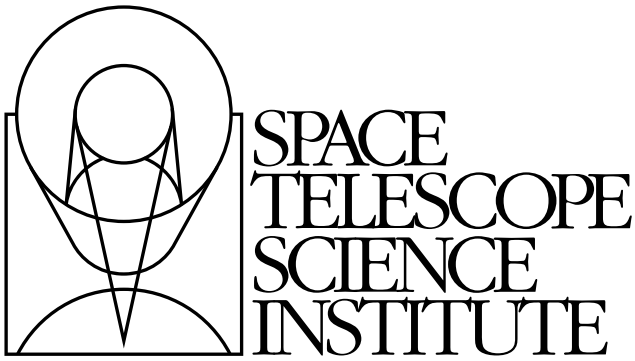

drizzlepac Documentation

Release 1.1.8(06-Feb-2013)

Warren Hack, Nadia Dencheva, Chris Sontag, Megan Sosey, Micha

May 03, 2013

CONTENTS

This package supports the use of MultiDrizzle as an integrated set of modules that can be run in an automated manner to combine images. The version of MultiDrizzle described here implements a single task to run the entire MultiDrizzle processing pipeline, while also providing the framework for users to create their own custom pipeline based on the modules in this package merged with their own custom code if desired.

This package relies on the STWCS and PyWCS packages in order to provide the support for the WCS-based distortion models and alignment of the input images.

Contents:

CHAPTER

ONE

PRIMARY USER INTERFACE: ASTRODRIZZLE()

AstroDrizzle - Python implementation of MultiDrizzle

AstroDrizzle automates the process of aligning images in an output frame, identifying cosmic-rays, removing distortion, and then combining the images after removing the identified cosmic-rays.

This process involves a number of steps, namely:

1. Processing the input images and input parameters
2. Creating a static mask
3. Performing sky subtraction
4. Drizzling onto separate output images
5. Creating the median image
6. Blotting the median image
7. Identifying and flagging cosmic-rays
8. Final combination

A full description of this process can be found in the DrizzlePac Handbook available online at:

http://drizzlepac.stsci.edu

Output: The primary output from this task is the distortion-corrected, cosmic-ray cleaned, and combined image as a FITS file.

This task requires numerous user-settable parameters to control the primary aspects of each of the processing steps.

`drizzlepac.astrodrizzle.`**`AstroDrizzle`**(*input=None*, *mdriztab=False*, *editpars=False*, *configobj=None*, *wcsmap=None*, ***input_dict*)

astrodrizzle Version 1.1.8 updated on 06-Feb-2013

Parameters

input : str or list of str (Default Value = '*flt.fits')

The name or names of the input files to be processed, which can be provided in any of the following forms:

- filename of a single image
- filename of an association (ASN) table
- wild-card specification for files in directory
- comma-separated list of filenames

- '@file' filelist containing list of desired input filenames. The file list needs to be provided as an ASCII text file containing a list of filenames for all input images with one filename on each line of the file. If inverse variance maps (IVM maps) have also been created by the user and are to be used (by specifying 'IVM' to the parameter final_wht_type), then these are simply provided as a second column in the filelist, with each IVM filename listed on the same line as a second entry, after its corresponding exposure filename.

Note: If the user specifies 'IVM' for the 'final_wht_type', but does not provide the names of IVM files, AstroDrizzle will automatically generate the IVM files itself for each input exposure.

Output : str (Default Value = '')

The rootname for the output drizzled products. This step can result in the creation of several files, including:

- copies of each input image as a FITS image, if workinplace=Yes and/or input images are in GEIS format.
- mask files and coeffs files created by PyDrizzle for use by 'drizzle'.

If an association file has been given as input, the specified filename will be used instead of the product name specified in the ASN file. Similarly, if a single exposure is provided, the rootname of the single exposure will be used for the output product instead of relying on the input rootname. If no value is provided when a filelist or wild-card specification is given as input, then a rootname of 'final' will be used for the output file name.

mdriztab : :

This button will immediately update the parameter values in the TEAL GUI based on those provided by the MDRIZTAB reference table referenced in the first input image. This requires that the MDRIZTAB reference file be available locally.

runfile : str (Default Value = 'astrodrizzle.log')

This log file will contain all the output messages generated during processing, including full details of any errors/exceptions. These messages will be a super-set of those reported to the screen during processing.

updatewcs : bool (Default Value = No)

This parameter specifies whether the WCS keywords are to be updated by running makewcs on the input data, or left alone. The update performed by makewcs not only recomputes the WCS based on the currently used IDCTAB, but also populates the header with the SIP coefficients. For ACS/WFC images, the time-dependence correction will also be applied to the WCS and SIP keywords. This parameter should be set to 'No' (False) when the WCS keywords have been carefully set by some other method, and need to be passed through to drizzle 'as is', otherwise those updates will be over-written by this update.

wcskey : str (Default Value = '')

This parameter corresponds to the *key* for the WCS being selected by the user. It allows the user to select which WCS solution should be used for processing the images when multiple WCS's have been updated in each input image header using the Paper I Multiple WCS FITS standard.

> **Warning:** Use of this parameter should be done only when all input images have been updated using the Paper I FITS standard for specifying Multiple WCS's in each image header. This parameter assumes that the same WCS letter corresponds to WCS's that have been updated in a consistent manner. For example, all input images have been updated to be consistent with their distortion model in the WCS's with key of *A*.

proc_unit : str (Default Value = 'native')

The units to be used for the final output drizzled product. Valid values and definitions are:

```
* **native**: Output DRZ product and input 'values' given in the native units of the
* **electrons**: Output DRZ product and input 'values' given in units of electrons.
```

coeffs : bool (Default Value = Yes)

This parameter determines whether or not to use the coefficients stored in the each input image header. If turned off, no distortion coefficients will be applied during the coordinate transformations.

context : bool (Default Value = Yes)

This parameter specifies whether or not to create a context image during the final drizzle combination. The context image contains the information regarding which image(s) contributed to each pixel encoded as a bit-mask. More information on context images can be obtained from the ACS Data Handbook.

group : int (Default Value = None)

This parameter establishes whether or not a single FITS extension, or group will be drizzled. If an extension is provided, then only that chip will be drizzled onto the output frame. Either a FITS extension number, a GEIS group number (such as '1'), or a FITS extension name (such as 'sci,1') may be specified.

build : bool (Default Value = No)

When this parameter is set to 'Yes' (True), AstroDrizzle will combine the separate 'drizzle' output files into a single multi-extension format FITS file. This combined output file will contain seperate SCI (science), WHT (weight), and CTX (context) extensions. If this parameter is set to 'No' (False), a separate simple FITS file will be created for each aforementioned extension.

crbit : int (Default Value = 4096)

This parameter sets the bit value for CR identification in the DQ array.

stepsize : int (Default Value = 10)

This parameter controls the internal grid of points used in the coordinate transformation from the input image to the output frame. The default value of 10 indicates that every 10th pixel will be transformed using the full WCS-based transformation. All remaining pixels will then be transformed using bilinear interpolation based on those pixels (i.e. every 10th pixel in the case of the default parameter setting) that were fully transformed.

resetbits : int (Default Value = 4096)

This parameter allows the user to specify which DQ bits of each input image DQ array should be reset to a value of 0. This operation is performed on the copy of the input data after updating the headers based on the 'updatewcs' parameter, and prior to starting any of the AstroDrizzle processing steps (static mask, sky subtraction, and so on).

num_cores: int (Default Value = None) :

This specifies the number of CPU cores to use during processing. Any value less than 2 will disable all use of parallel processing.

in_memory: bool (Default Value = False) :

This parameter sets whether or not to keep all intermediate products in memory when processing. This includes all single drizzle products (*single_sci and *single_wht), median image, blot images, and crmask images. *The use of this option will therefore require significantly more memory than usual to process the data while reducing the overall processing time by eliminating most of the disk activity. *Only* the products of the final drizzle step will get written out when this parameter gets specified as 'True'.

STATE OF INPUT FILES :

restore: bool (Default Value = No) :

Setting this to 'Yes' (True) directs AstroDrizzle to copy the input images from the 'OrIg_files' sub-directory and use them for processing, if they had been archived by AstroDrizzle using the 'preserve' or 'overwrite' parameters already. If set to 'Yes' and the input files had not been archived already, it will simply ignore this and work with the current input images.

preserve : bool (Default Value = Yes)

Copy input files to archive directory, if not already archived. This parameter determines whether or not astrodrizzle creates a copy of the input file in a sub-directory called 'OrIg_files'. If a copy already exists in this directory, then the previously existing version will NOT be overwritten.

overwrite : bool (Default Value = No)

Copy input files into archive, overwriting older files if required? This parameter will cause astrodrizzle to make a copy of each input file in the 'OrIg_files' directory regardless of whether a previous copy existed or not, and will overwrite any previous copy should it be present.

clean : bool (Default Value = No)

The temporary files created by AstroDrizzle can be automatically removed by setting this parameter to 'Yes' (True). The affected files include the coefficient and static mask files created by PyDrizzle, in addition to other intermediate files created by AstroDrizzle. It is often useful to retain the intermediate files and examine them when first learning how to run AstroDrizzle. However, when running AstroDrizzle routinely, or on a small disk drive, these files can be removed to conserve space.

STEP 1: STATIC MASK :

static : bool (Default Value = Yes)

Create a static bad-pixel mask from the data? This mask flags all pixels that deviate by more than a value of 'static_sig' sigma below the image median, since these pixels are typically the result of bad pixel oversubtraction in the dark image during calibration.

static_sig : float (Default Value = 4.0)

The number of sigma below the RMS to use as the clipping limit for creating the static mask.

STEP 2: SKY SUBTRACTION :

skysub : bool (Default Value = Yes)

Turn on or off sky subtraction on the input data.

skywidth : float (Default Value = 0.3)

Bin width, in sigma, used to sample the distribution of pixel flux values in order to compute the sky background statistics.

skystat : {'median', 'mode', 'mean'} (Default Value = 'median')

Statistical method for determining the sky value from the image pixel values.

skylower : float (Default Value = None)

Lower limit of usable pixel values for computing the sky. This value should be specified in the units of the input image(s).

skyupper : float (Default Value = None)

Upper limit of usable pixel values for computing the sky. This value should be specified in the units of the input image(s).

skyclip : int (Default Value = 5)

Number of clipping iterations to use when computing the sky value.

skylsigma : float (Default Value = 4.0)

Lower clipping limit, in sigma, used when computing the sky value.

skyusigma : float (Default Value = 4.0)

Upper clipping limit, in sigma, used when computing the sky value.

skyuser : str (Default Value = '')

Name of header keyword which records the sky value already subtracted from the image by the user.

skyfile : str (Default Value = '')

Name of file containing user-computed sky values to be used with each input image. This ASCII file should only contain 2 columns: image filename in column 1 and sky value in column 2. The sky value should be provided in units that match the units of the input image and for multi-chip images, the same value will be applied to all chips.

STEP 3: DRIZZLE SEPARATE IMAGES :

driz_separate : bool (Default Value = Yes)

This parameter specifies whether or not to drizzle each input image onto separate output images. The separate output images will all have the same WCS as the final combined output frame. These images are used to create the median image, needed for cosmic ray rejection.

driz_sep_kernel : str {'square','point','gaussian','turbo','tophat','lanczos3'} (Default Value = 'turbo')

Used for the initial separate drizzling operation only, this parameter specifies the form of the kernel function used to distribute flux onto the separate output images. The current options are:

- **square**: original classic drizzling kernel
- **point**: this kernel is a point so each input pixel can only contribute to the single pixel that is closest to the output position. It is equivalent to the limit as pixfrac -> 0, and is very fast.

- **gaussian**: this kernel is a circular gaussian with a FWHM equal to the value of pixfrac, measured in input pixels.
- **turbo**: this is similar to kernel="square" but the box is always the same shape and size on the output grid, and is always aligned with the X and Y axes. This may result in a significant speed increase.
- **tophat**: this kernel is a circular "top hat" shape of width pixfrac. It effects only output pixels within a radius of pixfrac/2 from the output position.
- **lanczos3**: a Lanczos style kernel, extending a radius of 3 pixels from the center of the detection. The Lanczos kernel is a damped and bounded form of the "sinc" interpolator, and is very effective for resampling single images when scale=pixfrac=1. It leads to less resolution loss than other kernels, and typically results in reduced correlated noise in outputs.

> **Warning:** The ''lanczos3" kernel tends to result in much slower processing as compared to other kernel options. This option should never be used for pixfrac != 1.0, and is not recommended for scale != 1.0.

The default for this step is **"turbo"** since it is much faster than **"square"**, and it is quite satisfactory for the purposes of generating the median image. More information about the different kernels can be found in the help file for the drizzle task.

driz_sep_wt_scl : float (Default Value = exptime)

This parameter specifies the weighting factor for input image. If driz_sep_wt_scl=exptime, then the scaling value will be set equal to the exposure time found in the image header. The use of the Default Value is recommended for producing optimal behavior for most scenarious. It is possible to set wt_scl=expsq for weighting by the square of the exposure time, which is optimal for read-noise dominated images.

driz_sep_pixfrac : float (Default Value = 1.0)

Fraction by which input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, or "dropsize", of a pixel in units of the input pixel size. If pixfrac is set to less than 0.001, the kernel parameter will be reset to 'point' for more efficient processing. In the step of drizzling each input image onto a separate output image, the default value of 1.0 is best in order to ensure that each output drizzled image is fully populated with pixels from the input image. For more information, see the help for the 'drizzle' task.

driz_sep_fillval : int (Default Value = None)

Value to be assigned to output pixels that have zero weight, or that receive flux from any input pixels during drizzling. This parameter corresponds to the 'fillval' parameter of the 'drizzle' task. If the default of 'None' is used, and if the weight in both the input and output images for a given pixel are zero, then the output pixel will be set to the value it would have had if the input had a non-zero weight. Otherwise, if a numerical value is provided (e.g. 0), then these pixels will be set to that value.

driz_sep_bits : int (Default Value = 0)

Integer sum of all the DQ bit values from the input image's DQ array that should be considered 'good' when building the weighting mask. This can also be used to reset pixels to good if they had been flagged as cosmic rays during a previous run of AstroDrizzle, by adding the value 4096 for ACS and WFPC2 data. Please see the section on Selecting the 'Bits' Parameter for a more detailed discussion.

STEP 3a: CUSTOM WCS FOR SEPARATE OUTPUTS :

driz_sep_wcs : bool (Default Value = No)

Define custom WCS for seperate output images?

driz_sep_refimage : str (Default Value = '')

Reference image from which a WCS solution can be obtained.

driz_sep_rot : float (Default Value = None)

Position Angle of output image's Y-axis relative to North. A value of 0.0 would orient the final output image to be North up. The default of None specifies that the images will not be rotated, but will instead be drizzled in the default orientation for the camera with the x and y axes of the drizzled image corresponding approximately to the detector axes. This conserves disk space, as these single drizzled images are only used in the intermediate step of creating a median image.

driz_sep_scale : float (Default Value = None)

Linear size of the output pixels in arcseconds/pixel for each separate drizzled image (used in creating the median for cosmic ray rejection). The default value of None specifies that the undistorted pixel scale for the first input image will be used as the pixel scale for all the output images.

driz_sep_outnx : int (Default Value = None)

Size, in pixels, of the X axis in the output images that each input will be drizzled onto. If no value is specified, the smallest size that can accommodate the full dithered field will be used.

driz_sep_outny : int (Default Value = None)

Size, in pixels, of the Y axis in the output images that each input will be drizzled onto. If no value is specified, the smallest size that can accommodate the full dithered field will be used.

driz_sep_ra : float (Default Value = None)

Right ascension (in decimal degrees) specifying the center of the output image. If this value is not designated, the center will automatically be calculated based on the distribution of image dither positions.

driz_sep_dec : float (Default Value = None)

Declination (in decimal degrees) specifying the center of the output image. If this value is not designated, the center will automatically be calculated based on the distribution of image dither positions.

STEP 4: CREATE MEDIAN IMAGE :

median : bool (Default Value = Yes)

This parameter specifies whether or not to create a median image. This median image will be used as the comparison 'truth' image :in the cosmic ray rejection step.

median_newmasks : bool (Default Value = Yes)

This parameter specifies whether or not new mask files will be created when the median image is created. These masks are generated from weight files previously produced by the "driz_separate" step, and contain all bad pixel information used to exclude pixels when calculating the median. Generally this step should be set to 'Yes' (True), unless

for some reason, it is desirable to include bad pixel information when generating the median.

combine_maskpt : float (Default Value = 0.3)

Percentage of weight image values, below which the are flagged.

combine_type : str {'median', 'mean', 'minmed', 'imedian', 'imean', 'iminmed'} (Default Value = 'minmed')

This parameter defines the method that will be used to create the median image. The 'mean' and 'median' options set the calculation type when running 'numcombine', a numpy method for median-combining arrays to create the median image. The "minmed" option will produce an image that is generally the same as the median, except in cases where the median is significantly higher than the minimum good pixel value. In this case, "minmed" will choose the minimum value. The sigma thresholds for this decision are provided by the "combine_nsigma" parameter. However, as the "combine_nsigma" parameter does not adjust for the larger probability of a single "nsigma" event with a greater number of images, "minmed" will bias the comparison image low for a large number of images. "minmed" is highly recommended for three images, and is good for four to six images, but should be avoided for ten or more images.

A value of 'median' is the recommended method for a large number of images, and works equally well as minmed down to approximately four images. However, the user should set the "combine_nhigh" parameter to a value of 1 when using "median" with four images, and consider raising this parameter's value for larger numbers of images. As a median averages the two inner values when the number of values being considered is even, the user may want to keep the total number of images minus "combine_nhigh" odd when using "median".

The options starting with 'i', such as 'imedian', works just like the normal median operation except when dealing with a pixel were all the values are flagged as 'bad'. In this case, the 'i' functions return the last pixel in the stack as if it were good. This will prevent saturated pixels in the image from leaving holes in the middle of the stars, for example.

combine_nsigma : float (Default Value = '4 3')

This parameter defines the sigmas used for accepting minimum values, rather than median values, when using the 'minmed' combination method. If two values are specified the first value will be used in the initial choice between median and minimum, while the second value will be used in the "growing" step to reject additional pixels around those identified in the first step. If only one value is specified, then it is used in both steps.

combine_nlow : int (Default Value = 0)

This parameter sets the number of low value pixels to reject automatically during image combination.

combine_nhigh : int (Default Value = 0)

This parameter sets the number of high value pixels to reject automatically during image combination.

combine_lthresh : float (Default Value = None)

Sets the lower threshold for clipping input pixel values during image combination. This value gets passed directly to 'imcombine' for use in creating the median image. If the parameter is set to "None", no thresholds will be imposed.

combine_hthresh : float (Default Value = None)

This parameter sets the upper threshold for clipping input pixel values during image combination. The value for this parameter is passed directly to 'imcombine' for use in creating the median image. If the parameter is set to "None", no thresholds will be imposed.

combine_grow : int (Default Value = 1)

Width, in pixels, beyond the limit set by the rejection algorithm being used, for additional pixels to be rejected in an image. This parameter is used to set the 'grow' parameter in 'imcombine' for use in creating the median image.

STEP 5: BLOT BACK THE MEDIAN IMAGE :

blot : bool (Default Value = Yes)

Perform the blot operation on the median image? If set to 'Yes' (True), the output will be median smoothed images that match each input chips location, and will be used in the cosmic ray rejection step.

blot_interp : str{'nearest', 'linear', 'poly3', 'poly5', 'sinc'} (Default Value = 'poly5')

This parameter defines the method of interpolation to be used when blotting drizzled images back to their original WCS solution. Valid options include:

- **nearest**: Nearest neighbor
- **linear**: Bilinear interpolation in x and y
- **poly3**: Third order interior polynomial in x and y
- **poly5**: Fifth order interior polynomial in x and y
- **sinc**: Sinc interpolation (accurate but slow)

The 'poly5' interpolation method has been chosen as the default because it is relatively fast and accurate.

If 'sinc' interpolation is selected, then the value of the parameter for 'blot_sinscl' will be used to specify the size of the sinc interpolation kernel.

blot_sinscl : float (Default Value = 1.0)

Size of the sinc interpolation kernel in pixels.

blot_addsky : bool (Default Value = Yes)

Add back a sky value using the MDRIZSKY value from the header. If 'Yes' (True), the blot_skyval parameter is ignored.

blot_skyval : float (Default Value = 0.0)

This is a user-specified custom sky value to be added to the blot image. This is only used if blot_addsky is 'No' (False).

STEP 6: REMOVE COSMIC RAYS WITH DERIV, DRIZ_CR :

driz_cr : bool (Default Value = Yes)

Perform cosmic-ray detection? If set to 'Yes' (True), cosmic-rays will be detected and used to create cosmic-ray masks based on the algorithms from 'deriv' and 'driz_cr'.

driz_cr_corr : bool (Default Value = No)

Create a cosmic-ray cleaned input image? If set to 'Yes' (True), a cosmic-ray cleaned _crclean image will be generated directly from the input image, and a corresponding _crmask file will be written to document detected pixels affected by cosmic-rays.

driz_cr_snr : list of floats (Default Value = '3.5 3.0')

The values for this parameter specify the signal-to-noise ratios for the 'driz_cr' task to be used in detecting cosmic rays. See the help file for 'driz_cr' for further discussion of this parameter.

driz_cr_grow : int (Default Value = 1)

The radius, in pixels, around each detected cosmic-ray, in which more stringent detection criteria for additional cosmic rays will be used.

driz_cr_ctegrow : int (Default Value = 0)

Length, in pixels, of the CTE tail that should be masked in the drizzled output.

driz_cr_scale : str (Default Value = '1.2 0.7')

Scaling factor applied to the derivative in 'driz_cr' when detecting cosmic-rays. See the help file for 'driz_cr' for further discussion of this parameter.

STEP 7: DRIZZLE FINAL COMBINED IMAGE :

driz_combine : bool (Default Value = Yes)

This parameter specifies whether or not to drizzle each input image onto the final output image. This applies the generated cosmic-ray masks to the input images and creates a final, cleaned, distortion-corrected image.

final_wht_type : {'EXP', 'ERR', 'IVM'} (Default Value = 'EXP')

Specify the type of weighting image to apply with the bad pixel mask

for the final drizzle step. The options for this parameter include: :

- **EXP**: The default of 'EXP' indicates that the images will be weighted according to their exposure time, which is the standard behavior for drizzle. This weighting is a good approximation in the regime where the noise is dominated by photon counts from the sources, while contributions from sky background, read-noise and dark current are negligible. This option is provided as the default since it produces reliable weighting for all types of data, including older instruments (eg., WFPC2), where more sophisticated options may not be available.
- **ERR**: Specifying 'ERR' is an alternative for ACS and STIS data. In these cases, the final drizzled images will be weighted according to the inverse variance of each pixel in the input exposure files, calculated from the error array data extension that is in each calibrated input exposure file. This array is exposure time dependent, and encapsulates all of the noise sources in each exposure including read-noise, dark current, sky background, and Poisson noise from the sources themselves. For WFPC2, the ERR array is not produced during the calibration process, and therefore is not a viable option. We advise extreme caution when selecting the "ERR" option, since the nature of this weighting scheme can introduce photometric discrepancies in sharp unresolved sources, although these effects are minimized for sources with gradual variations between pixels. The "EXP" weighting option does not suffer from these effects, and is therefore the recommended option.
- **IVM**: Specifying 'IVM' allows the user to either supply their own inverse-variance weighting map, or allow AstroDrizzle to generate one automatically on-the-fly during the final drizzle step. This parameter option may be necessary for specific purposes. For example, to create a drizzled weight file for software such as SExtractor, it is expected that a weight image containing all of the background noise sources (sky level, read-noise, dark current, etc), but not the Poisson noise from the objects themselves will be available. The user can create the inverse variance images and then specify their names using the 'input' parameter for AstroDrizzle to specify an '@file'. This would be a single ASCII file containing the list of input calibrated exposure filenames (one per line), with a second column containing the

name of the IVM file corresponding to each calibrated exposure. Each IVM file must have the same file format as the input file, and if provided as multi-extension FITS files (e.g., ACS or STIS data) then the IVM extension must have the EXTNAME of 'IVM'. If no IVM files are specified on input, then AstroDrizzle will rely on the flat-field reference file and computed dark value from the image header to automatically generate an IVM file specific to each exposure.

final_kernel : {'square','point','gaussian','turbo','tophat','lanczos3'} (Default Value = 'square')

This parameter specifies the form of the kernel function used to distribute flux onto the separate output images, for the initial separate drizzling operation only. The value options for this parameter include:

- **square**: original classic drizzling kernel
- **point**: this kernel is a point so each input pixel can only contribute to the single pixel that is closest to the output position. It is equivalent to the limit as pixfrac -> 0, and is very fast.
- **gaussian**: this kernel is a circular gaussian, measured in input pixels, with a FWHM value equal to the value of pixfrac.
- **turbo**: this is similar to kernel="square", except that the box is always the same shape and size on the output grid, and is always aligned with the X and Y axes. This may result in a significant speed increase.
- **tophat**: this kernel is a circular "top hat" shape of width pixfrac. It effects only output pixels within a radius of pixfrac/2 from the output position.
- **lanczos3**: a Lanczos style kernel, extending a radius of 3 pixels from the center of the detection. The Lanczos kernel is a damped and bounded form of the "sinc" interpolator, and is very effective for resampling single images when scale=pixfrac=1. It leads to less resolution loss than other kernels, and typically results in reduced correlated noise in outputs.

> **Warning:** The ''lanczos3" kernel tends to result in much slower processing as compared to other kernel options. This option should never be used for pixfrac != 1.0, and is not recommended for scale != 1.0.

The default for this step is **"turbo"** since it is much faster than **"square"**, and it is quite satisfactory for the purposes of generating the median image. More information about the different kernels can be found in the help file for the drizzle task.

final_wt_scl : float (Default Value = exptime)

This parameter specifies the weighting factor for input image. If final_wt_scl=exptime, then the scaling value will be set equal to the exposure time found in the image header. The use of the Default Value is recommended for producing optimal behavior for most scenarious. It is possible to set wt_scl=expsq for weighting by the square of the exposure time, which is optimal for read-noise dominated images.

final_pixfrac : float (Default Value = 1.0)

Fraction by which input pixels are "shrunk" before being drizzled onto the output image grid, given as a real number between 0 and 1. This specifies the size of the footprint, or "dropsize", of a pixel in units of the input pixel size. If pixfrac is set to less than 0.001, the kernel parameter will be reset to 'point' for more efficient processing. In the step of drizzling each input image onto a separate output image, the default value of 1.0 is best

in order to ensure that each output drizzled image is fully populated with pixels from the input image. For more information, see the help for the 'drizzle' task.

final_fillval : float (Default Value = None)

The value for this parameter is to be assigned to the output pixels that have zero weight or which do not receive flux from any input pixels during drizzling. This parameter corresponds to the 'fillval' parameter of the 'drizzle' task. If the default of 'None' is used, and if the weight in both the input and output images for a given pixel are zero, then the output pixel will be set to the value it would have had if the input had a non-zero weight. Otherwise, if a numerical value is provided (e.g. 0), then these pixels will be set to that numerical value.

final_bits : int (Default Value = 0)

Integer sum for all of the DQ bit values from the input image's DQ array that should be considered 'good' when building the weight mask. This can also be used to reset pixels to good if they had been flagged as cosmic rays during a previous run of AstroDrizzle, by adding the value 4096 for ACS and WFPC2 data. Please see the section on Selecting the 'Bits' Parameter for a more detailed discussion.

final_units : str (Default Value = 'cps')

This parameter determines the units of the final drizzle-combined image, and can either be 'counts' or 'cps'. It is passed through to 'drizzle' in the final drizzle step.

STEP 7a: CUSTOM WCS FOR FINAL OUTPUT :

final_wcs : bool (Default Value = No)

Obtain the WCS solution from a user-designated reference image?

final_refimage : str (Default Value = '')

Reference image from which a WCS solution can be obtained.

final_rot : float (Default Value = None)

Position Angle of output image's Y-axis relative to North. A value of 0.0 would orient the final output image to be North up. The default of None specifies that the images will not be rotated, but will instead be drizzled in the default orientation for the camera with the x and y axes of the drizzled image corresponding approximately to the detector axes. This conserves disk space, as these single drizzled images are only used in the intermediate step of creating a median image.

final_scale : float (Default Value = None)

Linear size of the output pixels in arcseconds/pixel for each separate drizzled image (used in creating the median for cosmic ray rejection). The default value of None specifies that the undistorted pixel scale for the first input image will be used as the pixel scale for all the output images.

final_outnx : int (Default Value = None)

Size, in pixels, of the X axis in the output images that each input will be drizzled onto. If no value is specified, the smallest size that can accommodate the full dithered field will be used.

final_outny : int (Default Value = None)

Size, in pixels, of the Y axis in the output images that each input will be drizzled onto. If no value is specified, the smallest size that can accommodate the full dithered field will be used.

final_ra : float (Default Valut = None)

Right ascension (in decimal degrees) specifying the center of the output image. If this value is not designated, the center will automatically be calculated based on the distribution of image dither positions.

final_dec : float (Default Valut = None)

Declination (in decimal degrees) specifying the center of the output image. If this value is not designated, the center will automatically be calculated based on the distribution of image dither positions.

INSTRUMENT PARAMETERS :

gain : float (Default Value = None)

Value used to override instrument specific default gain values. The value is assumed to be in units of electrons/count. This parameter should not be populated if the ''gainkeyword" parameter is in use.

gainkeyword : str (Default Value = '')

Keyword used to specify a value, which is used to override the instrument specific default gain values. The value is assumed to be in units of electrons/count. This parameter should not be populated if the ''gain" parameter is in use.

rdnoise : float (Default Value = None)

Value used to override instrument specific default readnoise values. The value is assumed to be in units of electrons. This parameter should not be populated if the ''rnkeyword" parameter is in use.

rnkeyword : str (Default Value = '')

Keyword used to specify a value, which is used to override the instrument specific default readnoise values. The value is assumed to be in units of electrons. This parameter should not be populated if the ''rdnoise" parameter is in use.

exptime : float (Default Value = None)

Value used to override default exposure time image header values. The value is assumed to be in units of seconds. This parameter should not be populated if the ''expkeyword" parameter is in use.

expkeyword : str (Default Value = '')

Keyword used to specify a value, which is used to override the default exposure time image header values. The value is assumed to be in units of seconds. This parameter should not be populated if the ''exptime" parameter is in use.

See also:

drizzlepac.adrizzle
Apply the 'drizzle' algorithm to the images

drizzlepac.ablot
Apply the 'blot' algorithm to drizzled images

drizzlepac.sky
Perform sky subtraction

drizzlepac.median
Create a median combined image from a set of drizzled images

drizzlepac.drizCR
Identify cosmic-rays by comparing blotted, median images to the original input images

Notes

Something to keep in mind is that the full AstroDrizzle interface will make backup copies of your original files and place them in the **OrIg_files/** directory of you current working directory.

Examples

The astrodrizzle task can be run from either the TEAL GUI or from the command-line using PyRAF or Python. These examples illustrate the various syntax options available.

Example 1: Drizzle a set of calibrated (_flt.fits) images using mostly default parameters. Select the 'World Coordinate System' keyword to the updated solution computed from 'tweakreg'. When combining images, ignore pixel flags of 64 and 32 in the DQ array of the (_flt.fits) images. Align the final product such that North is up, and set the final pixel scale to 0.05 arcseconds/pixel.

1. Run the task from PyRAF using the TEAL GUI:

```
--> import drizzlepac
--> epar astrodrizzle
```

2. Run the task from PyRAF using the command line.

```
--> import drizzlepac
--> from drizzlepac import astrodrizzle
--> astrodrizzle.AstroDrizzle('*flt.fits', output='final', wcskey='TWEAK', \
    driz_sep_bits='64,32', final_wcs=True, final_scale=0.05, final_rot=0)
```

   Or, run the same task from the PyRAF command line, but specify all parameters in a config file named "myparam.cfg":

```
--> astrodrizzle.AstroDrizzle('*flt.fits', configobj='myparam.cfg')
```

3. Run the task directly from Python:

```
>>> from drizzlepac import astrodrizzle
>>> astrodrizzle.AstroDrizzle('*flt.fits', output='final', wcskey='TWEAK', \
    driz_sep_bits='64,32', final_wcs=True, final_scale=0.05, final_rot=0)
```

4. Help can be accessed via the "Help" pulldown menu in the TEAL GUI. It can also be accessed from the PyRAF command-line and saved to a text file:

```
--> from drizzlepac import astrodrizzle
--> astrodrizzle.help()
```

   or:

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

CHAPTER

TWO

IMAGEOBJECT CLASSES

This class and related sub-classes manage all the instrument-specific images for processing by AstroDrizzle.

2.1 Base ImageObject Classes

class drizzlepac.imageObject.**baseImageObject**(*filename*)

Bases: object

Base ImageObject which defines the primary set of methods.

buildERRmask(*chip*, *dqarr*, *scale*)

Builds a weight mask from an input DQ array and an ERR array associated with the input image.

buildEXPmask(*chip*, *dqarr*)

Builds a weight mask from an input DQ array and the exposure time per pixel for this chip.

buildIVMmask(*chip*, *dqarr*, *scale*)

Builds a weight mask from an input DQ array and either an IVM array provided by the user or a self-generated IVM array derived from the flat-field reference file associated with the input image.

buildMask(*chip*, *bits=0*, *write=False*)

Build masks as specified in the user parameters found in the configObj object.

We should overload this function in the instrument specific implementations so that we can add other stuff to the badpixel mask? Like vignetting areas and chip boundries in nicmos which are camera dependent? these are not defined in the DQ masks, but should be masked out to get the best results in multidrizzle.

clean()

Deletes intermediate products generated for this imageObject.

close()

Close the object nicely and release all the data arrays from memory YOU CANT GET IT BACK, the pointers and data are gone so use the getData method to get the data array returned for future use. You can use putData to reattach a new data array to the imageObject.

findExtNum(*extname=None*, *extver=1*)

Find the extension number of the give extname and extver.

find_DQ_extension()

Return the suffix for the data quality extension and the name of the file which that DQ extension should be read from.

getAllData(*extname=None*, *exclude=None*)

This function is meant to make it easier to attach ALL the data extensions of the image object so that we can write out copies of the original image nicer.

If no extname is given, the it retrieves all data from the original file and attaches it. Otherwise, give the name of the extensions you want and all of those will be restored.

Ok, I added another option. If you want to get all the data extensions EXCEPT a particular one, leave extname=NONE and set exclude=EXTNAME. This is helpfull cause you might not know all the extnames the image has, this will find out and exclude the one you do not want overwritten.

getData(*exten=None*)
Return just the data array from the specified extension fileutil is used instead of pyfits to account for non-FITS input images. openImage returns a pyfits object.

getExtensions(*extname='SCI'*, *section=None*)
Return the list of EXTVER values for extensions with name specified in extname.

getGain(*exten*)

getHeader(*exten=None*)
Return just the specified header extension fileutil is used instead of pyfits to account for non-FITS input images. openImage returns a pyfits object.

getInstrParameter(*value*, *header*, *keyword*)
This method gets a instrument parameter from a pair of task parameters: a value, and a header keyword.

The default behavior is:

- if the value and header keyword are given, raise an exception.
- if the value is given, use it.
- if the value is blank and the header keyword is given, use the header keyword.
- if both are blank, or if the header keyword is not found, return None.

getKeywordList(*kw*)
Return lists of all attribute values for all active chips in the imageObject.

getNumpyType(*irafType*)
Return the corresponding numpy data type.

getOutputName(*name*)
Return the name of the file or PyFITS object associated with that name, depending on the setting of self.inmemory.

getReadNoiseImage(*chip*)

Notes

Method for returning the readnoise image of a detector (in electrons).

The method will return an array of the same shape as the image.

Units
electrons

getdarkcurrent()

Notes

Return the dark current for the detector. This value will be contained within an instrument specific keyword. The value in the image header will be converted to units of electrons.

Units
electrons

getdarkimg(*chip*)

Notes

Return an array representing the dark image for the detector.

The method will return an array of the same shape as the image.

Units
electrons

getexptimeimg(*chip*)

Returns
exptimeimg : numpy array

The method will return an array of the same shape as the image.

Notes

Return an array representing the exposure time per pixel for the detector. This method will be overloaded for IR detectors which have their own EXP arrays, namely, WFC3/IR and NICMOS images.

Units
None

getflat(*chip*)
Method for retrieving a detector's flat field.

Returns
flat: array :

This method will return an array the same shape as the image in **units of electrons**.

getskyimg(*chip*)

Notes

Return an array representing the sky image for the detector. The value of the sky is what would actually be subtracted from the exposure by the skysub step.

Units
electrons

info()
Return fits information on the _image.

putData(*data=None*, *exten=None*)
Now that we are removing the data from the object to save memory, we need something that cleanly puts the data array back into the object so that we can write out everything together using something like pyfits.writeto....this method is an attempt to make sure that when you add an array back to the .data section of the hdu it still matches the header information for that section (ie. update the bitpix to reflect the datatype of the array you are adding). The other header stuff is up to you to verify.

Data should be the data array exten is where you want to stick it, either extension number or a string like 'sci,1'

returnAllChips (*extname=None*, *exclude=None*)
Returns a list containing all the chips which match the extname given minus those specified for exclusion (if any).

saveVirtualOutputs (*outdict*)
Assign in-memory versions of generated products for this imageObject based on dictionary 'outdict'.

set_mt_wcs (*image*)
Reset the WCS for this image based on the WCS information from another imageObject.

set_units ()
Record the units for this image, both BUNITS from header and in_units as needed internally. This method will be defined specifically for each instrument.

set_wtscl (*chip*, *wtscl_par*)
Sets the value of the wt_scl parameter as needed for drizzling.

updateContextImage (*contextpar*)
Reset the name of the context image to None if parameter *context* == False.

updateData (*exten*, *data*)
Write out updated data and header to the original input file for this object.

updateIVMName (*ivmname*)
Update outputNames for image with user-supplied IVM filename.

updateOutputValues (*output_wcs*)
Copy info from output WCSObject into outputnames for each chip for use in creating outputimage object.

class drizzlepac.imageObject.**imageObject** (*filename*, *group=None*, *inmemory=False*)
Bases: drizzlepac.imageObject.baseImageObject

This returns an imageObject that contains all the necessary information to run the image file through any multidrizzle function. It is essentially a PyFits object with extra attributes.

There will be generic keywords which are good for the entire image file, and some that might pertain only to the specific chip.

compute_wcslin (*undistort=True*)
Compute the undistorted WCS based solely on the known distortion model information associated with the WCS.

setInstrumentParameters (*instrpars*)
Define instrument-specific parameters for use in the code. By definition, this definition will need to be overridden by methods defined in each instrument's sub-class.

set_units (*chip*)
Define units for this image.

class drizzlepac.imageObject.**WCSObject** (*filename*, *suffix='_drz'*)
Bases: drizzlepac.imageObject.baseImageObject

restore_wcs ()

2.2 ACS ImageObjects

class drizzlepac.acsData.**ACSInputImage** (*filename=None*, *group=None*)
Bases: drizzlepac.imageObject.imageObject

doUnitConversions()

getdarkcurrent(*extver*)
Return the dark current for the ACS detector. This value will be contained within an instrument specific keyword. The value in the image header will be converted to units of electrons.

Returns
darkcurrent: float :

Dark current value for the ACS detector in **units of electrons**.

SEPARATOR = '_'

class drizzlepac.acsData.**WFCInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.acsData.ACSInputImage

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

This method gets called from processInput.

class drizzlepac.acsData.**HRCInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.acsData.ACSInputImage

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

This method gets called from processInput.

class drizzlepac.acsData.**SBCInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.acsData.ACSInputImage

setInstrumentParameters(*instrpars*)
Sets the instrument parameters.

2.3 WFC3 ImageObjects

class drizzlepac.wfc3Data.**WFC3InputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.imageObject.imageObject

SEPARATOR = '_'

class drizzlepac.wfc3Data.**WFC3UVISInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.wfc3Data.WFC3InputImage

doUnitConversions()

getdarkcurrent(*chip*)
Return the dark current for the WFC3 UVIS detector. This value will be contained within an instrument specific keyword.

Returns
darkcurrent: float :

The dark current value with **units of electrons**.

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

class drizzlepac.wfc3Data.**WFC3IRInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.wfc3Data.WFC3InputImage

doUnitConversions()
WF3 IR data come out in electrons, and I imagine the photometry keywords will be calculated as such, so no image manipulation needs be done between native and electrons

getdarkcurrent()
Return the dark current for the WFC3/IR detector. This value will be contained within an instrument specific keyword.

Returns
darkcurrent: float :

The dark current value in **units of electrons**.

getdarkimg(*chip*)
Return an array representing the dark image for the detector.

Returns
dark: array :

Dark image array in the same shape as the input image with **units of cps**

getexptimeimg(*chip*)
Return an array representing the exposure time per pixel for the detector.

Returns
dark: array :

Exposure time array in the same shape as the input image

getskyimg(*chip*)

Notes

Return an array representing the sky image for the detector. The value of the sky is what would actually be subtracted from the exposure by the skysub step.

Units
electrons

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

2.4 STIS ImageObjects

class drizzlepac.stisData.**STISInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.imageObject.imageObject

doUnitConversions()
Convert the data to electrons.

This converts all science data extensions and saves the results back to disk. We need to make sure the data inside the chips already in memory is altered as well.

getflat(*chip*)
Method for retrieving a detector's flat field. For STIS there are three. This method will return an array the same shape as the image.

SEPARATOR = '_'

class drizzlepac.stisData.**CCDInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.stisData.STISInputImage

getReadNoise()
Method for returning the readnoise of a detector (in DN).

Units
DN

This should work on a chip, since different chips to be consistant with other detector classes where different chips have different gains.

getdarkcurrent()
Returns the dark current for the STIS CCD chip.

Returns
darkcurrent : float

Dark current value in **units of electrons** (or counts, if proc_unit=='native').

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

class drizzlepac.stisData.**NUVInputImage**(*input*, *dqname*, *platescale*, *memmap=0*, *proc_unit='native'*)
Bases: drizzlepac.stisData.STISInputImage

getdarkcurrent()
Returns the dark current for the STIS NUV detector.

Returns
darkcurrent : float

Dark current value in **units of electrons** (or counts, if proc_unit=='native').

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

class drizzlepac.stisData.**FUVInputImage**(*filename=None*, *group=None*)
Bases: drizzlepac.stisData.STISInputImage

getdarkcurrent()
Returns the dark current for the STIS FUV detector.

Returns
darkcurrent : float

Dark current value in **units of electrons** (or counts, if proc_unit=='native').

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

2.5 NICMOS ImageObjects

class drizzlepac.nicmosData.**NICMOSInputImage**(*filename=None*)
Bases: drizzlepac.imageObject.imageObject

doUnitConversions()
Convert the data to electrons

This converts all science data extensions and saves the results back to disk. We need to make sure the data inside the chips already in memory is altered as well.

getdarkcurrent()
Return the dark current for the NICMOS detectors.

Returns
darkcurrent : float

Dark current value with **units of cps**.

getdarkimg(*chip*)
Return an array representing the dark image for the detector.

Returns
dark : array

The dark array in the same shape as the image with **units of cps**.

getexptimeimg(*chip*)
Return an array representing the exposure time per pixel for the detector.

Returns
dark: array :

Exposure time array in the same shape as the input image

getflat(*chip*)
Method for retrieving a detector's flat field.

Returns
flat : array

The flat field array in the same shape as the input image with **units of cps**.

isCountRate()
isCountRate: Method or IRInputObject used to indicate if the science data is in units of counts or count rate. This method assumes that the keyword 'BUNIT' is in the header of the input FITS file.

SEPARATOR = '_'

class drizzlepac.nicmosData.**NIC1InputImage**(*filename=None*)
Bases: drizzlepac.nicmosData.NICMOSInputImage

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

class drizzlepac.nicmosData.**NIC2InputImage**(*filename=None*)
Bases: drizzlepac.nicmosData.NICMOSInputImage

createHoleMask()
Add in a mask for the coronographic hole to the general static pixel mask.

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

class drizzlepac.nicmosData.**NIC3InputImage**(*filename=None*)
Bases: drizzlepac.nicmosData.NICMOSInputImage

setInstrumentParameters(*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

2.6 WFPC2 ImageObjects

class drizzlepac.wfpc2Data.**WFPC2InputImage**(*filename*, *group=None*)
Bases: drizzlepac.imageObject.imageObject

buildMask(*chip*, *bits=0*, *write=False*)
Build masks as specified in the user parameters found in the configObj object.

doUnitConversions()
Apply unit conversions to all the chips, ignoring the group parameter. This insures that all the chips get the same conversions when this gets done, even if only 1 chip was specified to be processed.

find_DQ_extension()
Return the suffix for the data quality extension and the name of the file which that DQ extension should be read from.

getEffGain()
Method used to return the effective gain of a instrument's detector.

Returns
gain : float

The effective gain.

getReadNoise(*exten*)
Method for returning the readnoise of a detector (in counts).

Returns
readnoise : float

The readnoise of the detector in **units of counts/electrons**.

getdarkcurrent(*exten*)
Return the dark current for the WFPC2 detector. This value will be contained within an instrument specific keyword. The value in the image header will be converted to units of electrons.

Returns
darkcurrent : float

Dark current for the WFPC3 detector in **units of counts/electrons**.

getflat(*chip*)
Method for retrieving a detector's flat field.

Returns
flat : array

The flat-field array in the same shape as the input image.

setInstrumentParameters (*instrpars*)
This method overrides the superclass to set default values into the parameter dictionary, in case empty entries are provided.

SEPARATOR = '_'

CHAPTER

THREE

STEP 1: PROCESS INPUT

This module supports the interpretation and initial verification of all the input files specified by the user. These functions:

- reads in parameter values from MDRIZTAB reference file and merges those merges those values in with the rest of the parameters from the GUI/configObj, if use of MDRIZTAB was specified
- insure that all input files are multi-extension FITS files and converts them if they are not
- updates all input WCS's to be consistent with IDCTAB, if specified
- generates the ImageObject instances for each input file
- resets the DQ bits if specified by the user
- adds info about any user-provided IVM files to the ImageObjects
- generates the output WCS based on user inputs

Process input to MultiDrizzle/PyDrizzle.

The input can be one of:

- a python list of files
- a comma separated string of filenames (including wild card characters)
- an association table
- an @file (can have a second column with names of ivm files)

No mixture of instruments is allowed. No mixture of association tables, @files and regular fits files is allowed. Files can be in GEIS or MEF format (but not waiver fits).

Runs some sanity checks on the input files. If necessary converts files to MEF format (this should not be left to makewcs because 'updatewcs' may be False). Runs makewcs. The function 'process_input' returns an association table, ivmlist, output name

The common interface interpreter for MultiDrizzle tasks, 'processCommonInput()', not only runs 'process_input()' but 'createImageObject()' and 'defineOutput()' as well to fully setup all inputs for use with the rest of the MultiDrizzle steps either as stand-alone tasks or internally to MultiDrizzle itself.

`drizzlepac.processInput.`**`addIVMInputs`**(*imageObjectList*, *ivmlist*)
Add IVM filenames provided by user to outputNames dictionary for each input imageObject.

`drizzlepac.processInput.`**`applyContextPar`**(*imageObjectList*, *contextpar*)
Apply the value of the parameter *context* to the input list, setting the name of the output context image to None if *context* is False

`drizzlepac.processInput.`**`buildASNList`**(*rootnames*, *asnname*)
Return the list of filenames for a given set of rootnames

drizzlepac.processInput.**buildEmptyDRZ**(*input*, *output*)
Create an empty DRZ file.

This module creates an empty DRZ file in a valid FITS format so that the HST pipeline can handle the Multidrizzle zero expossure time exception where all data has been excluded from processing.

Parameters
input : str

filename of the initial input to process_input

output : str

filename of the default empty _drz.fits file to be generated

drizzlepac.processInput.**buildFileList**(*input*, *output=None*, *ivmlist=None*, *wcskey=None*, *updatewcs=True*, ***workinplace*)
Builds a file list which has undergone various instrument-specific checks for input to MultiDrizzle, including splitting STIS associations.

drizzlepac.processInput.**changeSuffixinASN**(*asnfile*, *suffix*)
Create a copy of the original asn file and change the name of all members to include the suffix.

drizzlepac.processInput.**checkDGEOFile**(*filenames*)
Verify that input file has been updated with NPOLFILE

This function checks for the presence of 'NPOLFILE' kw in the primary header when 'DGEOFILE' kw is present and valid (i.e. 'DGEOFILE' is not blank or 'N/A'). It handles the case of science files downloaded from the archive before the new software was installed there. If 'DGEOFILE' is present and 'NPOLFILE' is missing, print a message and let the user choose whether to (q)uit and update the headers or (c)ontinue and run astrodrizzle without the non-polynomial correction. 'NPOLFILE' will be populated in the pipeline before astrodrizzle is run.

In the case of WFPC2 the old style dgeo files are used to create detector to image correction at runtime.

Parameters
filenames : list of str

file names of all images to be checked

drizzlepac.processInput.**checkForDuplicateInputs**(*rootnames*)
Check input files specified in ASN table for duplicate versions with multiple valid suffixes (_flt and _flc, for example).

drizzlepac.processInput.**checkMultipleFiles**(*input*)
Evaluates the input to determine whether there is 1 or more than 1 valid input file.

drizzlepac.processInput.**convert_dgeo_to_d2im**(*dgeofile*, *output*, *clobber=True*)
Routine that converts the WFPC2 DGEOFILE into a D2IMFILE.

drizzlepac.processInput.**createImageObjectList**(*files*, *instrpars*, *group=None*, *undistort=True*, *inmemory=False*)
Returns a list of imageObject instances, 1 for each input image in the list of input filenames.

drizzlepac.processInput.**getMdriztabPars**(*input*)
High-level function for getting the parameters from MDRIZTAB

Used primarily for TEAL interface.

drizzlepac.processInput.**manageInputCopies**(*filelist*, ***workinplace*)
Creates copies of all input images in a sub-directory.

The copies are made prior to any processing being done to the images at all, including updating the WCS keywords. If there are already copies present, they will NOT be overwritten, but instead will be used to overwrite the current working copies.

drizzlepac.processInput.**processFilenames**(*input=None*, *output=None*, *infilesOnly=False*)
Process the input string which contains the input file information and return a filelist,output

drizzlepac.processInput.**process_input**(*input*, *output=None*, *ivmlist=None*, *updatewcs=True*, *prodonly=False*, *wcskey=None*, ***workinplace*)
Create the full input list of filenames after verifying and converting files as needed.

drizzlepac.processInput.**reportResourceUsage**(*imageObjectList*, *outwcs*, *num_cores*, *interactive=False*)
Provide some information to the user on the estimated resource usage (primarily memory) for this run.

drizzlepac.processInput.**resetDQBits**(*imageObjectList*, *cr_bits_value=4096*)
Reset the CR bit in each input image's DQ array

drizzlepac.processInput.**runmakewcs**(*input*)
Runs make wcs and recomputes the WCS keywords

Parameters
input : str or list of str

a list of files

Returns
output : list of str

returns a list of names of the modified files (For GEIS files returns the translated names.)

drizzlepac.processInput.**setCommonInput**(*configObj*, *createOutwcs=True*)
The common interface interpreter for MultiDrizzle tasks which not only runs 'process_input()' but 'createImageObject()' and 'defineOutput()' as well to fully setup all inputs for use with the rest of the MultiDrizzle steps either as stand-alone tasks or internally to MultiDrizzle itself.

Parameters
configObj : object

configObj instance or simple dictionary of input parameters

imageObjectList : list of imageObject objects

list of imageObject instances, 1 for each input exposure

outwcs : object

imageObject instance defining the final output frame

Notes

At a minimum, the configObj instance (dictionary) should contain:

configObj = {'input':None,'output':None,
'updatewcs':None}

If provided, the configObj should contain the values of all the multidrizzle parameters as set by the user with TEAL. If no configObj is given, it will retrieve the default values automatically. In either case, the values from the input_dict will be merged in with the configObj before being used by the rest of the code.

Examples

You can set *createOutwcs=False* for the cases where you only want the images processed and no output wcs information in necessary; as in:

>>>imageObjectList,outwcs = processInput.processCommonInput(configObj)

drizzlepac.processInput.**update_member_names**(*oldasndict*, *pydr_input*)
Update names in a member dictionary.

Given an association dictionary with rootnames and a list of full file names, it will update the names in the member dictionary to contain '_*' extension. For example a rootname of 'u9600201m' will be replaced by 'u9600201m_c0h' making sure that a MEf file is passed as an input and not the corresponding GEIS file.

drizzlepac.processInput.**update_wfpc2_d2geofile**(*filename*, *fhdu=None*)
Creates a D2IMFILE from the DGEOFILE for a WFPC2 image (input), and modifies the header to reflect the new usage.

drizzlepac.processInput.**userStop**(*message*)

3.1 ResetBits Update of Input

This module provides the capability to set a specific set of bit values in the input DQ arrays to zero. This allows a user to reset pixels previously erroneously flagged as cosmic-rays to good for reprocessing with improved alignment or detection parameters. resetbits - A module to set the value of specified array pixels to zero

This module allows a user to reset the pixel values of any integer array, such as the DQ array from an HST image, to zero.

License::
http://www.stsci.edu/resources/software_hardware/pyraf/LICENSE

3.1.1 PARAMETERS

filename
[str] full filename with path

bits
[str] sum or list of integers corresponding to all the bits to be reset

3.1.2 Optional Parameters

extver
[int, optional] List of version numbers of the arrays to be corrected (default: None, will reset all matching arrays)

extname
[str, optional] EXTNAME of the arrays in the FITS files to be reset (default: 'dq')

3.1.3 NOTES

This module performs a simple bitwise-and on all the pixels in the specified array and the integer value provided as input using the operation (array & ~bits).

3.1.4 Usage

It can be called not only from within Python, but also from the host-level operating system command line using the syntax:

```
resetbits filename bits [extver [extname]]
```

3.1.5 EXAMPLES

1. The following command will reset the 4096 bits in all the DQ arrays of the file 'input_flt.fits':

   ```
   resetbits input_flt.fits 4096
   ```

   or from the Python command line:

   ```
   >>> import resetbits
   >>> resetbits.reset_dq_bits("input_file_flt.fits", 4096)
   ```

2. To reset the 2,32,64 and 4096 (sum of 4194) bits in the second DQ array, specified as 'dq,2', in the file 'input_flt.fits':

   ```
   resetbits input_flt.fits 4194 2
   ```

   or from the Python command line:

   ```
   >>> import resetbits
   >>> resetbits.reset_dq_bits("input_file_flt.fits", 2+32+64+4096, extver=2)
   ```

drizzlepac.resetbits.**help**(*file=None*)
Print out syntax help for running resetbits

Parameters
file : str (Default = None)

If given, write out help to the filename specified by this parameter Any previously existing file with this name will be deleted before writing out the help.

drizzlepac.resetbits.**reset_dq_bits**(*input*, *bits*, *extver=None*, *extname='dq'*)
This function resets bits in the integer array(s) of a FITS file.

Parameters
filename : str

full filename with path

bits : str

sum or list of integers corresponding to all the bits to be reset

extver : int, optional

List of version numbers of the DQ arrays to be corrected [Default Value: None, will do all]

extname : str, optional

EXTNAME of the DQ arrays in the FITS file [Default Value: 'dq']

Notes

The default value of None for the 'extver' parameter specifies that all extensions with EXTNAME matching 'dq' (as specified by the 'extname' parameter) will have their bits reset.

Examples

1. The following command will reset the 4096 bits in all the DQ arrays of the file input_file_flt.fits:

```
reset_dq_bits("input_file_flt.fits", 4096)
```

2. To reset the 2,32,64 and 4096 bits in the second DQ array, specified as 'dq,2', in the file input_file_flt.fits:

```
reset_dq_bits("input_file_flt.fits", "2,32,64,4096", extver=2)
```

drizzlepac.resetbits.**run**(*configobj=None*)
Teal interface for running this code.

CHAPTER

FOUR

STEP 2: GENERATE A STATIC MASK

This step focuses entirely on creating a static mask for each detector used in the input images of all negative (presumably bad) pixel values. This class manages the creation of the global static mask which masks pixels which are negative in the SCI array. A static mask numpy object gets created for each global mask needed, one for each chip from each instrument/detector. Each static mask array has type Int16, and resides in memory.

Authors
Ivo Busko, Christopher Hanley, Warren Hack, Megan Sosey

Program
staticMask.py

Notes
Class that manages the creation of a global static mask which is used to mask pixels that are some sigma BELOW the mode computed for the image.

class drizzlepac.staticMask.**staticMask**(*configObj=None*)
This class manages the creation of the global static mask which masks pixels that are unwanted in the SCI array. A static mask object gets created for each global mask needed, one for each chip from each instrument/detector. Each static mask array has type Int16, and resides in memory.

addMember(*imagePtr=None*)
Combines the input image with the static mask that has the same signature.

Parameters
imagePtr : object

An imageObject reference

Notes

The signature parameter consists of the tuple:

```
(instrument/detector, (nx,ny), chip_id)
```

The signature is defined in the image object for each chip

close()
Deletes all static mask objects.

deleteMask(*signature*)
Delete just the mask that matches the signature given.

getFilename(*signature*)
Returns the name of the output mask file that should reside on disk for the given signature.

getMaskArray(*signature*)
Returns the appropriate StaticMask array for the image.

getMaskname(*chipid*)
Construct an output filename for the given signature:

```
signature=[instr+detector,(nx,ny),detnum]
```

The signature is in the image object and the name of the static mask file is saved as sci_chip.outputNames["staticMask"].

saveToFile(*imageObjectList*)
Saves the static mask to a file it uses the signatures associated with each mask to contruct the filename for the output mask image.

drizzlepac.staticMask.**buildSignatureKey**(*signature*)
Build static file filename suffix used by mkstemp()

drizzlepac.staticMask.**constructFilename**(*signature*)
Construct an output filename for the given signature:

```
signature=[instr+detector,(nx,ny),detnum]
```

The signature is in the image object.

drizzlepac.staticMask.**createMask**(*input=None*, *static_sig=4.0*, *group=None*, *editpars=False*, *configObj=None*, ***inputDict*)
The user can input a list of images if they like to create static masks as well as optional values for static_sig and inputDict.

The configObj.cfg file will set the defaults and then override them with the user options.

drizzlepac.staticMask.**createStaticMask**(*imageObjectList=[]*, *configObj=None*, *procSteps=None*)

drizzlepac.staticMask.**getHelpAsString**()
Return useful help from a file in the script directory called module.help

drizzlepac.staticMask.**help**()

drizzlepac.staticMask.**run**(*configObj*)

CHAPTER

FIVE

STEP 3: SUBTRACTING THE SKY

This step measures and subtracts the sky from each input image while recording the subtracted value in the image header. Function for computing and subtracting the backgroud of an image. The algorithm employed here uses a sigma clipped median of each *sci* image in a data file. Then the sky value for each detector is compared and the lowest value is subtracted from all chips in the detector. Finally, the MDRIZSKY keyword is updated in the header of the input files.

Authors

Christopher Hanley, Megan Sosey

drizzlepac.sky.**getHelpAsString**()

return useful help from a file in the script directory called module.help

drizzlepac.sky.**getreferencesky**(*image*, *keyval*)

drizzlepac.sky.**help**()

drizzlepac.sky.**run**(*configObj*, *outExt=None*)

drizzlepac.sky.**sky**(*input=None*, *outExt=None*, *configObj=None*, *group=None*, *editpars=False*, ***inputDict*)

Perform sky subtraction on input list of images

Parameters

input : str or list of str

a python list of image filenames, or just a single filename

configObj : configObject

an instance of configObject

inputDict : dict, optional

an optional list of parameters specified by the user

outExt : str

The extension of the output image. If the output already exists then the input image is overwritten

Notes

These are parameters that the configObj should contain by default, they can be altered on the fly using the inputDict

Parameters that should be in configobj:

Name	Definition
skyuser	'KEYWORD in header which indicates a sky subtraction value to use'.
skysub	'Perform sky subtraction?'
skywidth	'Bin width for sampling sky statistics (in sigma)'
skystat	'Sky correction statistics parameter'
skylower	'Lower limit of usable data for sky (always in electrons)'
skyupper	'Upper limit of usable data for sky (always in electrons)'
skyclip	'Number of clipping iterations'
skylsigma	'Lower side clipping factor (in sigma)'
skyusigma	'Upper side clipping factor (in sigma)'

The output from sky subtraction is a copy of the original input file where all the science data extensions have been sky subtracted.

drizzlepac.sky.**subtractSky**(*imageObjList*, *configObj*, *saveFile=False*, *procSteps=None*)

CHAPTER

SIX

STEP 4 AND 8: DRIZZLING THE IMAGES

The operation of drizzling each input image needs to be performed twice during processing:

- single drizzle step: this initial step drizzles each image onto the final output WCS as separate images
- final drizzle step: this step produces the final combined image based on the cosmic-ray masks determined by MultiDrizzle

drizzlepac.adrizzle.**buildDrizParamDict**(*configObj*, *single=True*)

drizzlepac.adrizzle.**create_output**(*filename*)

drizzlepac.adrizzle.**do_driz**(*insci*, *input_wcs*, *inwht*, *output_wcs*, *outsci*, *outwht*, *outcon*, *expin*, *in_units*, *wt_scl*, *wcslin_pscale=1.0*, *uniqid=1*, *pixfrac=1.0*, *kernel='square'*, *fillval='INDEF'*, *stepsize=10*, *wcsmap=None*)

Core routine for performing 'drizzle' operation on a single input image All input values will be Python objects such as ndarrays, instead of filenames File handling (input and output) will be performed by calling routine.

drizzlepac.adrizzle.**drizFinal**(*imageObjectList*, *output_wcs*, *configObj*, *build=None*, *wcsmap=None*, *procSteps=None*)

drizzlepac.adrizzle.**drizSeparate**(*imageObjectList*, *output_wcs*, *configObj*, *wcsmap=None*, *procSteps=None*)

drizzlepac.adrizzle.**drizzle**(*input*, *outdata*, *wcsmap=None*, *editpars=False*, *configObj=None*, ***input_dict*)

drizzlepac.adrizzle.**getHelpAsString**()

Return useful help from a file in the script directory called module.help

drizzlepac.adrizzle.**get_data**(*filename*)

drizzlepac.adrizzle.**help**()

drizzlepac.adrizzle.**mergeDQarray**(*maskname*, *dqarr*)

Merge static or CR mask with mask created from DQ array on-the-fly here.

drizzlepac.adrizzle.**run**(*configObj*, *wcsmap=None*)

Interface for running 'wdrizzle' from TEAL or Python command-line.

This code performs all file I/O to set up the use of the drizzle code for a single exposure to replicate the functionality of the original 'wdrizzle'.

drizzlepac.adrizzle.**run_driz**(*imageObjectList*, *output_wcs*, *paramDict*, *single*, *build*, *wcsmap=None*)

Perform drizzle operation on input to create output. The input parameters originally was a list of dictionaries, one for each input, that matches the primary parameters for an IRAF drizzle task.

This method would then loop over all the entries in the list and run 'drizzle' for each entry.

Parameters required for input in paramDict:

build,single,units,wt_scl,pixfrac,kernel,fillval, rot,scale,xsh,ysh,blotnx,blotny,outnx,outny,data

drizzlepac.adrizzle.**run_driz_chip**(*img*, *virtual_outputs*, *chip*, *output_wcs*, *outwcs*, *template*, *paramDict*, *single*, *doWrite*, *build*, *_versions*, *_numctx*, *_nplanes*, *_numchips*, *_outsci*, *_outwht*, *_outctx*, *_hdrlist*, *wcsmap*)

Perform the drizzle operation on a single chip. This is separated out from run_driz_img() so as to keep together the entirety of the code which is inside the loop over chips. See the run_driz() code for more documentation.

drizzlepac.adrizzle.**run_driz_img**(*img*, *virtual_outputs*, *chiplist*, *output_wcs*, *outwcs*, *template*, *paramDict*, *single*, *num_in_prod*, *build*, *_versions*, *_numctx*, *_nplanes*, *chipIdxCopy*, *_outsci*, *_outwht*, *_outctx*, *_hdrlist*, *wcsmap*)

Perform the drizzle operation on a single image. This is separated out from run_driz() so as to keep together the entirety of the code which is inside the loop over images. See the run_driz() code for more documentation.

drizzlepac.adrizzle.**updateInputDQArray**(*dqfile*, *dq_extn*, *chip*, *crmaskname*, *cr_bits_value*)

CHAPTER

SEVEN

STEP 5: CREATE A MEDIAN IMAGE

A median image gets generated from the stack of undistorted single drizzle images.

drizzlepac.createMedian.**createMedian**(*imgObjList*, *configObj*, *procSteps=None*)
Top-level interface to createMedian step called from top-level MultiDrizzle.

This function parses the input parameters then calls the *_median()* function to median-combine the input images into a single image.

drizzlepac.createMedian.**getHelpAsString**()
Return useful help from a file in the script directory called module.help

drizzlepac.createMedian.**median**(*input=None*, *configObj=None*, *editpars=False*, ***inputDict*)
Create a median image from the seperately drizzled images.

drizzlepac.createMedian.**run**(*configObj*)

CHAPTER

EIGHT

STEP 6: BLOTTING THE MEDIAN IMAGE

The median image now gets blotted back to create median-cleaned images which can be compared directly with each input image to identify cosmic-rays.

drizzlepac.ablot.**blot**(*data*, *outdata*, *configObj=None*, *wcsmap=<class drizzlepac.wcs_functions.WCSMap at 0x10478c3f8>*, *editpars=False*, ***input_dict*)

drizzlepac.ablot.**buildBlotParamDict**(*configObj*)

drizzlepac.ablot.**do_blot**(*source*, *source_wcs*, *blot_wcs*, *exptime*, *coeffs=True*, *interp='poly5'*, *sinscl=1.0*, *stepsize=10*, *wcsmap=None*)

Core functionality of performing the 'blot' operation to create a single blotted image from a single source image. All distortion information is assumed to be included in the WCS specification of the 'output' blotted image given in 'blot_wcs'.

Parameters

source :

Input numpy array of undistorted source image in units of 'cps'.

source_wcs :

HSTWCS object representing source image WCS.

blot_wcs :

HSTWCS object representing the blotted image WCS.

exptime :

drizzlepac.ablot.**getHelpAsString**()

Return useful help from a file in the script directory called module.help

drizzlepac.ablot.**help**()

drizzlepac.ablot.**run**(*configObj*, *wcsmap=None*)

Run the blot task based on parameters provided interactively by the user.

drizzlepac.ablot.**runBlot**(*imageObjectList*, *output_wcs*, *configObj={}*, *wcsmap=<class drizzlepac.wcs_functions.WCSMap at 0x10478c3f8>*, *procSteps=None*)

drizzlepac.ablot.**run_blot**(*imageObjectList*, *output_wcs*, *paramDict*, *wcsmap=<class drizzlepac.wcs_functions.WCSMap at 0x10478c3f8>*)

Perform the blot operation on the list of images.

CHAPTER

NINE

STEP 7: COSMIC-RAY IDENTIFICATION

The cosmic rays and bad pixels are now identified by comparing the input images with the associated blotted,median-cleaned images created.

drizzlepac.drizCR.**createCorrFile**(*outfile*, *arrlist*, *template*)
Create a _cor file with the same format as the original input image

The DQ array will be replaced with the mask array used to create the _cor file.

drizzlepac.drizCR.**drizCR**(*input=None*, *configObj=None*, *editpars=False*, ***inputDict*)
Look for cosmic rays.

drizzlepac.drizCR.**getHelpAsString**()
Return useful help from a file in the script directory called module.help

drizzlepac.drizCR.**run**(*configObj*)

drizzlepac.drizCR.**rundrizCR**(*imgObjList*, *configObj*, *procSteps=None*)

drizzlepac.drizCR.**setDefaults**(*configObj={}*)
Return a dictionary of the default parameters which also been updated with the user overrides.

CHAPTER

TEN

UTILITIES

Various utilities get used by Multidrizzle, including some to handle WCS interpretation, trailer file generation, output file generation and interpretation of the MDRIZTAB reference file.

10.1 Utility Functions

These functions perform various small operations within Multidrizzle. A library of utility functions

class drizzlepac.util.**ProcSteps**
This class allows MultiDrizzle to keep track of the start and end times of each processing step that gets run as well as computing/reporting the elapsed time for each step.

The code for each processing step must call the 'addStep()' method to initialize the information for that step, then the 'endStep()' method to record the end and elapsed times.

The 'reportTimes()' method can then be used to provide a summary of all the elapsed times and total run time.

addStep(*key*)
Add information about a new step to the dict of steps The value 'ptime' is the output from '_ptime()' containing both the formatted and unformatted time for the start of the step.

endStep(*key*)
Record the end time for the step.

If key==None, simply record ptime as end time for class to represent the overall runtime since the initialization of the class.

reportTimes()
Print out a formatted summary of the elapsed times for all the performed steps.

class drizzlepac.util.**WithLogging**

drizzlepac.util.**atfile_ivm**(*filename*)
Return the filename of the IVM file which is assumed to be the second word in the atfile the user gave.

drizzlepac.util.**atfile_sci**(*filename*)
Return the filename of the science image which is assumed to be the first word in the atfile the user gave.

drizzlepac.util.**check_blank**(*cvar*)
Converts blank value (from configObj?) into a value of None.

drizzlepac.util.**computeRange**(*corners*)
Determine the range spanned by an array of pixel positions.

drizzlepac.util.**compute_texptime**(*imageObjectList*)
Add up the exposure time for all the members in the pattern, since 'drizzle' doesn't have the necessary information to correctly set this itself.

drizzlepac.util.**countImages**(*imageObjectList*)

drizzlepac.util.**count_sci_extensions**(*filename*)
Return the number of SCI extensions and the EXTNAME from a input MEF file.

drizzlepac.util.**createFile**(*dataArray=None*, *outfile=None*, *header=None*)
Create a simple fits file for the given data array and header. Returns either the FITS object in-membory when outfile==None or

None when the FITS file was written out to a file.

drizzlepac.util.**displayBadRefimageWarningBox**(*display=True*, *parent=None*)
Displays a warning box for the 'input' parameter.

drizzlepac.util.**displayEmptyInputWarningBox**(*display=True*, *parent=None*)
Displays a warning box for the 'input' parameter.

drizzlepac.util.**displayMakewcsWarningBox**(*display=True*, *parent=None*)
Displays a warning box for the 'makewcs' parameter.

drizzlepac.util.**end_logging**(*filename=None*)
Close log file and restore system defaults.

drizzlepac.util.**findrootname**(*filename*)
Return the rootname of the given file.

drizzlepac.util.**getConfigObjPar**(*configObj*, *parname*)
Return parameter value without having to specify which section holds the parameter.

drizzlepac.util.**getDefaultConfigObj**(*taskname*, *configObj*, *input_dict={}*, *loadOnly=True*)
Return default configObj instance for task updated with user-specified values from input_dict.

Parameters

taskname : string

Name of task to load into TEAL

configObj : string

The valid values for 'configObj' would be:

```
None                      - loads last saved user .cfg file
'defaults'                - loads task default .cfg file
name of .cfg file (string)- loads user-specified .cfg file
```

input_dict : dict

Set of parameters and values specified by user to be different from what gets loaded in from the .cfg file for the task

loadOnly : bool

Setting 'loadOnly' to False causes the TEAL GUI to start allowing the user to edit the values further and then run the task if desired.

drizzlepac.util.**getFullParList**(*configObj*)
Return a single list of all parameter names included in the configObj regardless of which section the parameter was stored

drizzlepac.util.**getRotatedSize**(*corners*, *angle*)
Determine the size of a rotated (meta)image.

drizzlepac.util.**getSectionName**(*configObj*, *stepnum*)
Return section label based on step number.

drizzlepac.util.**get_detnum**(*hstwcs*, *filename*, *extnum*)

drizzlepac.util.**get_expstart**(*header*, *primary_hdr*)
shouldn't this just be defined in the instrument subclass of imageobject?

drizzlepac.util.**get_pool_size**(*usr_config_value=None*, *num_tasks=None*)
Determine size of thread/process-pool for parallel processing. This examines the cpu_count to decide and return the right pool size to use. Also take into account the user's wishes via the config object value, if specified. On top of that, don't allow the pool size returned to be any higher than the number of parallel tasks, if specified. Only use what we need (mp.Pool starts pool_size processes, needed or not). If number of tasks is unknown, call this with "num_tasks" set to None. Returns 1 when indicating that parallel processing should not be used. Consolidate all such logic here, not in the caller.

drizzlepac.util.**init_logging**(*logfile='astrodrizzle.log'*, *default=None*, *level=20*)
Set up logger for capturing stdout/stderr messages.

Must be called prior to writing any messages that you want to log.

drizzlepac.util.**interpret_bits_value**(*val*)
Converts input bits value from string to a single integer value or None. If a comma-separated set of values are provided, they are summed.

drizzlepac.util.**isASNTable**(*inputFilelist*)
Return TRUE if inputFilelist is a fits ASN file.

drizzlepac.util.**isCommaList**(*inputFilelist*)
Return True if the input is a comma separated list of names.

drizzlepac.util.**is_blank**(*val*)
Determines whether or not a value is considered 'blank'.

drizzlepac.util.**loadFileList**(*inputFilelist*)
Open up the '@ file' and read in the science and possible ivm filenames from the first two columns.

drizzlepac.util.**parse_colnames**(*colnames*, *coords=None*)
Convert colnames input into list of column numbers.

drizzlepac.util.**printParams**(*paramDictionary*, *all=False*, *log=None*)
Print nicely the parameters from the dictionary.

drizzlepac.util.**print_pkg_versions**(*packages=None*, *svn=False*, *log=None*)

drizzlepac.util.**readCommaList**(*fileList*)
Return a list of the files with the commas removed.

drizzlepac.util.**readcols**(*infile, cols=[0, 1, 2, 3], hms=False*)
Read the columns from an ASCII file as numpy arrays.

Parameters
infile : str

Filename of ASCII file with array data as columns.

cols : list of int

List of 0-indexed column numbers for columns to be turned into numpy arrays (DEFAULT- [0,1,2,3]).

Returns

outarr : list of numpy arrays

Simple list of numpy arrays in the order as specifed in the 'cols' parameter.

drizzlepac.util.**removeFileSafely**(*filename*, *clobber=True*)

Delete the file specified, but only if it exists and clobber is True.

drizzlepac.util.**runmakewcs**(*input*)

Runs 'updatewcs' to recompute the WCS keywords for the input image.

Parameters

input : list of str

A list of file names.

Returns

output : list of str

Returns a list of names of the modified files (For GEIS files returns the translated names).

drizzlepac.util.**update_input**(*filelist*, *ivmlist=None*, *removed_files=None*)

Removes files flagged to be removed from the input filelist. Removes the corresponding ivm files if present.

drizzlepac.util.**validateUserPars**(*configObj*, *input_dict*)

Compares input parameter names specified by user with those already recognized by the task.

Any parameters provided by the user that does not match a known task parameter will be reported and a ValueError exception will be raised.

drizzlepac.util.**verifyFilePermissions**(*filelist*, *chmod=True*)

Verify that images specified in 'filelist' can be updated.

A message will be printed reporting the names of any images which do not have write-permission, then quit.

drizzlepac.util.**verifyRefimage**(*refimage*)

Verify that the value of refimage specified by the user points to an extension with a proper WCS defined. It starts by making sure an extension gets specified by the user when using a MEF file. The final check comes by looking for a CD matrix in the WCS object itself. If either test fails, it returns a value of False.

drizzlepac.util.**verifyUniqueWcsname**(*fname*, *wcsname*)

Report whether or not the specified WCSNAME already exists in the file

drizzlepac.util.**verifyUpdatewcs**(*fname*)

Verify the existence of WCSNAME in the file. If it is not present, report this to the user and raise an exception. Returns True if WCSNAME was found in all SCI extensions.

10.2 WCS Utilities

These functions read and interpret the WCS information from input images and create the output WCS objects based on STWCS routines.

class drizzlepac.wcs_functions.**IdentityMap**(*input*, *output*)

forward(*pixx*, *pixy*)

class drizzlepac.wcs_functions.**LinearMap**(*xsh=0.0*, *ysh=0.0*, *rot=0.0*, *scale=1.0*)

forward(*pixx*, *pixy*)

class drizzlepac.wcs_functions.**WCSMap**(*input*, *output*, *origin=1*)
Sample class to demonstrate how to define a coordinate transformation

backward(*pixx*, *pixy*)
Transform pixx,pixy positions from the output frame back onto their original positions in the input frame.

checkWCS(*obj*, *name*)

forward(*pixx*, *pixy*)
Transform the input pixx,pixy positions in the input frame to pixel positions in the output frame.

This method gets passed to the drizzle algorithm.

get_pix_ratio()
Return the ratio of plate scales between the input and output WCS. This is used to properly distribute the flux in each pixel in 'tdriz'.

rd2xy(*wcs*, *ra*, *dec*)
Transform input sky positions into pixel positions in the WCS provided.

xy2rd(*wcs*, *pixx*, *pixy*)
Transform input pixel positions into sky positions in the WCS provided.

drizzlepac.wcs_functions.**apply_fitlin**(*data*, *P*, *Q*)

drizzlepac.wcs_functions.**build_hstwcs**(*crval1*, *crval2*, *crpix1*, *crpix2*, *naxis1*, *naxis2*, *pscale*, *orientat*)
Create an HSTWCS object for a default instrument without distortion based on user provided parameter values.

drizzlepac.wcs_functions.**build_pixel_transform**(*chip*, *output_wcs*)

drizzlepac.wcs_functions.**calcNewEdges**(*wcs*, *shape*)
This method will compute sky coordinates for all the pixels around the edge of an image AFTER applying the geometry model.

Parameters
wcs : obj

HSTWCS object for image

shape : tuple

numpy shape tuple for size of image

Returns
border : arr

array which contains the new positions for all pixels around the border of the edges in alpha,dec

drizzlepac.wcs_functions.**computeEdgesCenter**(*edges*)

drizzlepac.wcs_functions.**convertWCS**(*inwcs*, *drizwcs*)
Copy WCSObject WCS into Drizzle compatible array.

drizzlepac.wcs_functions.**createWCSObject**(*output*, *default_wcs*, *imageObjectList*)
Converts a PyWCS WCS object into a WCSObject(baseImageObject) instance.

drizzlepac.wcs_functions.**create_CD**(*orient*, *scale*, *cx=None*, *cy=None*)
Create a (un?)distorted CD matrix from the basic inputs

The 'cx' and 'cy' parameters, if given, provide the X and Y coefficients of the distortion as returned by reading the IDCTAB. Only the first 2 elements are used and should correspond to the 'OC[X/Y]10' and 'OC[X/Y]11' terms in that order as read from the expanded SIP headers.

The units of 'scale' should be 'arcseconds/pixel' of the reference pixel. The value of 'orient' should be the absolute orientation on the sky of the reference pixel.

drizzlepac.wcs_functions.**ddtohms**(*xsky*, *ysky*, *verbose=False*, *precision=6*)
Convert sky position(s) from decimal degrees to HMS format.

drizzlepac.wcs_functions.**fitlin**(*imgarr*, *refarr*)
Compute the least-squares fit between two arrays. A Python translation of 'FITLIN' from 'drutil.f' (Drizzle V2.9).

drizzlepac.wcs_functions.**fitlin_clipped**(*xy*, *uv*, *verbose=False*, *mode='rscale'*, *nclip=3*, *reject=3*)
Perform a clipped fit based on the number of iterations and rejection limit (in sigma) specified by the user. This will more closely replicate the results obtained by 'geomap' using 'maxiter' and 'reject' parameters.

drizzlepac.wcs_functions.**fitlin_rscale**(*xy*, *uv*, *verbose=False*)
Performs a linear, orthogonal fit between matched lists of positions 'xy' (input) and 'uv' (output).

Output: (same as for fit_arrays_general)

drizzlepac.wcs_functions.**get_hstwcs**(*filename*, *hdulist*, *extnum*)
Return the HSTWCS object for a given chip.

drizzlepac.wcs_functions.**get_pix_ratio_from_WCS**(*input*, *output*)
[Functional form of .get_pix_ratio() method of WCSMap]

drizzlepac.wcs_functions.**make_outputwcs**(*imageObjectList*, *output*, *configObj=None*)
Computes the full output WCS based on the set of input imageObjects provided as input, along with the predetermined output name from process_input. The user specified output parameters are then used to modify the default WCS to produce the final desired output frame. The input imageObjectList has the outputValues dictionary updated with the information from the computed output WCS. It then returns this WCS as a WCSObject(imageObject) instance.

drizzlepac.wcs_functions.**mergeWCS**(*default_wcs*, *user_pars*)
Merges the user specified WCS values given as dictionary derived from the input configObj object with the output PyWCS object computed using distortion.output_wcs().

The user_pars dictionary needs to have the following set of keys:

```
user_pars = {'ra':None,'dec':None,'scale':None,'rot':None,
             'outnx':None,'outny':None,'crpix1':None,'crpix2':None}
```

drizzlepac.wcs_functions.**removeAllAltWCS**(*hdulist*, *extlist*)
Removes all alternate WCS solutions from the header

drizzlepac.wcs_functions.**restoreDefaultWCS**(*imageObjectList*, *output_wcs*)
Restore WCS information to default values, and update imageObject accordingly.

drizzlepac.wcs_functions.**updateImageWCS**(*imageObjectList*, *output_wcs*)

drizzlepac.wcs_functions.**updateWCS**(*drizwcs*, *inwcs*)
Copy output WCS array from Drizzle into WCSObject.

drizzlepac.wcs_functions.**update_linCD**(*cdmat, delta_rot=0.0, delta_scale=1.0, cx=[0.0, 1.0], cy=[1.0, 0.0]*)

Modify an existing linear CD matrix with rotation and/or scale changes and return a new CD matrix. If 'cx' and 'cy' are specified, it will return a distorted CD matrix.

Only those terms which are varying need to be specified on input.

drizzlepac.wcs_functions.**wcsfit**(*img_wcs*, *ref_wcs*)

Perform a linear fit between 2 WCS for shift, rotation and scale. Based on the WCSLIN function from 'drutil.f'(Drizzle V2.9) and modified to allow for differences in reference positions assumed by PyDrizzle's distortion model and the coeffs used by 'drizzle'.

Parameters

img : obj

ObsGeometry instance for input image

ref_wcs : obj

Undistorted WCSObject instance for output frame

10.3 Output Image Generation

This module manages the creation of the output image FITS file.

class drizzlepac.outputimage.**OutputImage**(*plist*, *input_pars*, *build=True*, *wcs=None*, *single=False*, *blot=False*)

This class manages the creation of the array objects which will be used by Drizzle. The three arrays, SCI/WHT/CTX, will be setup either as extensions in a single multi-extension FITS file, or as separate FITS files.

The object 'plist' must contain at least the following members:

```
plist['output']   - name of output FITS image (for SCI)
plist['outnx']    - size of X axis for output array
plist['outny']    - size of Y axis for output array
```

If 'single=yes', then 'plist' also needs to contain:

```
plist['outsingle']
plist['outsweight']
plist['outscontext']
```

If 'blot=yes', then 'plist' also needs:

```
plist['data']
plist['blotImage']
plist['blotnx'],plist['blotny']
```

If 'build' is set to 'no', then each extension/array must be in separate FITS objects. This would also require:

```
plist['outdata']    - name of output SCI FITS image
plist['outweight']  - name of output WHT FITS image
plist['outcontext'] - name of output CTX FITS image
```

Optionally, the overall exposure time information can be passed as:

```
plist['texptime'] - total exptime for output
plist['expstart'] - start time of combined exposure
plist['expend']   - end time of combined exposure
```

addDrizKeywords(*hdr*, *versions*)
Add drizzle parameter keywords to header.

find_kwupdate_location(*hdr*, *keyword*)
Find the last keyword in the output header that comes before the new keyword in the original, full input headers. This will rely on the original ordering of keywords from the original input files in order to place the updated keyword in the correct location in case the keyword was removed from the output header prior to calling this method.

set_bunit(*bunit*)
Method used to update the value of the bunit attribute.

set_units(*units*)
Method used to record what units were specified by the user for the output product.

writeFITS(*template*, *sciarr*, *whtarr*, *ctxarr=None*, *versions=None*, *overwrite=True*, *blend=True*, *virtual=False*)
Generate PyFITS objects for each output extension using the file given by 'template' for populating headers.

The arrays will have the size specified by 'shape'.

10.4 MultiDrizzle Reference Table

This module supports the interpretation of the MDRIZTAB for processing as used in the pipeline.

drizzlepac.mdzhandler.**cleanBlank**(*value*)

drizzlepac.mdzhandler.**cleanInt**(*value*)

drizzlepac.mdzhandler.**cleanNaN**(*value*)

drizzlepac.mdzhandler.**findFormat**(*format*)

drizzlepac.mdzhandler.**getMdriztabParameters**(*files*)
Gets entry in MDRIZTAB where task parameters live. This method returns a record array mapping the selected row.

drizzlepac.mdzhandler.**toBoolean**(*flag*)

CHAPTER

ELEVEN

DRIZZLEPAC RELEASE NOTES

The code for this package gets released through a number of methods: namely, the use of the package for pipeline and archive processing of ACS and WFC3 data, SSB's semi-annual public release of the stsci_python package, and a weekly beta release of the development version. The following notes provide some details on what has been revised for each version.

11.1 DrizzlePac Release Notes

The code for this package gets released through a number of methods: namely,

- the use of the package for pipeline and archive processing of ACS and WFC3 data,
- SSB's semi-annual public release of the stsci_python package, and
- a weekly beta release of the development version as part of the IRAFX download.

The following notes provide some details on what has been revised for each version in reverse chronological order (most recent version at the top of the list).

11.1.1 DrizzlePac(astrodrizzle) v1.1.6dev(5-Dec-2012) in IRAFX

available starting: Dec 10, 2012

- tweakreg v1.1.0 source finding algorithm now runs many times faster (no algorithmic changes). No changes have been made yet to speed up the 2d histogram source matching code.
- The 'pixtopix' task was updated to make the 'outimage' parameter optional by using the input image as the default. This required no API changes, but the help files were updated
- Very minor update to guard against MDRIZTAB being specified without any explicit path.
- Update astrodrizzle to correctly report the exposure time, exposure start, and exposure end for the single drizzle products, in addition to insuring the final drizzle values remain correct.
- astrodrizzle also includes initial changes to safeguard the C code from getting improperly cast values from the configObj(TEAL) input.

11.1.2 DrizzlePac(astrodrizzle) v1.1.5dev(23-Oct-2012) in IRAFX

available starting: Oct 29, 2012

- Scaling of sky array for WFC3/IR IVM generation now correct

- template mask files for WFPC2 no longer generated so that WFPC2 data can now be processed using num_cores > 1 (parallel processing)
- interpretation of the 'group' parameter fixed to support a single integer, a comma-separated list of integers or a single 'sci,<n>' value. The values correspond to the FITS extension number of the extensions that should be combined. This fix may also speed up the initialization step as more direct use of pyfits was implemented for the interpretation of the 'group' parameter.

11.1.3 DrizzlePac(astrodrizzle) v1.1.1(31-Aug-2012) in HST Archive

available starting: Sept 26, 2012

The HST Archive and operational calibration pipeline started using this version on Sept 26, 2012.

11.1.4 DrizzlePac(astrodrizzle) v1.1.4dev(20-Sep-2012) in IRAFX

available starting: Sept 24, 2012

- Bug fixed to allow use of final_wht_type=IVM for processing WFPC2 data
- Revised Initialization processing to speed it up by using more up-to-date, direct pyfits calls.

11.1.5 DrizzlePac(astrodrizzle) v1.1.3(7-Sep-2012) in IRAFX

available starting: Sept 17, 2012

- Fixed the logic so that crclean images always get created regardless of the value of the 'clean' parameter.

11.1.6 DrizzlePac(astrodrizzle) v1.1.2(5-Sep-2012) in IRAFX

available starting: Sept 10, 2012

- Remove the restriction of only being able to process images which have WCSNAME keyword as imposed by r15631. The removal of this restriction will now allow for processing of non-updated input files with updatewcs=False for cases where no distortion model exists for the data (as required by CADC).
- Added log statements reporting what sky value was actually used in the drizzle and blot steps

11.1.7 DrizzlePac(astrodrizzle) v1.1.1(30-Aug-2012) in IRAFX

available starting: Sept 3, 2012

- Major revision to astrodrizzle allowing the option to process without writing out any intermediate products to disk. The intermediate products remain in memory requiring significantly more memory than usual. This improves the overall processing time by eliminating as much disk activity as possible as long as the OS does not start disk swapping due to lack of RAM.
- revised to turn off 'updatewcs' when coeffs=False(no) so that exposures with filter combinations not found in the IDCTAB will not cause an error

11.1.8 DrizzlePac(astrodrizzle) v1.0.7(21-Aug-2012) in IRAFX

available starting: Aug 27, 2012

- Fixes problems with missing single_sci images.
- Static mask step revised to skip updates to static mask if all pixel data falls within a single histogram bin. This avoids problems with masking out entire images, which happens if low S/N SBC data is processed with static_mask=yes.

11.1.9 DrizzlePac(astrodrizzle) v1.0.6(14-Aug-2012) in IRAFX

available starting: Aug 20, 2012

Use of IVM for final_wht now correct, as previous code used wrong inputs when IVM weighting was automatically generated by astrodrizzle.

11.1.10 DrizzlePac(astrodrizzle) v1.0.5(8-Aug-2012) in IRAFX

available starting: Aug 13, 2012

- Completely removed the use of the TIME arrays for weighting IR drizzle products so that the photometry for saturated sources in drizzled products now comes out correct.
- Corrected a problem with astrodrizzle which affected processing of WFPC2 data where CRPIX2 was not found when creating the output single sci image.

11.1.11 stsci_python v2.13 [Includes astrodrizzle v1.0.2(13-July-2012)]

available starting: Aug 3, 2012

The complete version of stsci_python can be downloaded from our download page

- stsci_python v2.13 Release Notes
- Old stsci_python release notes

11.1.12 DrizzlePac(astrodrizzle) v1.0.1(20-June-2012)

Used in archive/pipeline starting: July 10, 2012

Pipeline and archive started processing ACS data with this version.

11.1.13 DrizzlePac(astrodrizzle) v1.0.0(25-May-2012)

Used in archive/pipeline starting: June 6, 2012

Pipeline and archive first started using astrodrizzle by processing WFC3 images.

CHAPTER

TWELVE

IMAGE REGISTRATION TASKS

Documentation for the replacement task for IRAF's *tweakshifts*, currently named *tweakreg*, has been added to this package. These new modules describe how to run the new TEAL-enabled task, as well as use the classes in the task to generate catalogs interactively for any chip and work with that catalog. The current implementation of this code relies on a very basic source finding algorithm loosely patterned after the DAOFIND algorithm and does not provide all the same features or outputs found in DAOFIND. The fitting algorithm also reproduces the fitting performed by IRAF's *geomap* in a limited fashion; primarily, without iterations of outliers and only to perform fits equivalent to *geomap*'s 'shift' and 'rscale' solutions. These algorithms will be upgraded as soon as replacements are available.

12.1 TWEAKREG: Alignment of Images

Combining images using astrodrizzle requires that the WCS information in the headers of each input image align to within sub-pixel accuracy. The 'tweakreg' task allows the user to align sets of images to each other and/or to and external astrometric reference frame or image. ImageReg - A replacement for IRAF-based 'tweakshifts'

drizzlepac.tweakreg.**TweakReg**(*files=None*, *editpars=False*, *configobj=None*, *imagefindcfg=None*, ***input_dict*)

tweakreg Version 1.2.1 updated on 25-Jan-2013

Tweakreg provides an automated interface for computing residual shifts between input exposures being combined using AstroDrizzle. The offsets computed by Tweakreg correspond to pointing differences after applying the WCS information from the input image's headers. Such errors would, for example, be due to errors in guide-star positions when combining observations from different observing visits or from slight offsets introduced upon re-acquiring the guide stars in a slightly different position.

Parameters

input : str or list of str

Input files (passed in from *files* parameter) This paramater can be provided in any of several forms:

- filename of a single image
- filename of an association (ASN)table
- wild-card specification for files in directory (using *, ? etc)
- comma-separated list of filenames
- '@file' filelist containing list of desired input filenames (and optional inverse variance map filenames)

The '@file' filelist needs to be provided as an ASCII text file containing a list of filenames for all input images with one filename on each line of the file. If inverse variance maps have also been created by the user and are to be used (by specifying 'IVM' to the

parameter 'final_wht_type' described further below), then these are simply provided as a second column in the filelist, with each IVM filename listed on the same line as a second entry, after its corresponding exposure filename. [Default value: '*flt.fits']

refimage : str

Filename of reference image. Sources derived from this image will be used as the reference for matching with sources from all input images. [Default value: '']

exclusions: string :

This parameter allows the user to specify a set of files which contain regions in the image to ignore when finding sources. This file MUST have 1 line for each input image with the name of the input file in the first column. Subsequent columns would be used to specify an exclusion file for each chip in 'SCI,<n>' index order. If a chip does not require an exclusion file, the string 'None' or 'INDEF' can be used as a placeholder for that chip. Each chip's exclusion file should conform to the 'region' file format generated by DS9, and currently, only 'circle()' regions are interpreted and used and only with units for the positions of the exclusion center of 'fk[4|5]' and 'image'. Support for additional region file options will be added at a later date. [Default value: '']

updatewcs : bool

Update WCS keywords of input images? [Default value: No]

writecat : bool

Specify whether or not to write out the source catalogs generated for each input image by the built-in source extraction algorithm. [Default value: Yes]

clean : bool

Specify whether or not to remove the temporary files created by 'tweakreg', including any catalog files generated for the shift determination. [Default value = No]

verbose : bool

Specify whether or not to print extra messages during processing. [Default value = No]

runfile : string

Specify the filename of the processing log. [Default value = 'tweakreg.log']

UPDATE HEADER :

updatehdr : bool

Specify whether or not to update the headers of each input image directly with the shifts that were determined. This will allow the input images to be combined by AstroDrizzle without having to provide the shiftfile as well. [Default value = No]

wcsname : str

Name of updated WCS. [Default value = 'TWEAK']

HEADERLET CREATION :

headerlet: bool :

Specify whether or not to generate a headerlet from the images at the end of the task? If turned on, this will create a headerlet from the images regardless of the value of the 'updatehdr' parameter. [Default value = No]

attach: bool :

If creating a headerlet, choose whether or not to attach the new headerlet to the input image as a new extension. [Default value = Yes]

hdrfile: string :

Filename to use for writing out headerlet to a separate file. If the name does not contain '.fits', it will create a filename from the rootname of the input image, the value of this string, and it will end in '_hlet.fits'. For example, if only 'hdrlet1' is given, the full filename created will be 'j99da1f2q_hdrlet1_hlet.fits' when creating a headerlet for image 'j99da1f2q_flt.fits'. [Default value = '']

clobber: bool :

If a headerlet with 'hdrfile' already exists on disk, specify whether or not to overwrite that previous file. [Default value = No]

hdrname: string :

Unique name to give to headerlet solution. This name will be used to identify this specific WCS alignment solution contained in the headerlet. [Default value = '']

author: string, optional :

Name of the creator of the headerlet. [Default value = '']

descrip: string, optional :

Short (1-line) description to be included in headerlet as 'DESCRIP' keyword. This can be used to provide a quick look description of the WCS alignment contained in the headerlet. [Default value = '']

catalog: string, optional :

Name of reference catalog used as the basis for the image alignment. [Default value = '']

history: string, optional :

Filename of a file containing detailed information regarding the history of the WCS solution contained in the headerlet. This can include information on the catalog used for the alignment, or notes on processing that went into finalizing the WCS alignment stored in this headerlet. This information will be reformatted as 70-character wide FITS HISTORY keyword section. [Default value = '']

OPTIONAL SHIFTFILE OUTPUT :

shiftfile : bool

Create output shiftfile? [Default value: No]

outshifts : str

The name for the output shift file created by 'tweakreg'. This shiftfile will be formatted for use as direct input to AstroDrizzle. [Default value: 'shifts.txt']

outwcs : str

Filename to be given to the OUTPUT reference WCS file created by 'tweakreg'. This reference WCS defines the WCS from which the shifts get measured, and will be used by AstroDrizzle to interpret those shifts. This reference WCS file will be a FITS file that only contains the WCS keywords in a Primary header with no image data itself. The values will be derived from the FIRST input image specified. [Default value: 'shifts_wcs.fits']

COORDINATE FILE DESCRIPTION :

catfile : str

Name of file that contains a list of input images and associated catalog files generated by the user. Each line of this file will contain the name of an input image in the first column. The remaining columns will provide the names of the source catalogs for each chip in order of the science extension numbers ((SCI,1), (SCI,2), ...). [Default value: ‘’]

A sample catfile, with one line per image would look like:

```
image1_flt.fts  cat1_sci1.coo  cat1_sci2.coo
image2_flt.fts  cat2_sci1.coo  cat2_sci2.coo
```

xcol : int

Column number of X position from the user-generated catalog files specified in the catfile. [Default value = 1]

ycol : int

Column number of Y position from the user-generated catalog files specified in the catfile. [Default value = 2]

fluxcol : int

Column number for the flux values from the user-generated catalog files specified in the catfile. These values will only be used if a flux limit has been specified by the user using the ‘fluxmax’ or ‘fluxmin’ parameters. [Default value = None]

fluxmax : float

Limiting flux value for selecting valid objects in the input image’s catalog. If specified, this flux will serve as the upper limit of a range for selecting objects to be used in matching with objects identified in the reference image. If the value is set to None, all objects with fluxes brighter than the minimum specified in ‘fluxmin’ will be used. If both values are set to None, all objects will be used. [Default value = None]

fluxmin : float

Limiting flux value for selecting valid objects in the input image’s catalog. If specified, this flux value will serve as the lower limit of a range for selecting objects to be used in matching with objects identified in the reference image. If the value is set to None, all objects fainter than the limit specified by ‘fluxmax’ will be used. If both values are set to None, all objects will be used. [Default value = None]

fluxunits : str {‘counts’, ‘cps’, ‘mag’}

This allows the task to correctly interpret the flux limits specified by ‘fluxmax’ and ‘fluxmin’ when sorting the object list for trimming of fainter objects. [Default value = ‘counts’]

xyunits : str {‘pixels’, ‘degrees’}

Specifies whether the positions in this catalog are already sky pixel positions, or whether they need to be transformed to the sky. [Default value = ‘pixels’]

nbright : int

The number of brightest objects to keep after sorting the full object list. If nbright is set equal to None, all objects will be used. [Default value = None]

REFERENCE CATALOG DESCRIPTION :

refcat : str

Name of the external reference catalog file to be used in place of the catalog extracted from one of the input images. [Default value = ‘’]

refxcol : int

Column number of RA in the external catalog file specified by the refcat. [Default value = 1]

refycol : int

Column number of Dec in the external catalog file specified by the refcat. [Default value = 2]

refxyunits : str {‘pixels’,’degrees’}

Units of sky positions. [Default value = ‘degrees’]

rfluxcol : int

Column number of flux/magnitude values in the external catalog file specified by the refcat. [Default value = None]

rfluxmax : float

Limiting flux value used to select valid objects in the external catalog. If specified, the flux value will serve as the upper limit of a range for selecting objects to be used in matching with objects identified in the reference image. If the value is set to None, all objects with fluxes brighter than the minimum specified in ‘rfluxmin’ will be used. If both values are set to None, all objects will be used. [Default value = None]

rfluxmin : float

Limiting flux value used to select valid objects in the external catalog. If specified, the flux will serve as the lower limit of a range for selecting objects to be used in matching with objects identified in the reference image. If the value is set to None, all objects fainter than the limit specified by ‘rfluxmax’ will be used. If both values are set to None, all objects will be used. [Default value = None]

rfluxunits : {‘counts’, ‘cps’, ‘mag’}

This allows the task to correctly interpret the flux limits specified by ‘rfluxmax’ and ‘rfluxmin’ when sorting the object list for trimming of fainter objects. [Default value = ‘mag’]

refnbright : int

Number of brightest objects to keep after sorting the full object list. If refnbright is set to None, all objects will be used. Used in conjunction with refcat. [Default value = None]

OBJECT MATCHING PARAMETERS :

minobj : int

Minimum number of identified objects from each input image to use in matching objects from other images. [Default value = 15]

searchrad : float

The search radius for a match. [Default value = 1.0]

searchunits : str

Units for search radius. [Default value = 'arcseconds']

use2dhist : bool

Use 2d histogram to find initial offset? [Default value = Yes]

see2dplot : bool

See 2d histogram for initial offset? [Default value = Yes]

tolerance : float

The matching tolerance in pixels after applying an initial solution derived from the 'triangles' algorithm. This parameter gets passed directly to *xyxymatch* for use in matching the object lists from each image with the reference image's object list. [Default value = 1.0]

separation : float

The minimum separation for objects in the input and reference coordinate lists. Objects closer together than 'separation' pixels are removed from the input and reference coordinate lists prior to matching. This parameter gets passed directly to *xyxymatch* for use in matching the object lists from each image with the reference image's object list. [Default value = 0.0]

xoffset : float

Initial estimate for the offset in X between the images and the reference frame. This offset will be used for all input images provided. If the parameter value is set to None, no offset will be assumed in matching sources in 'xyxymatch'. [Default value = 0.0]

yoffset : float

Initial estimate for the offset in Y between the images and the reference frame. This offset will be used for all input images provided.If the parameter value is set to None, no offset will be assumed in matching sources in 'xyxymatch'. [Default value = 0.0]

CATALOG FITTING PARAMETERS :

fitgeometry : str {'shift', 'rscale'}

The fitting geometry to be used in fitting the matched object lists. This parameter is used in fitting the offsets, rotations and/or scale changes from the matched object lists. [Default value = 'rscale']

residplot : str {'No plot', 'vector', 'residuals', 'both'}

Plot residuals from fit? [Default value = 'both'] If 'both' is selected, the 'vector' and 'residuals' plots will be displayed in separate plotting windows at the same time.

nclip : int

Number of clipping iterations in fit. [Default value = 3]

sigma : float

Clipping limit in sigma units. [Default value = 3.0]

See also:

`astrodrizzle`

Notes

Tweakreg supports the use of calibrated, distorted images (such as FLT images for ACS and WFC3, or _c0m.fits images for WFPC2) as input images. All coordinates for sources derived from these images (either by this task or as provided by the user directly) will be corrected for distortion using the distortion model information specified in each image's header. This eliminates the need to run 'astrodrizzle' on the input images prior to running tweakreg.

Note: All calibrated input images must have been updated using 'updatewcs' from the STWCS package, or as run by 'astrodrizzle', to include the full distortion model in the header.

This task will use catalogs, and catalog-matching, based on the 'xyxymatch' algorithm to determine the offset between the input images. The primary mode of operation will be to extract a catalog of source positions from each input image using either a 'DAOFIND-like' algorithm or SExtractor (if the user has SExtractor installed). Alternatively, the user can provide their catalogs of source positions derived from **each input chip**.

The reference frame will be defined either by:

- the first input image (and associated catalog),
- a catalog derived from a reference image specified by the user, or
- **a catalog of undistorted sky positions (RA/Dec) and fluxes** provided by the user.

For a given observation, the distortion model is applied to all distorted input positions, and the sources from each chip are then combined into a single catalog of undistorted positions.

The undistorted positions for each observation then get passed to 'xyxymatch' for matching to objects from the reference catalog.

The source lists from each image will generally include cosmic-rays as detected sources, which can at times significantly confuse object identification between images. Observations that include long exposures often have more cosmic-ray events than source objects. As such, isolating the cosmic-ray events in those cases would significantly improve the efficiency of common source identification between images. One such method for trimming potential false detections from each source list would be to set a flux limit to exclude detections below that limit. As the fluxes reported in the default source object lists are provided as magnitude values, setting the *fluxmax* or *fluxmin* parameter value to a magnitude- based limit, and then setting the *ascend* parameter to *True*, will allow for the creations of catalogs trimmed of all sources fainter than the provided limit. The trimmed source list can then be used in matching sources between images and in establishing the final fitting for the shifts.

A fit can then be performed on the matched set of positions between the input and the reference to produce the 'shiftfile'. If the user is confident that the solution will be correct, the header of each input image can be updated directly with the fit derived for that image. Otherwise, the 'shiftfile' can be passed to AstroDrizzle for aligning the images.

Format of Exclusion Catalog

The format for the exclusions catalog requires 1 line in the file for every input image, regardless of whether or not that image has any defined exclusion regions. A sample file would look like:

```
j99da1emq_flt.fits
j99da1f2q_flt.fits test_exclusion.reg
```

This file specifies no exclusion files for the first image, and only an regions file for SCI,1 of the second image. Should an exclusion regions file only be needed for the second chip in the second image, the file would need to look like:

```
j99da1emq_flt.fits
j99da1f2q_flt.fits None test_sci2_exclusion.reg
```

The value 'None' could also be replaced by 'INDEF' if desired, but either string needs to be present to signify no regions file for that chip while the code continues parsing the line to find a file for the second chip.

Format of Region Files

The format of the exclusions catalogs referenced in the 'exclusions' file defaults to the format written out by DS9 using the 'DS9/Funtools' region file format. A sample file with circle() regions will look like:

```
# Region file format: DS9 version 4.1
# Filename: j99da1f2q_flt.fits[SCI]
global color=green dashlist=8 3 width=1 font="helvetica 10 normal roman" select=1 highlite=1 das
image
circle(3170,198,20)
circle(3269,428,20)
circle(3241.1146,219.78132,20.000014)
```

This task can also work with regions files which have a much simpler format for each region with 2 values for the source center and a third value for the radius of the region. The units of all the values can either pixels or sky coordinates (either sexagesimal RA/Dec or decimal degrees RA/Dec plus radius in arcseconds). The values on each line can be separated either by spaces or commas.

A sample file with free formatted regions would look like:

```
image
3170,198,20
3269 428 20
3241.1146,219.78132,20.000014
```

The first (non-comment) line in the file will specify what units were used for specifying the regions. Supported options are:

- 'image'
- 'physical'
- 'pixel'
- 'fk5'
- 'fk4'
- 'sky'

The terms 'image', 'physical', and 'pixel' all refer to values in *pixels*, while the remaining naturally refer to values specified in *sky coordinates*. A **default units of 'pixels'** will be assumed should the units line not be found at the beginning of the regions file.

The format of the regions can actually be different from one line to the next, in case multiple regions files written out in different formats need to be merged. So, regions from the second example could be included in the first examples file if needed.

Examples

The tweakreg task can be run from either the TEAL GUI or from the command-line using PyRAF or Python. These examples illustrate the various syntax options available.

Example 1: Align a set of calibrated (_flt.fits) images using IMAGEFIND, a built-in source finding algorithm based on DAOPHOT. Auto-detect the sky sigma value and select sources > 200 sigma. (Auto-sigma is computed from the first input exposure as: 1.5 * imstat(image,nclip=3,fields='stddev').) Set the convolution kernel width to ~2x the value of the PSF FWHM. Save the residual offsets (dx, dy, rot, scale, xfit_rms, yfit_rms) to a text file.

1. Run the task from PyRAF using the TEAL GUI:

```
--> import drizzlepac
--> epar tweakreg
```

2. Run the task from PyRAF using the command line.

```
--> import drizzlepac
--> from drizzlepac import tweakreg
--> tweakreg.TweakReg('*flt.fits', updatehdr=False, fwhmpsf=3.5, threshold=200, \
                      shiftfile=True, outshifts='shift.txt')
```

Or, run the same task from the PyRAF command line, but specify all parameters in a config file named "myparam.cfg":

```
--> tweakreg.TweakReg('*flt.fits', configobj='myparam.cfg')
```

3. Run the task directly from Python:

```
>>> from drizzlepac import tweakreg
>>> tweakreg.TweakReg('*flt.fits', updatehdr=False, fwhmpsf=3.5, threshold=200, \
                      shiftfile=True, outshifts='shift.txt')
```

Alternately, edit the imagefind parameters in a TEAL GUI window prior to running the task:

```
>>> tweakreg.edit_imagefindpars()
```

4. Help can be accessed via the "Help" pulldown menu in the TEAL GUI. It can also be accessed from the PyRAF command-line and saved to a text file:

```
--> from drizzlepac import tweakreg
--> tweakreg.help()
```

or:

```
--> tweakreg.help(file='help.txt')
--> page help.txt
```

drizzlepac.tweakreg.**help**(*file=None*)

Print out syntax help for running tweakreg

Parameters

file : str (Default = None)

If given, write out help to the filename specified by this parameter Any previously existing file with this name will be deleted before writing out the help.

12.2 Imagefindpars: Source finding parameters

This interface provides a mechanism for setting the parameters used by the built-in source finding algorithm based on the published algorithms used by DAOFIND. imagefindpars Version 1.2.1 updated on 25-Jan-2013

These parameters control the operation of the algorithm that extracts sources from the image as called by *TWEAKREG: Alignment of Images*. The algorithm implemented follows the concepts defined by DAOFIND (without actually using any DAOFIND code).

Input

The algorithm simply requires the image to be read in as a numpy array, along with the user specified parameters given here.

Output
The primary output, as used by *TWEAKREG: Alignment of Images*, is a list of numpy arrays representing the X and Y positions of detected sources along with flux for each source.

12.2.1 Parameters

computesig
[bool] This parameter controls whether or not to automatically compute a sigma value to be used for object identification. If set to True, then the value computed will override any user input for the parameter 'skysigma'. The automatic sigma value gets computed from each input exposure as:

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

This single value will then be used for object identification for all input exposures. [Default: Yes]

skysigma
[float] The standard deviation of the sky pixels. This value will only be used if computesig is False. [Default: 0.0]

conv_width
[float] The convolution kernel width in pixels. Recommended values (~2x the PSF FWHM): ACS/WFC & WFC3/UVIS ~3.5 pix and WFC3/IR ~2.5 pix. [Default: 3.5]

peakmin
[float] This parameter allows the user to select only those sources whose peak value (in the units of the input image) is greater than this value. [Default: None]

peakmax
[float] This parameter allows the user to select only those sources whose peak value (in the units of the input image) is less than this value. [Default: None]

threshold
[float] The object detection threshold above the local background in units of sigma. [Default: 4.0]

nsigma
[float] The semi-major axis of the Gaussian convolution kernel used to compute the density enhancement and mean density images in Gaussian sigma. [Default: 1.5]

fluxmin
[float] This parameter allows the user to select only those sources whose total flux (in the units of the input image) is greater than this value. [Default: None]

fluxmax
[float] This parameter allows the user to select only those sources whose total flux (in the units of the input image) is less than this value. [Default: None]

`drizzlepac.imagefindpars.`**`getHelpAsString`**(*docstring=False*)
Teal help function to describe the parameters being set Descriptions of parameters come from .help file

`drizzlepac.imagefindpars.`**`help`**()

12.3 Image Class

class `drizzlepac.imgclasses.`**`Image`**(*filename*, *input_catalogs=None*, *exclusions=None*, ***kwargs*)
Bases: `object`

Primary class to keep track of all WCS and catalog information for a single input image. This class also performs all matching and fitting.

Parameters

filename : str

Filename for image.

input_catalogs : list of str or None

Filename of catalog files for each chip, if specified by user.

kwargs : dict

Parameters necessary for processing derived from input configObj object.

buildDefaultRefWCS()
Generate a default reference WCS for this image.

buildSkyCatalog()
Convert sky catalog for all chips into a single catalog for the entire field-of-view of this image.

clean()
Remove intermediate files created.

close()
Close any open file handles and flush updates to disk

compute_fit_rms()

get_shiftfile_row()
Return the information for a shiftfile for this image to provide compatability with the IRAF-based MultiDrizzle.

get_wcs()
Helper method to return a list of all the input WCS objects associated with this image.

get_xy_catnames()
Return a string with the names of input_xy catalog names

match(*refimage*, ***kwargs*)
Uses xyxymatch to cross-match sources between this catalog and a reference catalog (refCatalog).

openFile()
Open file and set up filehandle for image file

performFit(***kwargs*)
Perform a fit between the matched sources.

Parameters

kwargs : dict

Parameter necessary to perform the fit; namely, *fitgeometry*.

Notes

This task still needs to implement (eventually) interactive iteration of
the fit to remove outliers.

sortSkyCatalog()
Sort and clip the source catalog based on the flux range specified by the user. It keeps a copy of the original full list in order to support iteration.

transformToRef(*ref_wcs*, *force=False*)
Transform sky coords from ALL chips into X,Y coords in reference WCS.

updateHeader(*wcsname=None*)
Update header of image with shifts computed by *perform_fit()*.

writeHeaderlet(***kwargs*)
Write and/or attach a headerlet based on update to PRIMARY WCS

write_fit_catalog()
Write out the catalog of all sources and resids used in the final fit.

write_outxy(*filename*)
Write out the output(transformed) XY catalog for this image to a file.

write_skycatalog(*filename*)
Write out the all_radec catalog for this image to a file.

class drizzlepac.imgclasses.**RefImage**(*wcs_list*, *catalog*, *xycatalog=None*, ***kwargs*)
Bases: object

This class provides all the information needed by to define a reference tangent plane and list of source positions on the sky.

clean()
Remove intermediate files created

close()

get_shiftfile_row()
Return the information for a shiftfile for this image to provide compatability with the IRAF-based MultiDrizzle.

transformToRef()
Transform reference catalog sky positions (self.all_radec) to reference tangent plane (self.wcs) to create output X,Y positions.

write_skycatalog(*filename*)
Write out the all_radec catalog for this image to a file.

12.4 Classes to manage Catalogs and WCS's

This module provides the classes used to generate and manage source catalogs for each input chip. Those positions can be transformed to undistorted sky positions, written out to files, or plotted using various methods defined for these classes.

drizzlepac.catalogs.**generateCatalog**(*wcs*, *mode='automatic'*, *catalog=None*, ***kwargs*)
Function which determines what type of catalog object needs to be instantiated based on what type of source selection algorithm the user specified.

Parameters
wcs : obj

WCS object generated by STWCS or PyWCS

catalog : str or ndarray

Filename of existing catalog or ndarray of image for generation of source catalog

kwargs : dict

Parameters needed to interpret source catalog from input catalog with *findmode* being required.

Returns

catalog : obj

A Catalog-based class instance for keeping track of WCS and associated source catalog

class drizzlepac.catalogs.**ImageCatalog**(*wcs*, *catalog_source*, ***kwargs*)

Bases: drizzlepac.catalogs.Catalog

Class which generates a source catalog from an image using Python-based, daofind-like algorithms

Required input *kwargs* parameters:

```
computesig, skysigma, threshold, peakmin, peakmax,
hmin, conv_width, [roundlim, sharplim]
```

generateXY()

Generate source catalog from input image using DAOFIND-style algorithm

class drizzlepac.catalogs.**UserCatalog**(*wcs*, *catalog_source*, ***kwargs*)

Bases: drizzlepac.catalogs.Catalog

Class to manage user-supplied catalogs as inputs.

Required input *kwargs* parameters:

```
xyunits, xcol, ycol[, fluxcol, [idcol]]
```

This class requires the input of a WCS and a source for the catalog, along with any arguments necessary for interpreting the catalog.

Parameters

wcs : obj

Input WCS object generated using STWCS or HSTWCS

catalog_source : str or ndarray

Catalog generated from this image(ndarray) or read from this file(str)

kwargs : dict

Parameters for interpreting the catalog file or for performing the source extraction from the image. These will be set differently depending on the type of catalog being instantiated.

generateXY()

Method to interpret input catalog file as columns of positions and fluxes.

plotXYCatalog(***kwargs*)

Plots the source catalog positions using matplotlib's *pyplot.plot()*

Plotting *kwargs* that can also be passed include any keywords understood by matplotlib's *pyplot.plot()* function such as:

```
vmin, vmax, cmap, marker
```

set_colnames()

COLNAMES = ['xcol', 'ycol', 'fluxcol']

IN_UNITS = None

class drizzlepac.catalogs.**RefCatalog**(*wcs*, *catalog_source*, ***kwargs*)
Bases: drizzlepac.catalogs.UserCatalog

Class which manages a reference catalog.

Notes

A *reference catalog* is defined as a catalog of undistorted source positions given in RA/Dec which would be used as the master list for subsequent matching and fitting.

This class requires the input of a WCS and a source for the catalog, along with any arguments necessary for interpreting the catalog.

Parameters

wcs : obj

Input WCS object generated using STWCS or HSTWCS

catalog_source : str or ndarray

Catalog generated from this image(ndarray) or read from this file(str)

kwargs : dict

Parameters for interpreting the catalog file or for performing the source extraction from the image. These will be set differently depending on the type of catalog being instantiated.

buildXY(*catalogs*)

generateRaDec()

generateXY()

COLNAMES = ['refxcol', 'refycol', 'rfluxcol']

IN_UNITS = 'degrees'

class drizzlepac.catalogs.**Catalog**(*wcs*, *catalog_source*, ***kwargs*)
Bases: object

Base class for keeping track of a source catalog for an input WCS

Warning: This class should never be instantiated by itself, as necessary methods are not defined yet.

This class requires the input of a WCS and a source for the catalog, along with any arguments necessary for interpreting the catalog.

Parameters

wcs : obj

Input WCS object generated using STWCS or HSTWCS

catalog_source : str or ndarray

Catalog generated from this image(ndarray) or read from this file(str)

kwargs : dict

Parameters for interpreting the catalog file or for performing the source extraction from the image. These will be set differently depending on the type of catalog being instantiated.

apply_exclusions(*exclusions*)
Trim sky catalog to remove any sources within regions specified by exclusions file

buildCatalogs(*exclusions=None*)
Primary interface to build catalogs based on user inputs.

generateRaDec()
Convert XY positions into sky coordinates using STWCS methods

generateXY()
Method to generate source catalog in XY positions Implemented by each subclass

plotXYCatalog(***kwargs*)
Method which displays the original image and overlays the positions of the detected sources from this image's catalog.

Plotting *kwargs* that can be provided are:

```
vmin, vmax, cmap, marker
```

Default colormap is *summer*.

set_colnames()
Method to define how to interpret a catalog file Only needed when provided a source catalog as input

writeXYCatalog(*filename*)
Write out the X,Y catalog to a file

12.5 Functions to Manage WCS Table Extension

These functions provide the basic support for initializing, creating and updating the WCS table extension which serves as the archive of updates made to the WCS information in the image headers.

stwcs.wcsutil.wcscorr.**archive_wcs_file**(*image*, *wcs_id=None*)
Update WCSCORR table with rows for each SCI extension to record the newly updated WCS keyword values.

stwcs.wcsutil.wcscorr.**create_wcscorr**(*descrip=False*, *numrows=1*, *padding=0*)
Return the basic definitions for a WCSCORR table. The dtype definitions for the string columns are set to the maximum allowed so that all new elements will have the same max size which will be automatically truncated to this limit upon updating (if needed).

The table is initialized with rows corresponding to the OPUS solution for all the 'SCI' extensions.

stwcs.wcsutil.wcscorr.**delete_wcscorr_row**(*wcstab*, *selections=None*, *rows=None*)
Sets all values in a specified row or set of rows to default values

This function will essentially erase the specified row from the table without actually removing the row from the table. This avoids the problems with trying to resize the number of rows in the table while preserving the ability to update the table with new rows again without resizing the table.

Parameters
wcstab: object :

PyFITS binTable object for WCSCORR table

selections: dict :

Dictionary of wcscorr column names and values to be used to select the row or set of rows to erase

rows: int, list :

If specified, will specify what rows from the table to erase regardless of the value of 'selections'

stwcs.wcsutil.wcscorr.**find_wcscorr_row**(*wcstab*, *selections*)

Return an array of indices from the table (NOT HDU) 'wcstab' that matches the selections specified by the user.

The row selection criteria must be specified as a dictionary with column name as key and value(s) representing the valid desired row values. For example, {'wcs_id':'OPUS','extver':2}.

stwcs.wcsutil.wcscorr.**init_wcscorr**(*input*, *force=False*)

This function will initialize the WCSCORR table if it is not already present, and look for WCS keywords with a prefix of 'O' as the original OPUS generated WCS as the initial row for the table or use the current WCS keywords as initial row if no 'O' prefix keywords are found.

This function will NOT overwrite any rows already present.

This function works on all SCI extensions at one time.

stwcs.wcsutil.wcscorr.**restore_file_from_wcscorr**(*image*, *id='OPUS'*, *wcskey=''*)

Copies the values of the WCS from the WCSCORR based on ID specified by user. The default will be to restore the original OPUS-derived values to the Primary WCS. If wcskey is specified, the WCS with that key will be updated instead.

stwcs.wcsutil.wcscorr.**update_wcscorr**(*dest*, *source=None*, *extname='SCI'*, *wcs_id=None*, *active=True*)

Update WCSCORR table with a new row or rows for this extension header. It copies the current set of WCS keywords as a new row of the table based on keyed WCSs as per Paper I Multiple WCS standard).

Parameters

dest : HDUList

The HDU list whose WCSCORR table should be appended to (the WCSCORR HDU must already exist)

source : HDUList, optional

The HDU list containing the extension from which to extract the WCS keywords to add to the WCSCORR table. If None, the dest is also used as the source.

extname : str, optional

The extension name from which to take new WCS keywords. If there are multiple extensions with that name, rows are added for each extension version.

wcs_id : str, optional

The name of the WCS to add, as in the WCSNAMEa keyword. If unspecified, all the WCSs in the specified extensions are added.

active: bool, optional :

When True, indicates that the update should reflect an update of the active WCS information, not just appending the WCS to the file as a headerlet

stwcs.wcsutil.wcscorr.**update_wcscorr_column**(*wcstab*, *column*, *values*, *selections=None*, *rows=None*)

Update the values in 'column' with 'values' for selected rows

Parameters

wcstab: object :

PyFITS binTable object for WCSCORR table

column: string :

Name of table column with values that need to be updated

values: string, int, or list :

Value or set of values to copy into the selected rows for the column

selections: dict :

Dictionary of wcscorr column names and values to be used to select the row or set of rows to erase

rows: int, list :

If specified, will specify what rows from the table to erase regardless of the value of 'selections'

12.6 Functions to Manage Legacy OPUS WCS Keywords in the WCS Table

The previously released versions of `makewcs` provided with MultiDrizzle *archives* the original OPUS generated WCS keywords using header keywords which have a prefix of "O", such as "OCRPIX1". In order to avoid overwriting or ignoring these original values, these functions can be used to convert the prefixed OPUS WCS keywords into WCS table entries compatible with the new code.

Strictly to provide complete support for these OPUS keywords, the code will also create, if the user desires, prefix "O" WCS keywords from the alternate WCS FITS conventions OPUS keywords. This would allow images processed using the new code only can then be used with older versions of MultiDrizzle, if the user needs such compatibility.

`stwcs.wcsutil.convertwcs.`**`archive_prefix_OPUS_WCS`**(*fobj*, *extname='SCI'*)

Identifies WCS keywords which were generated by OPUS and archived using a prefix of 'O' for all 'SCI' extensions in the file

Parameters

fobj: string or pyfits.HDUList :

Filename or pyfits object of a file

`stwcs.wcsutil.convertwcs.`**`create_prefix_OPUS_WCS`**(*fobj*, *extname='SCI'*)

Creates alternate WCS with a prefix of 'O' for OPUS generated WCS values to work with old MultiDrizzle.

Parameters

fobj: string or pyfits.HDUList :

Filename or pyfits object of a file

Raises

IOError: :

if input PyFITS object was not opened in 'update' mode

12.7 TWEAKUTILS: Utility Functions for Tweakreg

The functions in this module support the various aspects of tweakreg, including finding the objects in the images and plotting the residuals.

drizzlepac.tweakutils.**build_pos_grid**(*start*, *end*, *nstep*, *mesh=False*)
Return a grid of positions starting at X,Y given by 'start', and ending at X,Y given by 'end'. The grid will be completely filled in X and Y by every 'step' interval.

drizzlepac.tweakutils.**build_xy_zeropoint**(*imgxy*, *refxy*, *searchrad=3.0*, *histplot=False*, *figure_id=1*)
Create a matrix which contains the delta between each XY position and each UV position.

drizzlepac.tweakutils.**createWcsHDU**(*wcs*)
Generate a WCS header object that can be used to populate a reference WCS HDU.

For most applications, stwcs.wcsutil.HSTWCS.wcs2header() will work just as well.

drizzlepac.tweakutils.**find_xy_peak**(*img*, *center=None*, *sigma=3.0*)
Find the center of the peak of offsets

drizzlepac.tweakutils.**gauss**(*x*, *sigma*)
Compute 1-D value of gaussian at position x relative to center.

drizzlepac.tweakutils.**gauss_array**(*nx*, *ny=None*, *fwhm=1.0*, *sigma_x=None*, *sigma_y=None*, *zero_norm=False*)
Computes the 2D Gaussian with size nx*ny.

Parameters

nx : int

ny : int [Default: None]

Size of output array for the generated Gaussian. If ny == None, output will be an array nx X nx pixels.

fwhm : float [Default: 1.0]

Full-width, half-maximum of the Gaussian to be generated

sigma_x : float [Default: None]

sigma_y : float [Default: None]

Sigma_x and sigma_y are the stddev of the Gaussian functions.

zero_norm : bool [Default: False]

The kernel will be normalized to a sum of 1 when True.

Returns

gauss_arr : array

A numpy array with the generated gaussian function

drizzlepac.tweakutils.**isfloat**(*value*)
Return True if all characters are part of a floating point value

drizzlepac.tweakutils.**make_vector_plot**(*coordfile, columns=[1, 2, 3, 4], data=None, figure_id=None, title=None, axes=None, every=1, limit=None, xlower=None, ylower=None, output=None, headl=4, headw=3, xsh=0.0, ysh=0.0, fit=None, scale=1.0, vector=True, textscale=5, append=False, linfit=False, rms=True*)

Convert a XYXYMATCH file into a vector plot or set of residuals plots.

This function provides a single interface for generating either a vector plot of residuals or a set of 4 plots showing residuals. The data being plotted can also be adjusted for a linear fit on-the-fly.

Parameters

coordfile : string

Name of file with matched sets of coordinates. This input file can be a file compatible for use with IRAF's geomap.

columns : list [Default: [0,1,2,3]]

Column numbers for the X,Y positions from each image

data : list of arrays

If specified, this can be used to input matched data directly

title : string

Title to be used for the generated plot

axes : list

List of X and Y min/max values to customize the plot axes

every : int [Default: 1]

Slice value for the data to be plotted

limit : float

Radial offset limit for selecting which sources are included in the plot

xlower : float

ylower : float

Limit in X and/or Y offset for selecting which sources are included in the plot

output : string

Filename of output file for generated plot

headl : int [Default: 4]

Length of arrow head to be used in vector plot

headw : int [Default: 3]

Width of arrow head to be used in vector plot

xsh : float

ysh : float

Shift in X and Y from linear fit to be applied to source positions from the first image

scale : float

Scale from linear fit to be applied to source positions from the first image

fit : array

Array of linear coefficients for rotation (and scale?) in X and Y from a linear fit to be applied to source positions from the first image

vector : bool [Default: True]

Specifies whether or not to generate a vector plot. If False, task will generate a set of 4 residuals plots instead

textscale : int [Default: 5]

Scale factor for text used for labelling the generated plot

append : bool [Default: False]

If True, will overplot new plot on any pre-existing plot

linfit : bool [Default: False]

If True, a linear fit to the residuals will be generated and added to the generated residuals plots

rms : bool [Default: True]

Specifies whether or not to report the RMS of the residuals as a label on the generated plot(s).

drizzlepac.tweakutils.**ndfind_old**(*array, hmin, fwhm, sharplim=[0.2, 1.0], roundlim=[-1, 1], minpix=5, datamax=None*)

Source finding algorithm based on NDIMAGE routines

This function provides a simple replacement for the DAOFIND task.

Parameters

array : arr

Input image as numpy array

hmin : float

Limit for source detection in pixel values

fwhm : float

Full-width half-maximum of the PSF in the image

minpix : int

Minimum number of pixels for any valid source

sharplim : tuple

[Not used at this time]

roundlim : tuple

[Not used at this time]

datamax : float

Maximum good pixel value found in any detected source

Returns

x : arr

Array of detected source X positions (in array coordinates, 0-based)

y : arr

Array of detected source Y positions (in array coordinates, 0-based)

flux : arr

Array of detected source fluxes in pixel values

id : arr

Array of detected source ID numbers

drizzlepac.tweakutils.**parse_atfile_cat**(*input*)
Return the list of catalog filenames specified as part of the input @-file

drizzlepac.tweakutils.**parse_colname**(*colname*)
Common function to interpret input column names provided by the user.

This function translates column specification provided by the user into a column number.

Parameters
colname : :

Column name or names to be interpreted

Returns
cols : list

The return value will be a list of strings.

Notes

This function will understand the following inputs:

```
'1,2,3' or   'c1,c2,c3' or ['c1','c2','c3']
'1-3'   or   'c1-c3'
'1:3'   or   'c1:c3'
'1 2 3' or   'c1 c2 c3'
'1'     or   'c1'
1
```

drizzlepac.tweakutils.**parse_exclusions**(*exclusions*)
Read in exclusion definitions from file named by 'exclusions' and return a list of positions and distances

drizzlepac.tweakutils.**parse_skypos**(*ra*, *dec*)
Function to parse RA and Dec input values and turn them into decimal degrees

Input formats could be:
["nn","nn","nn.nn"] "nn nn nn.nnn" "nn:nn:nn.nn" "nnH nnM nn.nnS" or "nnD nnM nn.nnS" nn.nnnnnnnn "nn.nnnnnnn"

drizzlepac.tweakutils.**plot_zeropoint**(*pars*)
Plot 2d histogram.

Pars will be a dictionary containing:
data, figure_id, vmax, title_str, xp,yp, searchrad

drizzlepac.tweakutils.**radec_hmstodd**(*ra*, *dec*)
Function to convert HMS values into decimal degrees.

This function relies on the astrolib.coords package to perform the conversion to decimal degrees.

Parameters
ra : list or array

List or array of input RA positions

dec : list or array

List or array of input Dec positions

Returns
pos : arr

Array of RA,Dec positions in decimal degrees

See also:

`astrolib.coords`

Notes

Formats of ra and dec inputs supported:

```
["nn","nn","nn.nn"]
"nn nn nn.nnn"
"nn:nn:nn.nn"
"nnH nnM nn.nnS" or "nnD nnM nn.nnS"
```

`drizzlepac.tweakutils.`**`read_ASCII_cols`**(*infile, cols=[1, 2, 3]*)

Interpret input ASCII file to return arrays for specified columns.

Returns

outarr : list of arrays

The return value will be a list of numpy arrays, one for each 'column'.

Notes

The specification of the columns should be expected to have lists for each 'column', with all columns in each list combined into a single entry. For example:

```
cols = ['1,2,3','4,5,6',7]
```

where '1,2,3' represent the X/RA values, '4,5,6' represent the Y/Dec values and 7 represents the flux value for a total of 3 requested columns of data to be returned.

`drizzlepac.tweakutils.`**`read_FITS_cols`**(*infile*, *cols=None*)

Read columns from FITS table

`drizzlepac.tweakutils.`**`readcols`**(*infile*, *cols=None*)

Function which reads specified columns from either FITS tables or ASCII files

This function reads in the columns specified by the user into numpy arrays regardless of the format of the input table (ASCII or FITS table).

Parameters

infile : string

Filename of the input file

cols : string or list of strings

Columns to be read into arrays

Returns

outarr : array

Numpy array or arrays of columns from the table

`drizzlepac.tweakutils.`**`write_shiftfile`**(*image_list*, *filename*, *outwcs='tweak_wcs.fits'*)

Write out a shiftfile for a given list of input Image class objects

12.8 UPDATEHDR: Functions for Updating WCS with New Solutions

The functions in this module support updating the WCS information in distorted images with the alignment solution determined by 'tweakreg' or saved in a shiftfile.

drizzlepac.updatehdr.**create_unique_wcsname**(*fimg*, *extnum*, *wcsname*)

This function evaluates whether the specified wcsname value has already been used in this image. If so, it automatically modifies the name with a simple version ID using wcsname_NNN format.

fimg

[obj] PyFITS object of image with WCS information to be updated

extnum

[int] Index of extension with WCS information to be updated

wcsname

[str] Value of WCSNAME specified by user for labelling the new WCS

Returns

uniqname : str

Unique WCSNAME value

drizzlepac.updatehdr.**update_from_shiftfile**(*shiftfile*, *wcsname=None*, *force=False*)

Update headers of all images specified in shiftfile with shifts from shiftfile.

Parameters

shiftfile : str

Filename of shiftfile.

wcsname : str

Label to give to new WCS solution being created by this fit. If a value of None is given, it will automatically use 'TWEAK' as the label. [Default =None]

force : bool

Update header even though WCS already exists with this solution or wcsname? [Default=False]

drizzlepac.updatehdr.**update_wcs**(*image*, *extnum*, *new_wcs*, *wcsname=''*, *verbose=False*)

Updates the WCS of the specified extension number with the new WCS after archiving the original WCS.

The value of 'new_wcs' needs to be the full HSTWCS object.

Parameters

image : str

Filename of image with WCS that needs to be updated

extnum : int

Extension number for extension with WCS to be updated/replaced

new_wcs : object

Full HSTWCS object which will replace/update the existing WCS

wcsname : str

Label to give newly updated WCS

verbose : bool, int

Print extra messages during processing? [Default: False]

drizzlepac.updatehdr.**updatewcs_with_shift**(*image*, *reference*, *wcsname=None*, *rot=0.0*, *scale=1.0*, *xsh=0.0*, *ysh=0.0*, *fit=None*, *xrms=None*, *yrms=None*, *verbose=False*, *force=False*, *sciext='SCI'*)

Update the SCI headers in 'image' based on the fit provided as determined in the WCS specified by 'reference'. The fit should be a 2-D matrix as generated for use with 'make_vector_plot()'.

Parameters

image : str or PyFITS.HDUList object

Filename, or PyFITS object, of image with WCS to be updated. All extensions with EXTNAME matches the value of the 'sciext' parameter value (by default, all 'SCI' extensions) will be updated.

reference : str

Filename of image/headerlet (FITS file) which contains the WCS used to define the tangent plane in which all the fit parameters (shift, rot, scale) were measured.

wcsname : str

Label to give to new WCS solution being created by this fit. If a value of None is given, it will automatically use 'TWEAK' as the label. If a WCS has a name with this specific value, the code will automatically append a version ID using the format '_n', such as 'TWEAK_1', 'TWEAK_2',or 'TWEAK_update_1'. [Default =None]

rot : float

Amount of rotation measured in fit to be applied. [Default=0.0]

scale : float

Amount of scale change measured in fit to be applied. [Default=1.0]

xsh : float

Offset in X pixels from defined tangent plane to be applied to image. [Default=0.0]

ysh : float

Offset in Y pixels from defined tangent plane to be applied to image. [Default=0.0]

fit : arr

Linear coefficients for fit [Default = None]

xrms : float

RMS of fit in RA (in decimal degrees) that will be recorded as CRDER1 in WCS and header [Default = None]

yrms : float

RMS of fit in Dec (in decimal degrees) that will be recorded as CRDER2 in WCS and header [Default = None]

verbose : bool

Print extra messages during processing? [Default=False]

force : bool

Update header even though WCS already exists with this solution or wcsname? [Default=False]

sciext : string

Value of FITS EXTNAME keyword for extensions with WCS headers to be updated with the fit values. [Default='SCI']

Notes

The algorithm used to apply the provided fit solution to the image involves applying the following steps to the WCS of each of the input image's chips:

1. **compute RA/Dec with full distortion correction for** reference point as (Rc_i,Dc_i)
2. **find the Xc,Yc for each Rc_i,Dc_i and get the difference from the** CRPIX position for the reference WCS as (dXc_i,dYc_i)
3. **apply fit (rot&scale) to (dXc_i,dYc_i) then apply shift, then add** CRPIX back to get new (Xcs_i,Ycs_i) position
4. compute (Rcs_i,Dcs_i) as the sky coordinates for (Xcs_i,Ycs_i)
5. compute delta of (Rcs_i-Rc_i, Dcs_i-Dcs_i) as (dRcs_i,dDcs_i)
6. **apply the fit to the chip's undistorted CD matrix, the apply linear** distortion terms back in to create a new CD matrix
7. add (dRcs_i,dDcs_i) to CRVAL of the reference chip's WCS
8. update header with new WCS values

CHAPTER

THIRTEEN

COORDINATE TRANSFORMATION TASKS

These tasks support transformations of source positions to and from distorted and drizzled images.

13.1 pixtopix: Coordinate transformation to/from drizzled images

This task allows a user to perform coordinate transformations with the full WCS and distortion model to and from drizzled image positions. This task serves as a replacement for the STSDAS.dither task 'tran'.

pixtosky - A module to perform coordinate transformation from pixel coordinates
in one image to pixel coordinates in another frame

License:
http://www.stsci.edu/resources/software_hardware/pyraf/LICENSE

13.1.1 PARAMETERS

inimage
[str] full filename with path of input image, an extension name ['sci',1] should be provided if input is a multi-extension FITS file

outimage
[str, optional] full filename with path of output image, an extension name ['sci',1] should be provided if output is a multi-extension FITS file. If no image gets specified, the input image will be used to generate a default output WCS using stwcs.distortion.util.output_wcs().

direction
[str] Direction of transform (forward or backward). The 'forward' transform takes the pixel positions (assumed to be from the 'input' image) and determines their position in the 'output' image. The 'backward' transform converts the pixel positions (assumed to be from the 'output' image) into pixel positions in the 'input' image.

13.1.2 Optional Parameters

x
[float, optional] X position from image

y
[float, optional] Y position from image

coords
[str, optional] full filename with path of file with starting x,y coordinates

colnames
[str, optional] comma separated list of column names from 'coords' files containing x,y coordinates, respectively. Will default to first two columns if None are specified. Column names for ASCII files will use 'c1','c2',... convention.

separator
[str, optional] non-blank separator used as the column delimiter in the coords file

precision
[int, optional] Number of floating-point digits in output values

output
[str, optional] Name of output file with results, if desired

verbose
[bool] Print out full list of transformation results (default: False)

13.1.3 RETURNS

outx
[float] X position of transformed pixel. If more than 1 input value, then it will be a numpy array.

outy
[float] Y position of transformed pixel. If more than 1 input value, then it will be a numpy array.

13.1.4 NOTES

This module performs a full distortion-corrected coordinate transformation based on all WCS keywords and any recognized distortion keywords from the input image header.

13.1.5 Usage

It can be called from within Python using the syntax:

```
>>> from drizzlepac import pixtopix
>>> outx,outy = pixtopix.tran("input_flt.fits[sci,1]",
                "output_drz.fits[sci,1],"forward",100,100)
```

13.1.6 EXAMPLES

1. **The following command will transform the position 256,256 from**
   'input_flt.fits[sci,1]' into a position on the output image 'output_drz.fits[sci,1]' using:

   ```
   >>> from drizzlepac import pixtopix
   >>> outx,outy = pixtopix.tran("input_file_flt.fits[sci,1]",
                   "output_drz.fits[sci,1],"forward", 256,256)
   ```

2. **The set of X,Y positions from 'output_drz.fits[sci,1]' stored as**
   the 3rd and 4th columns from the ASCII file 'xy_sci1.dat' will be transformed into pixel positions from 'input_flt.fits[sci,1]' and written out to 'xy_flt1.dat' using:

   ```
   >>> from drizzlepac import pixtopix
   >>> x,y = pixtopix.tran("input_flt.fits[sci,1]", "output_drz.fits[sci,1]",
           "backward", coords='xy_sci1.dat', colnames=['c3','c4'],
           output="xy_flt1.dat")
   ```

drizzlepac.pixtopix.**tran**(*inimage*, *outimage*, *direction='forward'*, *x=None*, *y=None*, *coords=None*, *colnames=None*, *separator=None*, *precision=6*, *output=None*, *verbose=True*)

Primary interface to perform coordinate transformations in pixel coordinates between 2 images using STWCS and full distortion models read from each image's header.

13.2 pixtosky: Coordinate transformation to sky coordinates

This task allows a user to perform coordinate transformations with the full WCS and distortion model on source positions from an input image to sky coordinates. This task serves as a replacement for the IRAF.STSDAS task 'xy2rd', albeit with the added capability of understanding the full distortion model provided in the headers of images updated for use with astrodrizzle and tweakreg. pixtosky - A module to perform coordinate transformation from pixel to sky coordinates.

License:

http://www.stsci.edu/resources/software_hardware/pyraf/LICENSE

13.2.1 PARAMETERS

input

[str] full filename with path of input image, an extension name ['sci',1] should be provided if input is a multi-extension FITS file

13.2.2 Optional Parameters

x

[float, optional] X position from input image

y

[float, optional] Y position from input image

coords

[str, optional] full filename with path of file with x,y coordinates

colnames

[str, optional] comma separated list of column names from 'coords' files containing x,y coordinates, respectively. Will default to first two columns if None are specified. Column names for ASCII files will use 'c1','c2',... convention.

separator

[str, optional] non-blank separator used as the column delimiter in the coords file

hms

[bool, optional] Produce output in HH:MM:SS.S format instead of decimal degrees? (default: False)

precision

[int, optional] Number of floating-point digits in output values

output

[str, optional] Name of output file with results, if desired

verbose

[bool] Print out full list of transformation results (default: False)

13.2.3 RETURNS

ra
[float] Right Ascension of pixel. If more than 1 input value, then it will be a numpy array.

dec
[float] Declination of pixel. If more than 1 input value, then it will be a numpy array.

13.2.4 NOTES

This module performs a full distortion-corrected coordinate transformation based on all WCS keywords and any recognized distortion keywords from the input image header.

13.2.5 Usage

It can be called from within Python using the syntax:

```
>>> from drizzlepac import pixtosky
>>> r,d = pixtosky.xy2rd("input_flt.fits[sci,1]",100,100)
```

13.2.6 EXAMPLES

1. **The following command will transform the position 256,256 into a**
   position on the sky for the image 'input_flt.fits[sci,1]' using:

   ```
   >>> from drizzlepac import pixtosky
   >>> r,d = pixtosky.xy2rd("input_file_flt.fits[sci,1]", 256,256)
   ```

2. **The set of X,Y positions from 'input_flt.fits[sci,1]' stored as**
   the 3rd and 4th columns from the ASCII file 'xy_sci1.dat' will be transformed and written out to 'radec_sci1.dat' using:

   ```
   >>> from drizzlepac import pixtosky
   >>> r,d = pixtosky.xy2rd("input_flt.fits[sci,1]", coords='xy_sci1.dat',
       colnames=['c3','c4'], output="radec_sci1.dat")
   ```

drizzlepac.pixtosky.**xy2rd**(*input*, *x=None*, *y=None*, *coords=None*, *colnames=None*, *separator=None*, *hms=True*, *precision=6*, *output=None*, *verbose=True*)
Primary interface to perform coordinate transformations from pixel to sky coordinates using STWCS and full distortion models read from the input image header.

13.3 skytopix: Coordinate transformation from sky coordinates

This task allows a user to perform coordinate transformations with the full WCS and distortion model on source positions from sky coordinates to the WCS defined by an image. This task serves as a replacement for the IRAF.STSDAS task 'rd2xy', albeit with the added capability of understanding the full distortion model provided in the headers of images updated for use with astrodrizzle and tweakreg. skytopix - A module to perform coordinate transformation from sky to pixel coordinates.

License:
http://www.stsci.edu/resources/software_hardware/pyraf/LICENSE

13.3.1 PARAMETERS

input
[str] full filename with path of input image, an extension name ['sci',1] should be provided if input is a multi-extension FITS file

13.3.2 Optional Parameters

ra
[string, optional] RA from input image

dec
[string, optional] Dec from input image

coordfile
[str, optional] full filename with path of file with sky coordinates

colnames
[str, optional] comma separated list of column names from 'coordfile' files containing x,y coordinates, respectively. Will default to first two columns if None are specified. Column names for ASCII files will use 'c1','c2',... convention.

separator
[str, optional] non-blank separator used as the column delimiter in the coords file

precision
[int, optional] Number of floating-point digits in output values

output
[str, optional] Name of output file with results, if desired

verbose
[bool] Print out full list of transformation results (default: False)

13.3.3 RETURNS

x
[float] X position of pixel. If more than 1 input value, then it will be a numpy array.

y
[float] Y position of pixel. If more than 1 input value, then it will be a numpy array.

13.3.4 NOTES

This module performs a full distortion-corrected coordinate transformation based on all WCS keywords and any recognized distortion keywords from the input image header.

13.3.5 Usage

It can be called from within Python using the syntax:

```
>>> from drizzlepac import skytopix
>>> x,y = skytopix.rd2xy("input_flt.fits[sci,1]","00:22:36.79","-72:4:9.0")
```

13.3.6 EXAMPLES

1. **The following command will transform the position 00:22:36.79 -72:4:9.0 into a**
   position on the image 'input_flt.fits[sci,1]' using:

   ```
   >>> from drizzlepac import skytopix
   >>> x,y = skytopix.rd2xy("input_file_flt.fits[sci,1]", "00:22:36.79","-72:4:9.0")
   ```

2. **The set of sky positions from 'input_flt.fits[sci,1]' stored as**
   the 3rd and 4th columns from the ASCII file 'radec_sci1.dat' will be transformed and written out to 'xy_sci1.dat' using:

   ```
   >>> from drizzlepac import skytopix
   >>> x,y = skytopix.rd2xy("input_flt.fits[sci,1]", coordfile='radec_sci1.dat',
       colnames=['c3','c4'], output="xy_sci1.dat")
   ```

drizzlepac.skytopix.**rd2xy**(*input*, *ra=None*, *dec=None*, *coordfile=None*, *colnames=None*, *precision=6*, *output=None*, *verbose=True*)

Primary interface to perform coordinate transformations from pixel to sky coordinates using STWCS and full distortion models read from the input image header.

CHAPTER

FOURTEEN

ACS HEADER UPDATE TASK

A task, 'updatenpol', has been written to automate the updating of ACS image headers with the filename of the appropriate NPOLFILE based on the DGEOFILE specified in the image header. This task should be used to update all ACS images prior to processing them with 'astrodrizzle'.

14.1 Updatenpol

This task will update the header of ACS file(s) with the names of the NPOLFILE and D2IMFILE reference files for use with the new C version of MultiDrizzle (astrodrizzle). **updatenpol**: Update the header of ACS file(s) with the names of new NPOLFILE and D2IMFILE reference files for use with the C version of MultiDrizzle (astrodrizzle).

License

http://www.stsci.edu/resources/software_hardware/pyraf/LICENSE

Usage

This task can be run from the operating system command line with:

```
updatenpol [options] input [refdir]
```

Command-line Options

input
The specification of the files to be updated, either as a single filename, an ASN table name, or wild-card specification of a list of files.

refdir
The name of the directory containing all the new reference files (**_npl.fits* and **_d2i.fits* files). If no directory is given, it will look in *jref$* by default.

`-h`
Print the help (this text).

`-l`
If specified, copy NPOLFILEs and D2IMFILEs to local directory for use with the input files.

`-i`
If specified, the program will interactively request the exact names of the NPOLFILE and D2IMFILE reference files to be used for updating the header of each file. The value of 'refdir' will be ignored in interactive mode.

Warning: It will ask for the names of the NPOLFILE and D2IMFILE for EACH separate INPUT file when the option *-i* has been specified.

Example

1. This command will update all the FLT files in the current directory with the new NPOLFILE and D2IMFILE reference files found in the 'myjref' directory as defined in the environment:

```
updatenpol *flt.fits myjref$
```

Compatability with MultiDrizzle

The new version of MultiDrizzle (*astrodrizzle*) and `updatewcs` only work with the new NPOLFILE reference file for the DGEO correction (to replace the use of DGEOFILE). In fact, astrodrizzle has been extensively modified to prompt the user with a very lengthy explanation on whether it should stop and allow the user to update the header or continue without applying the DGEO correction under circumstances when the NPOLFILE keyword can not be found for ACS.

`drizzlepac.updatenpol.`**`find_d2ifile`**(*flist*, *detector*)

Search a list of files for one that matches the detector specified.

`drizzlepac.updatenpol.`**`find_npolfile`**(*flist*, *detector*, *filters*)

Search a list of files for one that matches the configuration of detector and filters used.

`drizzlepac.updatenpol.`**`help`**(*file=None*)

Print out syntax help for running updatenpol

Parameters

file : str (Default = None)

If given, write out help to the filename specified by this parameter Any previously existing file with this name will be deleted before writing out the help.

`drizzlepac.updatenpol.`**`run`**(*configobj=None*, *editpars=False*)

Teal interface for running this code.

`drizzlepac.updatenpol.`**`update`**(*input*, *refdir='jref$'*, *local=None*, *interactive=False*, *wcsupdate=True*)

Updates headers of files given as input to point to the new reference files NPOLFILE and D2IMFILE required with the new C version of MultiDrizzle.

Parameters

input : string or list

Name of input file or files acceptable forms:

- single filename with or without directory
- @-file
- association table
- python list of filenames
- wildcard specification of filenames

refdir : string

Path to directory containing new reference files, either environment variable or full path.

local : boolean

Specifies whether or not to copy new reference files to local directory for use with the input files.

interactive : boolean

Specifies whether or not to interactively ask the user for the exact names of the new reference files instead of automatically searching a directory for them.

updatewcs : boolean

Specifies whether or not to update the WCS information in this file to use the new reference files.

Notes

Warning: This program requires access to the *jref$* directory in order to evaluate the DGEOFILE specified in the input image header. This evaluation allows the program to get the information it needs to identify the correct NPOLFILE.

The use of this program now requires that a directory be set up with all the new NPOLFILE and D2IMFILE reference files for ACS (a single directory for all files for all ACS detectors will be fine, much like jref). Currently, all the files generated by the ACS team has initially been made available at:

```
/grp/hst/acs/lucas/new-npl/
```

The one known limitation to how this program works comes from confusion if more than 1 file could possibly be used as the new reference file. This would only happen when NPOLFILE reference files have been checked into CDBS multiple times, and there are several versions that apply to the same detector/filter combination. However, that can be sorted out later if we get into that situation at all.

Examples

1. A set of associated images specified by an ASN file can be updated to use the NPOLFILEs and D2IMFILE found in the local directory defined using the *myjref$* environment variable under PyRAF using:

```
>>>import updatenpol
>>>updatenpol.update('j8bt06010_asn.fits', 'myref$')
```

2. Another use under Python would be to feed it a specific list of files to be updated using:

```
>>> updatenpol.update(['file1_flt.fits','file2_flt.fits'],'myjref$')
```

3. Files in another directory can also be processed using:

```
>>> updatenpol.update('data$*flt.fits','../new/ref/')
```

CHAPTER

FIFTEEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

d

s

d

s

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X