)
final_pixfrac=1, final_scale=0.032" (0.7)	0.161 +/- 0.0101" (5. +/- 0.3129 pix)	PC1: 27.89 +/- 2.967 (10.6%) WF3: 6.194 +/- 0.3976 (6.4%)	WF2: 6.191 +/- 0.3896 (6.3%) WF4: 6.185 +/- 0.4849 (7.8%)

It's clear from the median PSF FWHM values that reducing the ***final_pixfrac*** value while keeping ***final_scale*** at 0.046"/pix provides a tighter PSF FWHM, but at some expense of the signal-to-noise ratio (as evident in the weight image statistics). Still the FWHM scatter (shown in Figure 5) remains fairly large. While sub-pixel dithering can recover some resolution in highly undersampled images, it should be done cautiously, taking into consideration the location of the source(s) and science goals.

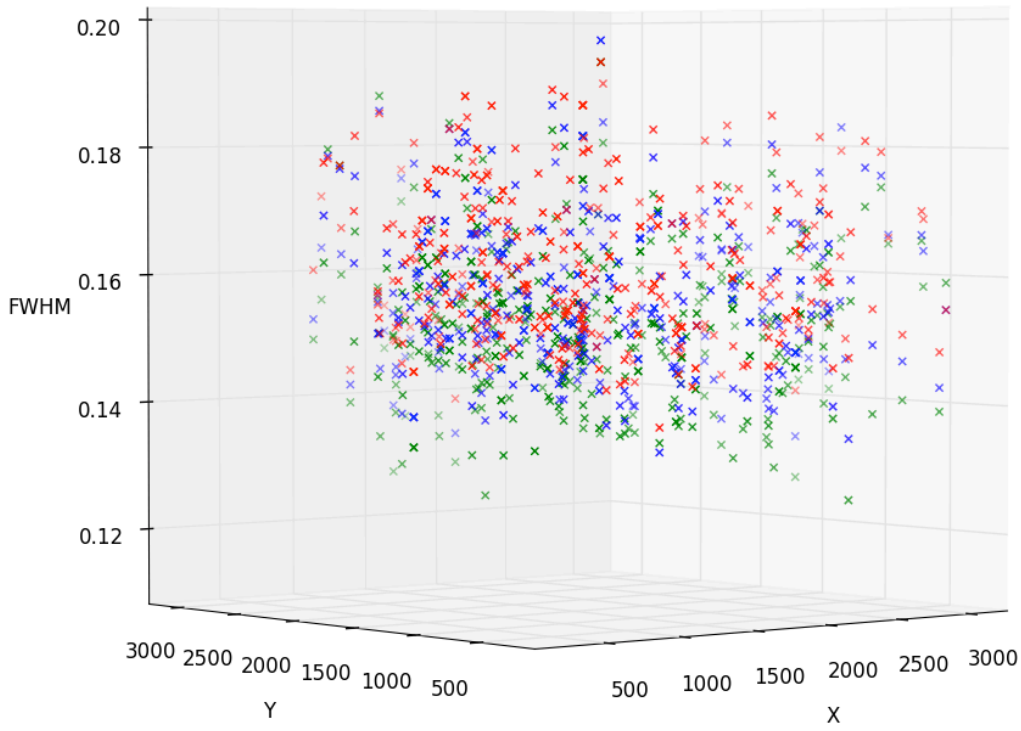


Figure 5. The PSF FWHM (in arcsec) for ***final_scale=0.046***", with ***final_pixfrac*** of ***1*** (red), ***0.9*** (blue), and ***0.8*** (green). While the eye can pick out a decrease in overall FWHM values as ***final_pixfrac*** is reduced, the scatter remains quite large.