



subpixal Documentation

Release 0.1.0 (2019-09-30 18:26:45 -0400)

Mihai Cara

Dec 16, 2019

CONTENTS

1	Content	3
1.1	Image Alignment	3
1.2	Source Catalogs	6
1.3	Image Resampling	12
1.4	Source Cutouts	14
1.5	Blot Algorithm for Cutouts	20
1.6	Image Cross-Correlation and Interlacing	21
1.7	Centroid Algorithm	22
1.8	Utilities used by <code>subpixal</code>	23
1.9	LICENSE	24
2	Development Notes	27
2.1	Release Notes	27
3	Indices and tables	29
	Python Module Index	31
	Index	33

`subpixal` is a package that provides tools for **SUB-PIXel** cross-correlation image **AL**ignment using algorithms developed by Andrew Fruchter and Rebekah Hounsell. This package also provides tools for correcting image `FITS` `WCS` using known linear transformations.

CONTENT

1.1 Image Alignment

Main module that performs image alignment and WCS correction.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

```
subpixal.align.align_images(catalog, resample, wcslin=None, fitgeom='general',
                             nclip=3, sigma=3.0, nmax=10, eps_shift=0.003,
                             use_weights=True, cc_type='NCC', wc-
                             sname='SUBPIXAL', wcsupdate='batch', com-
                             bine_seg_mask=True, iterative=False, history='last')
```

Perform *relative* image alignment using sub-pixel cross-correlation. Image alignment is performed by adjusting each image's WCS so that images align on the sky (i.e., sources from the catalog overlap). Input image data (provided through the `resample` parameter) are not changed.

Parameters

catalog [catalogs.ImageCatalog] A catalog object of ImageCatalog-derived type. This object will hold source-finding and source filtering parameters and should be able to find sources in provided images on demand.

resample [resample.Resample] An object of resample.Resample-derived type that can resample its images onto a common output grid.

wcslin [astropy.wcs.WCS, None, optional] A [WCS](http://docs.astropy.org/en/stable/api/astropy.wcs.WCS.html#astropy.wcs.WCS) (<http://docs.astropy.org/en/stable/api/astropy.wcs.WCS.html#astropy.wcs.WCS>) object that does not have non-linear distortions. This WCS defines a tangent plane in which image alignment will be performed. When not provided or set to `None` (<https://docs.python.org/3/library/constants.html#None>), it is set to `drz_cutouts[0].wcs`.

fitgeom [{ 'shift', 'rscale', 'general' }, optional] The fitting geometry to be used in fitting cutout displacements. This parameter is used in fitting the offsets, rotations and/or scale changes from the matched object lists. The 'general' fit geometry allows for independent scale and rotation for each axis.

nclip [int, optional] Number (a non-negative integer) of clipping iterations in fit.

sigma [float, optional] Clipping limit in sigma units.

nmax [int, tuple of two int, optional] A positive integer number indicating the number of resample-alignment iterations to be performed. After detecting that resampled images do not change significantly, the algorithm will automatically switch to a faster resampling `fast_add_image()` and `fast_drop_image()` methods instead of performing “full” resample that includes sky re-computation, cosmic ray detection, etc.

When `nmax` is a tuple of integers, first number indicates the maximum number of iterations to be performed and the second number indicates the maximum number of iterations with “full” resample to be performed.

eps_shift [float, optional] The algorithm will stop iterations when found shifts are below `eps_shift` value for all images.

use_weights [bool, optional] Indicates whether to perform a weighted fit when catalog contains a 'weights' column.

cc_type [{ 'CC', 'NCC', 'ZNCC' }, optional] Cross-correlation algorithm to be used. 'CC' indicates the “standard” cross-correlation algorithm. 'NCC' refers to the normalized cross-correlation and 'ZNCC' refers to the zero-normalized cross-correlation, see, e.g., [Terminology in image processing](https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing) (https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing).

wcsname [str, None, optional] Label to give newly updated WCS. The default value will set the WCS name to SUBPIXAL.

wcsupdate [{ 'otf', 'batch' }, optional] Indicates when to update the WCS of an image: on-the-fly ('otf') setting will update image WCS as soon as the image was aligned while the 'batch' mode will first compute WCS corrections for all images and *then* will update their WCS at once. With 'otf' setting, next image (within the same iteration) will be aligned to a drizzled image obtained using (at least some) already aligned (in this iteration) images.

combine_seg_mask: bool, optional Indicates whether to combine segmentation mask with cutout's mask. When `True` (<https://docs.python.org/3/library/constants.html#True>), segmentation image is used to create a mask that indicates “good” pixels in the image. This mask is combined with cutout's mask.

iterative [bool, optional] If `True` (<https://docs.python.org/3/library/constants.html#True>), after each iteration user will be asked whether to continue or stop alignment process.

history [{ 'all', 'last', None }] On return this function returns “fit history” containing information that can be used to analyze the goodness of fit. When `history` is 'all', then info from each iteration is saved. When `history` is 'last' only info for the last iteration is saved, and when `history` is `None` (<https://docs.python.org/3/library/constants.html#None>), no information is saved.

Returns

fit_history [list of dict] A list of Python dictionaries containing fit information as well as “image” information such as image cutouts, blots, cross-correlation image, etc.

```
subpixal.align.find_linear_fit(img_cutouts, drz_cutouts, wcslin=None,
                               fitgeom='general', nclip=3, sigma=3.0,
                               use_weights=True, cc_type='NCC')
```

Perform linear fit to displacements (found using cross-correlation) between `img_cutouts` and “blot” of `drz_cutouts` onto `img_cutouts`.

Parameters

img_cutouts [Cutout] Cutouts whose WCS should be aligned.

drz_cutouts [Cutout] Cutouts that serve as “reference” to which `img_cutouts` will be aligned.

wcslin [astropy.wcs.WCS, None, optional] A `WCS` (<http://docs.astropy.org/en/stable/api/astropy.wcs.WCS.html#astropy.wcs.WCS>) object that does not have non-linear distortions. This `WCS` defines a tangen plane in which image alignment will be performed. When not provided or set to `None` (<https://docs.python.org/3/library/constants.html#None>), internally `wcslin` will be set to `drz_cutouts[0].wcs`.

fitgeom [{‘shift’, ‘rscale’, ‘general’}, optional] The fitting geometry to be used in fitting cutout displacements. This parameter is used in fitting the offsets, rotations and/or scale changes from the matched object lists. The ‘general’ fit geometry allows for independent scale and rotation for each axis.

nclip [int, optional] Number (a non-negative integer) of clipping iterations in fit.

sigma [float, optional] Clipping limit in sigma units.

use_weights [bool, optional] Indicates whether to perform a weighted fit when all input `drz_cutouts.src_weight` are not `None` (<https://docs.python.org/3/library/constants.html#None>).

cc_type [{‘CC’, ‘NCC’, ‘ZNCC’}, optional] Cross-correlation algorithm to be used. ‘CC’ indicates the “standard” cross-correlation algorithm. ‘NCC’ refers to the normalized cross-correlation and ‘ZNCC’ refers to the zero-normalized cross-correlation, see, e.g., [Terminology in image processing](https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing) (https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing).

Returns

fit [dict] A dictionary of various fit parameters computed during the fit. Use `fit.keys()` to find which parameters are being returned.

interlaced_cc [numpy.ndarray] Interlaced (super-sampled) cross-correlation image. This is provided as a diagnostic tool for debugging purposes.

nonshifted_blts [Cutout] A list of cutouts of blotted `drz_cutouts` without applying any sub-pixel shifts. This is provided as a diagnostic tool for debugging

purposes.

`subpixal.align.correct_wcs(imwcs, wcslin, rotmat, shifts, fitgeom)`

Correct input WCS using supplied linear transformations defined in a linear WCS. This function modifies `imwcs` with the corrected WCS parameters.

`subpixal.align.update_image_wcs(image, ext, wcs, wcsname=None)`

Updates the WCS of the specified extension with the new WCS after archiving the original WCS.

Parameters

image [str, `astropy.io.fits.HDUList`] Filename of image with WCS that needs to be updated

ext [int, str or tuple of (string, int)] The key identifying the HDU. If `ext` is a tuple, it is of the form `(name, ver)` where `ver` is an `EXTVER` value that must match the HDU being searched for.

If the key is ambiguous (e.g. there are multiple ‘SCI’ extensions) the first match is returned. For a more precise match use the `(name, ver)` pair.

If even the `(name, ver)` pair is ambiguous (it shouldn’t be but it’s not impossible) the numeric index must be used to index the duplicate HDU.

wcs [object] Full `HSTWCS` object which will replace/update the existing WCS

wcsname [str, None, optional] Label to give newly updated WCS. The default value will set the WCS name to `SUBPIXAL`.

1.2 Source Catalogs

A module that manages catalogs and source finding algorithms (i.e., `SExtractor` source finding).

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

class `subpixal.catalogs.ImageCatalog`

A class for finding sources in images and handling catalog data: storing, filtering, and retrieving sources.

append_filters (*self*, *fcond*)

Add one or more conditions for *selecting* sources from the raw catalog to already set filters. See `set_filters()` for description of parameter `fcond`.

catalog (*self*)

Get catalog (after applying masks and selection filters).

compute_position_std (*self*, *catalog*)

This function is called to compute source position error estimate. This function uses the following simplified estimate: $\sigma_{\text{pos}} = \sigma_{\text{Gaussian}} / \text{SNR} = \text{FWHM} / (2\sqrt{2 \ln 2} \text{SNR})$. Sub-classes can implement more accurate position error computation.

Parameters

catalog [astropy.table.Table] A table containing *required_colnames* columns.

Returns

pos_std [numpy.ndarray] Position error computed from input catalog data.

compute_weights (*self*, *catalog*)

This function is called to compute source weights in a catalog. Currently, all weights are set equal to 1. Sub-classes should implement more meaningful weight computation.

Parameters

catalog [astropy.table.Table] A table containing *required_colnames* columns.

Returns

weights [numpy.ndarray] Weights computed from input catalog data.

abstract execute (*self*)

Find sources in the image. Compute catalog applying masks and selecting only sources that satisfy all set filters.

property filters

Get a list of all active selection filters.

get_segmentation_image (*self*)

Get segmentation image used to identify catalog's sources.

property image_extn

Get image extension number when the image was set using a string file name. When image was set (in `py:meth:set_image`) using a `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>), this property is `None` (<https://docs.python.org/3/library/constants.html#None>).

property mask_type

Get mask type: 'coords', 'image', or `None` (<https://docs.python.org/3/library/constants.html#None>) (mask not set).

remove_all_filters (*self*)

Remove all selection filters.

remove_filter (*self*, *key*, *op=None*)

Remove a specific filter by column name and, optionally, by comparison operator.

Parameters

key [str] Column name to which selection criteria (filter) is applied. If more conditions match a column name vale, all of them will be removed.

op [str, optional] Specifies the comparison operation used in a filter. This allows narrowing down which filters should be removed.

property required_colnames

Get a list of the minimum column names that are *required* to be present in the raw catalog **after** catalog column name mapping has been applied.

set_default_filters (*self*)

Set default source selection criteria.

set_filters (*self*, *fcond*)

Set conditions for *selecting* sources from the raw catalog.

Parameters

fcond [tuple, list of tuples] Each selection condition must be specified as a tuple of the form (colname, cond, value) OR (colname, nrows) where:

- colname is a column name from the raw catalog **after** catalog column name mapping has been applied. Use `rawcat_colnames` to get a list of available column names.
- cond is a **string** representing a selection condition, i.e., a comparison operator. The following operators are supported: ['>', '>=', '==', '!=', '<', '<=', 'h', 'l']. The 'h' or 'l' operators are used to select a specific number of rows (specified by the value) that have highest or lowest values in the column specified by colname. Selection of highest/lowest values is performed last, after all other comparison-based filters have been applied.
- value is a numeric value to be used for comparison of column values. When cond is either 'h' or 'l', this value must be a *positive integer* number of rows to be .

Multiple selection conditions can be provided as a list of the condition tuples described above.

set_image (*self*, *image*)

Set image to be used for source finding.

Parameters

image: numpy.ndarray, str When setting an image either a `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>) of image data or a string file name is acceptable. Image file name may be followed by an extension specification such as 'file1.fits[1]' or 'file1.fits[(sci,1)]' (by default, the first image-like extension will be used).

set_mask (*self*, *mask*)

Get/Set mask used to ignore (mask) “bad” sources from the catalog.

Parameters

mask [str, tuple of two 1D lists of int, 2D numpy.ndarray] A mask can be provided in several ways:

- When mask is a string, it is assumed to be the name of a simple FITS file containing a boolean mask indicating “bad” pixels using value `True` (<https://docs.python.org/3/library/constants.html#True>) (=ignore these pixels) and “good” pixels using value `False`

(<https://docs.python.org/3/library/constants.html#False>) (=no need to mask).

- `mask` can also be provided directly as a boolean 2D “image” in the form of a boolean `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>).
- Finally, `mask` can be a tuple of exactly two lists (or 1D `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>)) containing **integer** coordinates of the “pixels” to be masked as “bad”. Any source with coordinates within such a “pixel” will be excluded from the catalog.

```
class subpixal.catalogs.SExImageCatalog (image=None,          sexconfig=None,
                                         max_stellarity=1.0,      sextrac-
                                         tor_cmd='sex')
```

A catalog class specialized in finding sources using `SExtractor` and then loading and processing raw `SExtractor` catalogs and its output files.

Parameters

image [str] A FITS image file name.

sexconfig [str] File name of the `SExtractor` configuration file to be used for finding sources in the `image`.

max_stellarity [float, None, optional] Maximum stellarity for selecting sources from the catalog. When `max_stellarity` is `None` (<https://docs.python.org/3/library/constants.html#None>), source filtering by ‘stellarity’ is turned off.

sextractor_cmd [str, optional] Command to invoke `SExtractor`.

append_filters (*self*, *fcond*)

Add one or more conditions for *selecting* sources from the raw catalog to already set filters. See `set_filters()` for description of parameter `fcond`.

property catalog

Get catalog (after applying masks and selection filters).

compute_position_std (*self*, *catalog*)

This function is called to compute source position error estimate. This function uses the following simplified estimate: $\sigma_{\text{pos}} = \sigma_{\text{Gaussian}}/\text{SNR} = \text{FWHM}/(2\sqrt{2\ln 2}\text{SNR})$. Sub-classes can implement more accurate position error computation.

Parameters

catalog [astropy.table.Table] A table containing `required_colnames` columns.

Returns

pos_std [numpy.ndarray] Position error computed from input catalog data.

compute_weights (*self*, *catalog*)

This function is called to compute source weights in a catalog. This function estimates weights as $1/\sigma_{\text{pos}}$.

Parameters

catalog [astropy.table.Table] A table containing *required_colnames* columns.

Returns

weights [astropy.table.Column] Weights computed from input catalog data.

execute (*self*)

Find sources in the image. Compute catalog applying masks and selecting only sources that satisfy all set filters.

property filters

Get a list of all active selection filters.

get_segmentation_image (*self*)

Get segmentation file name stored in the SExtractor's configuration file or *None* (<https://docs.python.org/3/library/constants.html#None>).

property image_extn

Get image extension number when the image was set using a string file name. When image was set (in `py:meth:set_image`) using a `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>), this property is *None* (<https://docs.python.org/3/library/constants.html#None>).

property mask_type

Get mask type: 'coords', 'image', or *None* (<https://docs.python.org/3/library/constants.html#None>) (mask not set).

remove_all_filters (*self*)

Remove all selection filters.

remove_filter (*self*, *key*, *op=None*)

property required_colnames

Get a list of the minimum column names that are *required* to be present in the raw catalog **after** catalog column name mapping has been applied.

set_default_filters (*self*)

Sets default filters for selecting sources from the raw catalog.

Default selection criteria are: `flux > 0 AND fwhm > 0 AND semi-major-a > 0 AND semi-major-b > 0 (AND stellarity <= max_stellarity, if max_stellarity is not None` (<https://docs.python.org/3/library/constants.html#None>)).

set_filters (*self*, *fcond*)

Set conditions for *selecting* sources from the raw catalog.

Parameters

fcond [tuple, list of tuples] Each selection condition must be specified as a tuple of the form (colname, cond, value) OR (colname, nrows) where:

- colname is a column name from the raw catalog **after** catalog column name mapping has been applied. Use `rawcat_colnames` to get a list of available column names.
- cond is a **string** representing a selection condition, i.e., a comparison operator. The following operators are supported: ['>', '>=', '==', '!=', '<', '<=', 'h', 'l']. The 'h' or 'l' operators are used to select a specific number of rows (specified by the value) that have highest or lowest values in the column specified by colname. Selection of highest/lowest values is performed last, after all other comparison-based filters have been applied.
- value is a numeric value to be used for comparison of column values. When cond is either 'h' or 'l', this value must be a *positive integer* number of rows to be .

Multiple selection conditions can be provided as a list of the condition tuples described above.

set_image (self, image)

Set image to be used for source finding.

Parameters

image: `numpy.ndarray`, `str` When setting an image either a `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>) of image data or a string file name is acceptable. Image file name may be followed by an extension specification such as 'file1.fits[1]' or 'file1.fits[(sci,1)]' (by default, the first image-like extension will be used).

set_mask (self, mask)

Get/Set mask used to ignore (mask) “bad” sources from the catalog.

Parameters

mask [str, tuple of two 1D lists of int, 2D `numpy.ndarray`] A mask can be provided in several ways:

- When mask is a string, it is assumed to be the name of a simple FITS file containing a boolean mask indicating “bad” pixels using value `True` (<https://docs.python.org/3/library/constants.html#True>) (=ignore these pixels) and “good” pixels using value `False` (<https://docs.python.org/3/library/constants.html#False>) (=no need to mask).
- mask can also be provided directly as a boolean 2D “image” in the form of a boolean `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>).

- Finally, `mask` can be a tuple of exactly two lists (or 1D `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>)) containing **integer** coordinates of the “pixels” to be masked as “bad”. Any source with coordinates within such a “pixel” will be excluded from the catalog.

property `sexconfig`

Set/Get `SExtractor` configuration file.

1.3 Image Resampling

A module that manages resampling of images onto a common output frame and also “inverse” blotting.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

class `subpical.resample.Resample` (*config=None, **kwargs*)

An abstract class providing interface for resampling and combining sets of images onto a rectified frame.

property `computed_sky`

abstract `execute` (*self*)

Run resampling algorithm.

abstract `fast_add_image` (*self, add_file_name*)

Re-calculate resampled image using all input images and adding another image to the list of input images specified by the `add_file_name` parameter.

Parameters

`add_file_name` [str] File name of the image to be added to the input image list when re-calculating the resampled image.

abstract `fast_drop_image` (*self, drop_file_name*)

Re-calculate resampled image using all input images other than the one specified by `drop_file_name`.

Parameters

`drop_file_name` [str] File name of the image to be dropped from the list of input images when re-calculating the resampled image.

property `input_image_names`

Get an `OrderedDict` of input file names and image extensions or `None` (<https://docs.python.org/3/library/constants.html#None>).

property `output_crclean`

Get file names of the Cosmic Ray (CR) cleaned images (if any).

property `output_ctx`

Get output file name for context data file or `None` (<https://docs.python.org/3/library/constants.html#None>).

property output_sci

Get output file name for output science image or `None`
(<https://docs.python.org/3/library/constants.html#None>).

property output_wht

Get output file name for output weight image or `None`
(<https://docs.python.org/3/library/constants.html#None>).

abstract property reference_image

Get/Set Reference image. When `reference_image` is `None`
(<https://docs.python.org/3/library/constants.html#None>), output WCS and grid are computed automatically.

abstract set_config_parameters (*self*, ***kwargs*)

Override individual configuration parameters.

class `subpixal.resample.Drizzle` (*config=None*, ***kwargs*)

execute (*self*)

Run resampling algorithm.

fast_add_image (*self*, *add_file_name*)

Re-calculate resampled image using all input images and adding another image to the list of input images specified by the `add_file_name` parameter.

Parameters

add_file_name [str] File name of the image to be added to the input image list when re-calculating the resampled image.

fast_drop_image (*self*, *drop_file_name*)

Re-calculate resampled image using all input images other than the one specified by `drop_file_name`.

Parameters

drop_file_name [str] File name of the image to be dropped from the list of input images when re-calculating the resampled image.

fast_replace_image (*self*, *drop_file_name*, *add_file_name*)

Re-calculate resampled image using all input images and adding another image to the list of input images specified by the `add_file_name` parameter.

Parameters

add_file_name [str] File name of the image to be added to the input image list when re-calculating the resampled image.

property reference_image

Get/Set Reference image. When `reference_image` is `None`
(<https://docs.python.org/3/library/constants.html#None>), output WCS and grid are computed automatically.

set_config (*self*, *config=None*, ***kwargs*)

```
set_config_parameters(self, **kwargs)
    Override individual configuration parameters.

taskname = 'astrodrizzle'
```

1.4 Source Cutouts

A module that provides tools for creating and mapping image cutouts.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthel.stsci.edu) (<https://hsthel.stsci.edu>))

License *LICENSE*

```
class subpixal.cutout.Cutout(data, wcs, blc=(0, 0), trc=None, src_pos=None,
                             src_weight=None, dq=None, weight=None, src_id=0,
                             data_units='rate', exptime=1, mode='strict', fill-
                             val=nan)
```

This is a class designed to facilitate work with image cutouts. It holds both information about the cutout (location in the image) as well as information about the image and source: source ID, exposure time, image units, WCS, etc.

This class also provides convenience tools for creating cutouts, saving them to or loading from files, and for converting pixel coordinates to world coordinates (and vice versa) using cutout's pixel grid while preserving all distortion corrections from image's WCS.

Parameters

data: `numpy.ndarray` Image data from which the cutout will be extracted.

wcs: `astropy.wcs.WCS` World Coordinate System object describing coordinate transformations from image's pixel coordinates to world coordinates.

blc: **tuple of two int** Bottom-Left Corner coordinates (x, y) in the data of the cutout to be extracted.

trc: **tuple of two int, None, optional** Top-Right Corner coordinates (x, y) in the data of the cutout to be extracted. Pixel with the coordinates trc is included. When trc is set to `None` (<https://docs.python.org/3/library/constants.html#None>), trc is set to the shape of the data image: (nx, ny).

src_pos: **tuple of two int, None, optional** Image coordinates (x, y) in the input “data” image of the source contained in this cutout. If src_pos is set to the default value (`None` (<https://docs.python.org/3/library/constants.html#None>)), then it will be set to the center of the cutout.

Warning: TODO: The algorithm for src_pos computation most likely will need to be revised to obtain better estimates for the position of the source in the cutout.

src_weight [float, None, optional] The weight of the source in the cutout to be used in alignment when fitting geometric transformations.

dq: numpy.ndarray Data quality array associated with image data. If provided, this array will be cropped the same way as image data and stored within the `Cutout` object.

weight: numpy.ndarray Weight array associated with image data. If provided, this array will be cropped the same way as image data and stored within the `Cutout` object.

src_id [any type, None] Anything that can be used to associate the source being extracted with a record in a catalog. This value is simply stored within the `Catalog` object.

data_units: {'counts', 'rate'}, optional Indicates the type of data units: count-like or rate-like (counts per unit of time). This provides the information necessary for unit conversion when needed.

exptime: float, optional Exposure time of image `imdata`.

mode: {'strict', 'fill'} Allowed overlap between extraction rectangle for the cutout and the input image. When mode is 'strict' then a `PartialOverlapError` error will be raised if the extraction rectangle is not *completely* within the boundaries of input image. When mode is 'fill', then parts of the cutout that are outside the boundaries of the input image will be filled with the value specified by the `fillval` parameter.

fillval: scalar All elements of the cutout that are outside the input image will be assigned this value. This parameter is ignored when mode is set to 'strict'.

Raises

`NoOverlapError` When cutout is completely outside of the input image.

`PartialOverlapError` When cutout only partially overlaps input image and mode is set to 'strict'.

DEFAULT_ACCURACY = 1e-05

DEFAULT_MAXITER = 50

DEFAULT_QUIET = True

property blc

Set/Get coordinate of the bottom-left corner.

property cutout_src_pos

Get/set source position in the *cutout's image coordinates*.

property data

Get image data.

property data_units

Get/Set image data units. Possible values are: 'rate' or 'counts'.

property dq

Set/Get cutout's data quality.

property dx

Set/Get displacement of the image grid along the X-axis in pixels.

property dy

Set/Get displacement of the image grid along the Y-axis in pixels.

property exptime

Get/Set exposure time.

property extraction_slice

Get slice object that shows the slice *in the input data array* used to extract the cutout.

get_bbox(*self*, *wrt*='orig')

Get a `numpy.ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>) of pixel coordinates of vertices of the bounding box. The returned array has the shape (4, 2) and contains the coordinates of the outer corners of pixels (centers of pixels are considered to have integer coordinates).

Parameters

wrt [{ 'orig', 'blc', 'world' }, optional]

property height

Get width of the cutout.

property insertion_slice

Get slice object that shows the slice *in the cutout data array* into which image data were placed. This slice coincides with the entire cutout data array when `mode` is 'strict' but can point to a smaller region when `mode` is 'fill'.

property mask

Set/Get cutout's mask.

property naxis

Get FITS NAXIS property of the cutout.

pix2world(*self*, **args*, *origin*=0)

Convert `_cutout_`'s pixel coordinates to world coordinates.

property pscale

Get pixel scale in the tangent plane at the reference point.

property src_id

Set/Get source ID.

property src_pos

Get/set source position in the *cutout's image*.

property src_weight

Get/set source's weight for fitting geometric transformations.

property trc

Set/Get coordinate of the top-right corner.

property wcs

Get image's WCS from which the cutout was extracted.

property weight

Set/Get cutout's pixel weight.

property width

Get width of the cutout.

world2pix (*self*, **args*, *origin*=0)

Convert world coordinates to `_cutout_`'s pixel coordinates.

```
subpixal.cutout.create_primary_cutouts(catalog, segmentation_image, imdata,
                                       imwcs, imdq=None, dqbitmask=0,
                                       imweight=None, data_units='counts',
                                       exptime=1, pad=1)
```

A function for creating first-order cutouts from a (drizzle-)combined image given a source catalog and a segmentation image.

Parameters

catalog [ImageCatalog, astropy.table.Table] A table of sources which need to be extracted. catalog must contain a column named 'id' which contains IDs of segments from the segmentation_image. If catalog is an `astropy.table.Table` (<http://docs.astropy.org/en/stable/api/astropy.table.Table.html#astropy.table.Table>), then its meta attribute may contain an optional 'weight_colname' item indicating which column in the table shows source weight. If not provided, unweighted fitting will be performed.

segmentation_image: numpy.ndarray A 2D segmentation image identifying sources from the catalog in imdata.

imdata: numpy.ndarray Image data array.

imwcs: astropy.wcs.WCS World coordinate system of image imdata.

imdq: numpy.ndarray, None, optional Data quality (DQ) array corresponding to imdata.

dqbitmask [int, str, None, optional] Integer sum of all the DQ bit values from the input imdq DQ array that should be considered “good” when building masks for cutouts. For example, if pixels in the DQ array can be combinations of 1, 2, 4, and 8 flags and one wants to consider DQ “defects” having flags 2 and 4 as being acceptable, then dqbitmask should be set to 2+4=6. Then a DQ pixel having values 2,4, or 6 will be considered a good pixel, while a DQ pixel with a value, e.g., 1+2=3, 4+8=12, etc. will be flagged as a “bad” pixel.

Alternatively, one can enter a comma- or '+'-separated list of integer bit flags that should be added to obtain the final “good” bits. For example, both 4, 8 and 4+8 are equivalent to setting dqbitmask to 12.

Default value (0) will make *all* non-zero pixels in the DQ mask to be considered

“bad” pixels, and the corresponding image pixels will be flagged in the `mask` property of the returned cutouts.

Set `dqbitmask` to `None`

(<https://docs.python.org/3/library/constants.html#None>) to not consider DQ array when computing cutout’s `mask`.

In order to reverse the meaning of the `dqbitmask` parameter from indicating values of the “good” DQ flags to indicating the “bad” DQ flags, prepend ‘~’ to the string value. For example, in order to mask only pixels that have corresponding DQ flags 4 and 8 and to consider as “good” all other pixels set `dqbitmask` to `~4+8`, or `~4, 8`. To obtain the same effect with an `int` (<https://docs.python.org/3/library/functions.html#int>) input value (except for 0), enter $-(4+8+1)=-9$. Following this convention, a `dqbitmask` string value of `'~0'` would be equivalent to setting `dqbitmask=None`.

imweight: `numpy.ndarray`, `None`, optional Pixel weight array corresponding to `imdata`.

data_units: `{‘counts’, ‘rate’}`, optional Indicates the type of data units: count-like or rate-like (counts per unit of time). This provides the information necessary for unit conversion when needed.

exptime: `float`, optional Exposure time of image `imdata`.

pad: `int`, optional Number of pixels to pad around the minimal rectangle enclosing a source segmentation.

Returns

segments [list of `Cutout`] A list of extracted `Cutout` s.

```
subpixal.cutout.create_cutouts(primary_cutouts, segmentation_image, drz_data,
                               drz_wcs, flt_data, flt_wcs, drz_dq=None,
                               drz_dqbitmask=0, drz_weight=None,
                               drz_data_units='rate', drz_exptime=1,
                               flt_dq=None, flt_dqbitmask=0, flt_weight=None,
                               flt_data_units='counts', flt_exptime=1, pad=2,
                               combine_seg_mask=True)
```

A function for mapping “primary cutouts” (cutouts formed from a drizzle-combined image) to “input images” (generally speaking, distorted images) and some other “drizzle-combined” image. This “other” drizzle-combined image may be the same image used to create primary cutouts.

This function performs the following mapping/cutout extractions:

```
>primary_cutouts->imcutouts->drz_cutouts
```

That is, this function takes as input `primary_cutouts` and finds equivalent cutouts in the “input” (distorted) “flt” image. Then it takes the newly found `imcutouts` cutouts and finds/extracts

equivalent cutouts in the “drz” (usually distortion-corrected) image. Fundamentally, this function first calls `create_input_image_cutouts` to create `imcutouts` and then it calls `drz_from_input_cutouts` to create `drz_cutouts`.

Parameters

primary_cutouts [list of `Cutout`] A list of `Cutout` s that need to be mapped to *another* image.

segmentation_image: numpy.ndarray A 2D segmentation image identifying sources from the catalog in `imdata`. This is used for creating boolean mask of bad (not within a segmentation region) pixels.

drz_data: numpy.ndarray Image data array of “drizzle-combined” image.

drz_wcs: astropy.wcs.WCS World coordinate system of “drizzle-combined” image.

flt_data: numpy.ndarray Image data array of “distorted” image (input to the `drizzle`).

flt_wcs: astropy.wcs.WCS World coordinate system of “distorted” image.

drz_dq: numpy.ndarray, None, optional Data quality array corresponding to `drz_data`.

drz_dqbitmask [int, str, None, optional] Integer sum of all the DQ bit values from the input `drz_dq` DQ array that should be considered “good” when building masks for cutouts. For more details, see [create_primary_cutouts](#).

drz_weight: numpy.ndarray, None, optional Pixel weight array corresponding to `drz_data`.

drz_data_units: {'counts', 'rate'}, optional Indicates the type of data units for the `drz_data`: count-like or rate-like (counts per unit of time). This provides the information necessary for unit conversion when needed.

drz_exptime: float, optional Exposure time of image `drz_data`.

flt_dq: numpy.ndarray, None, optional Data quality array corresponding to `flt_data`.

flt_dqbitmask [int, str, None, optional] Integer sum of all the DQ bit values from the input `flt_dq` DQ array that should be considered “good” when building masks for cutouts. For more details, see [create_primary_cutouts](#).

flt_weight: numpy.ndarray, None, optional Pixel weight array corresponding to `flt_data`.

flt_data_units: {'counts', 'rate'}, optional Indicates the type of data units for the `flt_data`: count-like or rate-like (counts per unit of time). This provides the information necessary for unit conversion when needed.

flt_exptime: float, optional Exposure time of image `flt_data`.

pad: int, optional Number of pixels to pad around the minimal rectangle enclosing a mapped cutout (a cutout to be extracted).

combine_seg_mask: bool, optional Indicates whether to combine segmentation mask with cutout's mask. When `True` (<https://docs.python.org/3/library/constants.html#True>), segmentation image is used to create a mask that indicates “good” pixels in the image. This mask is combined with cutout's mask.

Returns

flt_cutouts [list of Cutout] A list of `Cutout` s extracted from the `flt_data`. These cutouts are large enough to enclose cutouts from the input `primary_cutouts` when `pad=1` (to make sure even partial pixels are included).

drz_cutouts [list of Cutout] A list of extracted `Cutout` s from the `drz_data`. These cutouts are large enough to enclose cutouts from the `flt_cutouts` when `pad=1` (to make sure even partial pixels are included).

exception `subpixal.cutout.NoOverlapError`

Raised when cutout does not intersect the extraction image.

exception `subpixal.cutout.PartialOverlapError`

Raised when cutout only partially overlaps the extraction image.

1.5 Blot Algorithm for Cutouts

A module that provides blotting algorithm for image cutouts and a default WCS-based coordinate mapping class.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

class `subpixal.blot.BlotWCSMap` (*source_cutout*, *target_cutout*)

Coordinate mapping class that performs coordinate transformation from the source cutout to the “target” cutout. The target cutout simply provides a coordinate system. This class implements coordinate transformation in the `__call__()` method.

Parameters

source_cutout [Cutout] A cutout that defines source coordinate system (input to the `__call__(x, y)` method).

target_cutout [Cutout] A cutout that provides target coordinates system to which source coordinates need to be mapped.

`subpixal.blot.blot_cutout` (*source_cutout*, *target_cutout*, *interp='poly5'*, *sincsl=1.0*, *wcsmmap=None*)

Performs ‘blot’ operation to create a single blotted image from a single source image. All distortion information is assumed to be included in the WCS of the `source_cutout` and `target_cutout`.

Parameters

source_cutout [Cutout] Cutout that needs to be “blotted”. Provides source image for the “blot” operation and a WCS.

target_cutout [Cutout] Cutout to which `source_cutout` will be “blotted”. This cutout provides a WCS and an output grid.

interp [{‘nearest’, ‘linear’, ‘poly3’, ‘poly5’, ‘spline3’, ‘sinc’}, optional] Form of interpolation to use when blotting pixels.

sinscl [float, optional] Scale for sinc interpolation kernel (in output, blotted pixels)

wcsmap [callable, optional] Custom mapping class to use to provide transformation from source cutout image coordinates to target cutout image coordinates.

1.6 Image Cross-Correlation and Interlacing

A module that provides algorithm for creating sub-pixel cross-correlation images and computing displacements.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelphelp.stsci.edu) (<https://hsthelphelp.stsci.edu>))

License *LICENSE*

`subpixal.cc.find_displacement(ref_image, image00, image10, image01, image11, cc_type='NCC', full_output=False)`

Find subpixel displacements between one reference cutout and a set of four “dithered” cutouts. This is achieved by finding peak position in a “supersampled” cross-correlation image obtained by interlacing cross-correlation maps of reference cutout with each dithered cutout.

Parameters

ref_image [numpy.ndarray] Image of a reference cutout.

image00 [numpy.ndarray] Image whose displacement relative to reference image needs to be computed. It must have same shape as `ref_image`.

image10 [numpy.ndarray] “Same” image as `image00` but sampled at a 1/2 pixel displacement along the X-axis. It must have same shape as `ref_image`.

image01 [numpy.ndarray] “Same” image as `image00` but sampled at a 1/2 pixel displacement along the Y-axis. It must have same shape as `ref_image`.

image11 [numpy.ndarray] “Same” image as `image00` but sampled at a 1/2 pixel displacement along both X-axis and Y-axis. It must have same shape as `ref_image`.

cc_type [{‘CC’, ‘NCC’, ‘ZNCC’}, optional] Cross-correlation algorithm to be used. ‘CC’ indicates the “standard” cross-correlation algorithm. ‘NCC’ refers to the normalized cross-correlation and ‘ZNCC’ refers to the zero-normalized cross-correlation, see, e.g., [Terminology in image processing](https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing) (https://en.wikipedia.org/wiki/Cross-correlation#Terminology_in_image_processing).

full_output [bool, optional] Return displacements only (`full_output=False`) or also interlaced cross-correlation image and direct (non-interlaced) cross-correlation images

Returns

- dx** [float] Displacement of `image00` with regard to `ref_image` along the X-axis (columns).
- dy** [float] Displacement of `image00` with regard to `ref_image` along the Y-axis (rows).
- icc** [numpy.ndarray, Optional] Interlaced (“supersampled”) cross-correlation image. Returned only when `full_output` is `True` (<https://docs.python.org/3/library/constants.html#True>).
- ccs** [list of numpy.ndarray, Optional] List of cross-correlation images between `ref_image` and `image??`. Returned only when `full_output` is `True` (<https://docs.python.org/3/library/constants.html#True>).

1.7 Centroid Algorithm

Utilities for finding peak in an image.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

`subpixal.centroid.find_peak` (`image_data`, `xmax=None`, `ymax=None`, `peak_fit_box=5`,
`peak_search_box=None`, `mask=None`)

Find location of the peak in an array. This is done by fitting a second degree 2D polynomial to the data within a `peak_fit_box` and computing the location of its maximum. When `xmax` and `ymax` are both `None` (<https://docs.python.org/3/library/constants.html#None>), an initial estimate of the position of the maximum will be performed by searching for the location of the pixel/array element with the maximum value. This kind of initial brute-force search can be performed even when `xmax` and `ymax` are provided but when one suspects that these input coordinates may not be very accurate by specifying an expanded brute-force search box through parameter `peak_search_box`.

Parameters

image_data [numpy.ndarray] Image data.

xmax [float, None, optional] Initial guess of the x-coordinate of the peak. When both `xmax` and `ymax` are `None` (<https://docs.python.org/3/library/constants.html#None>), the initial (pre-fit) estimate of the location of the peak will be obtained by a brute-force search for the location of the maximum-value pixel in the *entire* `image_data` array, regardless of the value of `peak_search_box` parameter.

ymax [float, None, optional] Initial guess of the y-coordinate of the peak. When both `xmax` and `ymax` are `None` (<https://docs.python.org/3/library/constants.html#None>), the initial (pre-fit) estimate of the location of the peak will be obtained by a brute-force search for the location of the maximum-value pixel in the *entire* `image_data` array, regardless of the value of `peak_search_box` parameter.

peak_fit_box [int, tuple of int, optional] Size (in pixels) of the box around the input estimate of the maximum (given by `xmax` and `ymax`) to be used for quadratic fitting from which peak location is computed. If a single integer number is provided, then it is assumed that fitting box is a square with sides of length given by `peak_fit_box`. If a tuple of two values is provided, then first value indicates the width of the box and the second value indicates the height of the box.

peak_search_box [str {'all', 'off', 'fitbox'}, int, tuple of int, None, optional] Size (in pixels) of the box around the input estimate of the maximum (given by `xmax` and `ymax`) to be used for brute-force search of the maximum value pixel. This search is performed before quadratic fitting in order to improve the original estimate of the peak given by input `xmax` and `ymax`. If a single integer number is provided, then it is assumed that search box is a square with sides of length given by `peak_fit_box`. If a tuple of two values is provided, then first value indicates the width of the box and the second value indicates the height of the box. 'off' or `None` (<https://docs.python.org/3/library/constants.html#None>) turns off brute-force search of the maximum. When `peak_search_box` is 'all' then the entire `image_data` data array is searched for maximum and when it is set to 'fitbox' then the brute-force search is performed in the same box as `peak_fit_box`.

Note: This parameter is ignored when both `xmax` and `ymax` are `None` (<https://docs.python.org/3/library/constants.html#None>) since in that case the brute-force search for the maximum is performed in the entire input array.

mask [numpy.ndarray, optional] A boolean type `ndarray` (<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray>) indicating “good” pixels in image data (`True` (<https://docs.python.org/3/library/constants.html#True>)) and “bad” pixels (`False` (<https://docs.python.org/3/library/constants.html#False>)). If not provided all pixels in `image_data` will be used for fitting.

Returns

coord [tuple of float] A pair of coordinates of the peak.

1.8 Utilities used by subpical

This module provides utility functions for use by `subpical` module.

Author Mihai Cara (for help, contact [HST Help Desk](https://hsthelpp.stsci.edu) (<https://hsthelpp.stsci.edu>))

License *LICENSE*

`subpical.utils.parse_file_name(image_name)`
Parse image file names including possible extensions.

Parameters

image_name [str] An image file name and (optionally) extension specification, e.g.:
'j1234567q_flt.fits[1]', 'j1234568q_flt.fits[sci,2]', etc.

Returns

file_name [str] File name itself **without** extension specification.

ext [tuple, int, None] A tuple of two elements: *extension name* (a string) and *extension version* (an integer number), e.g., ('SCI', 2). Alternatively, an extension can be specified using an integer *extension number*. When no extension was specified, **ext** returns `None` (<https://docs.python.org/3/library/constants.html#None>).

Examples

```
>>> import subpixal
>>> subpixal.utils.parse_file_name('j1234568q_flt.fits[sci,2]')
('j1234568q_flt.fits', ('sci', 2))
```

`subpixal.utils.py2round(x)`

This function returns a rounded up value of the argument, similar to Python 2.

`subpixal.utils.get_ext_list(image, extname='SCI')`

Return a list of all extension versions of `extname` extensions. `image` can be either a file name or a `astropy.io.fits.HDUList` (<http://docs.astropy.org/en/stable/io/fits/api/hdulist.html#astropy.io.fits.HDUList>) object.

This function returns a list of fully qualified extensions: a list of tuples of the form ('extname', 'extver').

Examples

```
>>> get_ext_list('j9irwlrqq_flt.fits')
[('SCI', 1), ('SCI', 2)]
```

1.9 LICENSE

Copyright (C) 2018, Association of Universities for Research in Astronomy

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DEVELOPMENT NOTES

2.1 Release Notes

2.1.1 0.1.0 (30-September-2019)

- Added `combine_seg_mask` argument to `align.align_images()` and other functions that allows users to turn off combining segmentation mask with other DQ masks for the cutouts. Practical application of this option is to turn off zeroing of pixels that are outside of the segmentation mask in the blotted cutouts. [#44]
- Added support for zero-normalized cross-correlation (ZNCC) and normalized cross-correlation (NCC) algorithms. [#42]
- Allow alignment code to run with without cosmic ray-cleaned images. [#41]
- Reliability enhancement in handling cases when sky computation is turned off. [#40]
- Modified the formula for computing RMSE of the fit in *image pixels* to take into account weights when available. [#39]

2.1.2 0.0.5 (22-February-2019)

- Added support for weighted fitting. Added parameter '`use_weights`' that can be used to enable/disable weighted fitting when input catalogs have a column called '`weight`'. [#38]

2.1.3 0.0.4 (03-January-2019)

- Added support for keeping top/bottom number of sources according to values in a specified catalog's column. [#37]
- The direction of the displacement as well as the direction of the fit have been reversed (bug fix). [#36]
- Instead of reporting XRMS and YRMS (rms of the fit in the tangent plane; i.e., the RMS displacement of the image source positions wrt. reference source positions, now the code will report total RMS `FIT_RMS` computed as $\sqrt{XRMS^2 + YRMS^2}$ and `IMG_RMS` (equivalent of `FIT_RMS` but computed in input image pixels - hence the problem with this measure for images affected by distortion). [#36]

- Added a parameter (`wcsupdate`) that allows a choice of when to update image headers with an aligned WCS: as soon as an image is fit (and then it can be used by next images) or wait until the end of the iteration and update all images at once. [#36]

2.1.4 0.0.3 (27-December-2018)

- Make sure `execute()` is called before returning segmentation image data. [#32]
- Add missing import. [#32]
- Setup dependency clean-up. [#31]
- Fix changelog. [#30]

2.1.5 0.0.2 (23-December-2018)

- Initial fully operational release. [#29]

2.1.6 0.0.1 (10-April-2018)

- Initial release. [#1]

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

- `subpixal.align`, [3](#)
- `subpixal.blot`, [20](#)
- `subpixal.catalogs`, [6](#)
- `subpixal.cc`, [21](#)
- `subpixal.centroid`, [22](#)
- `subpixal.cutout`, [14](#)
- `subpixal.resample`, [12](#)
- `subpixal.utils`, [23](#)

A

`align_images()` (in module `subpical.align`), 3
`append_filters()` (`subpical.catalogs.ImageCatalog` method), 6
`append_filters()` (`subpical.catalogs.SExImageCatalog` method), 9

B

`blc()` (`subpical.cutout.Cutout` property), 15
`blot_cutout()` (in module `subpical.blot`), 20
`BlotWCSMap` (class in `subpical.blot`), 20

C

`catalog()` (`subpical.catalogs.ImageCatalog` method), 6
`catalog()` (`subpical.catalogs.SExImageCatalog` property), 9
`compute_position_std()` (`subpical.catalogs.ImageCatalog` method), 6
`compute_position_std()` (`subpical.catalogs.SExImageCatalog` method), 9
`compute_weights()` (`subpical.catalogs.ImageCatalog` method), 7
`compute_weights()` (`subpical.catalogs.SExImageCatalog` method), 9
`computed_sky()` (`subpical.resample.Resample` property), 12
`correct_wcs()` (in module `subpical.align`), 6
`create_cutouts()` (in module `subpical.cutout`), 18
`create_primary_cutouts()` (in module `subpical.cutout`), 17

`Cutout` (class in `subpical.cutout`), 14
`cutout_src_pos()` (`subpical.cutout.Cutout` property), 15

D

`data()` (`subpical.cutout.Cutout` property), 15
`data_units()` (`subpical.cutout.Cutout` property), 15
`DEFAULT_ACCURACY` (`subpical.cutout.Cutout` attribute), 15
`DEFAULT_MAXITER` (`subpical.cutout.Cutout` attribute), 15
`DEFAULT_QUIET` (`subpical.cutout.Cutout` attribute), 15
`dq()` (`subpical.cutout.Cutout` property), 15
`Drizzle` (class in `subpical.resample`), 13
`dx()` (`subpical.cutout.Cutout` property), 16
`dy()` (`subpical.cutout.Cutout` property), 16

E

`execute()` (`subpical.catalogs.ImageCatalog` method), 7
`execute()` (`subpical.catalogs.SExImageCatalog` method), 10
`execute()` (`subpical.resample.Drizzle` method), 13
`execute()` (`subpical.resample.Resample` method), 12
`exptime()` (`subpical.cutout.Cutout` property), 16
`extraction_slice()` (`subpical.cutout.Cutout` property), 16

F

`fast_add_image()` (`subpical.resample.Drizzle` method), 13
`fast_add_image()` (`subpical.resample.Resample` method), 12

`fast_drop_image()` (sub-
pixel.resample.Drizzle method), 13
`fast_drop_image()` (sub-
pixel.resample.Resample method), 12
`fast_replace_image()` (sub-
pixel.resample.Drizzle method), 13
`filters()` (subpical.catalogs.ImageCatalog
property), 7
`filters()` (subpical.catalogs.SExImageCatalog
property), 10
`find_displacement()` (in module sub-
pixel.cc), 21
`find_linear_fit()` (in module sub-
pixel.align), 5
`find_peak()` (in module subpical.centroid), 22

G

`get_bbox()` (subpical.cutout.Cutout *method*), 16
`get_ext_list()` (in module subpical.utils), 24
`get_segmentation_image()` (sub-
pixel.catalogs.ImageCatalog method),
 7
`get_segmentation_image()` (sub-
pixel.catalogs.SExImageCatalog method),
 10

H

`height()` (subpical.cutout.Cutout *property*), 16

I

`image_extn()` (subpical.catalogs.ImageCatalog
property), 7
`image_extn()` (sub-
pixel.catalogs.SExImageCatalog property),
 10
 ImageCatalog (class in subpical.catalogs), 6
`input_image_names()` (sub-
pixel.resample.Resample property),
 12
`insertion_slice()` (subpical.cutout.Cutout
property), 16

M

`mask()` (subpical.cutout.Cutout *property*), 16
`mask_type()` (subpical.catalogs.ImageCatalog
property), 7
`mask_type()` (sub-
pixel.catalogs.SExImageCatalog property),
 10

N

`naxis()` (subpical.cutout.Cutout *property*), 16
 NoOverlapError, 20

O

`output_circlean()` (sub-
pixel.resample.Resample property),
 12
`output_ctx()` (subpical.resample.Resample
property), 12
`output_sci()` (subpical.resample.Resample
property), 12
`output_wht()` (subpical.resample.Resample
property), 13

P

`parse_file_name()` (in module subpical.utils),
 23
 PartialOverlapError, 20
`pix2world()` (subpical.cutout.Cutout *method*),
 16
`pscale()` (subpical.cutout.Cutout *property*), 16
`py2round()` (in module subpical.utils), 24

R

`reference_image()` (sub-
pixel.resample.Drizzle property), 13
`reference_image()` (sub-
pixel.resample.Resample property),
 13
`remove_all_filters()` (sub-
pixel.catalogs.ImageCatalog method),
 7
`remove_all_filters()` (sub-
pixel.catalogs.SExImageCatalog method),
 10
`remove_filter()` (sub-
pixel.catalogs.ImageCatalog method),
 7
`remove_filter()` (sub-
pixel.catalogs.SExImageCatalog method),
 10
`required_colnames()` (sub-
pixel.catalogs.ImageCatalog property),
 7
`required_colnames()` (sub-
pixel.catalogs.SExImageCatalog property),
 10

Resample (*class in subpical.resample*), 12

S

set_config() (*subpical.resample.Drizzle method*), 13

set_config_parameters() (*subpical.resample.Drizzle method*), 13

set_config_parameters() (*subpical.resample.Resample method*), 13

set_default_filters() (*subpical.catalogs.ImageCatalog method*), 8

set_default_filters() (*subpical.catalogs.SExImageCatalog method*), 10

set_filters() (*subpical.catalogs.ImageCatalog method*), 8

set_filters() (*subpical.catalogs.SExImageCatalog method*), 10

set_image() (*subpical.catalogs.ImageCatalog method*), 8

set_image() (*subpical.catalogs.SExImageCatalog method*), 11

set_mask() (*subpical.catalogs.ImageCatalog method*), 8

set_mask() (*subpical.catalogs.SExImageCatalog method*), 11

sexconfig() (*subpical.catalogs.SExImageCatalog property*), 12

SExImageCatalog (*class in subpical.catalogs*), 9

src_id() (*subpical.cutout.Cutout property*), 16

src_pos() (*subpical.cutout.Cutout property*), 16

src_weight() (*subpical.cutout.Cutout property*), 16

subpical.align (*module*), 3

subpical.blot (*module*), 20

subpical.catalogs (*module*), 6

subpical.cc (*module*), 21

subpical.centroid (*module*), 22

subpical.cutout (*module*), 14

subpical.resample (*module*), 12

subpical.utils (*module*), 23

T

taskname (*subpical.resample.Drizzle attribute*), 14

trc() (*subpical.cutout.Cutout property*), 16

U

update_image_wcs() (*in module subpical.align*), 6

W

wcs() (*subpical.cutout.Cutout property*), 16

weight() (*subpical.cutout.Cutout property*), 17

width() (*subpical.cutout.Cutout property*), 17

world2pix() (*subpical.cutout.Cutout method*), 17