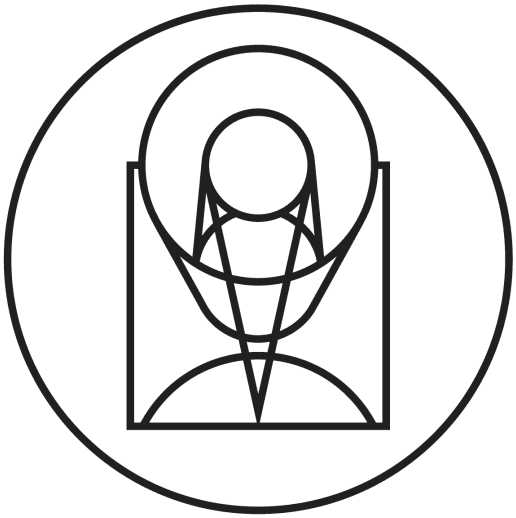

subpixal Documentation

Release 0.0.0.dev39+ga77505a7 (2018-10-04 03:22:33 -0400)

Mihai Cara

Oct 04, 2018

Contents

`subpixal` is a package that provides tools for **SUB-PIX**el 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.

CHAPTER 1

Content

1.1 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://hsthelp.stsci.edu))

License *LICENSE*

class subpixal.catalogs.**SourceCatalog**
A class for handling catalog data: storing, filtering, and retrieving sources.

append_filters (*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
Get catalog (after applying masks and selection filters).

catmap
Get raw catalog column name mapping.

execute()
Compute catalog applying masks and selecting only sources that satisfy all set filters.

filters
Get a list of all active selection filters.

is_dirty()
Returns the "dirty" status. When a catalog is marked as "dirty", sources must be (re-)extracted from the raw catalog. In order to update

mark_dirty()
Mark the catalog as "dirty", indicating whether or not sources should be re-extracted from the raw catalog when `execute()` is run. Masking and filtering criteria will be applied to the raw catalog during this run of `execute()`.

mask
Get mask indicating "bad" (`True` (https://docs.python.org/3/library/constants.html#True)) and "good" (`False` (https://docs.python.org/3/library/constants.html#False)) sources when `mask_type` is `'image'` or a 2D array of shape `(N, 2)` containing integer coordinates of "bad" pixels.

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

predefined_catmaps
Get names of available (pre-defined) column name mappings.

rawcat
Get raw catalog.

rawcat_colnames
Get a list of the column names in the raw catalog **after** catalog column name mapping has been applied.

remove_filter(*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.

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.

reset_filters()
Remove selection filters.

set_default_filters()
Set default source selection criteria.

set_filters(*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, comp, value)` 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.
- `comp` is a **string** representing a comparison operator. The following operators are suported: `['>', '>=', '==', '!=', '<', '<=']`.
- `value` is a numeric value to be used for comparison of column values.

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

set_raw_catalog(*rawcat*, *catmap=None*, *origin=0*)

Parameters

rawcat [astropy.table.Table] An `Table` (http://docs.astropy.org/en/stable/api/astropy.table.Table.html# containing source data.

catmap [dict, str, optional] A `dict` (https://docs.python.org/3/library/stdtypes.html#dict) that provides mapping (a dictionary) between source's `'x'`, `'y'`, `'flux'`, etc. and the corresponding column names in a `Table` (http://docs.astropy.org/en/stable/api/astropy.table.Table.html#astropy.table.Table) catalog. Instead of a dictionary, this parameter may be a pre-defined mapping name supported by this class. To get the list of all pre-defined catalog mapping supported by this class, use *`SourceCatalog`*'s *`predefined_catmaps`* property. When `catmap` is `None` (https://docs.python.org/3/library/constants.html#None), no column name mapping will be applied.

origin: int, float, optional Coordinates of the sources of *`SourceCatalog.catalog`* are zero-based. The `origin` is used for converting `rawcat`'s coordinates when raw catalog's source coordinates are not zero-based.

class subpixal.catalogs.**SExCatalog**(*rawcat=None*, *max_stellarity=1.0*)

A catalog class specialized for handling `SExtractor` output catalogs, such as being able to load raw `SExtractor` catalogs directly from text files.

Parameters

rawcat [astropy.table.Table, str] An `Table` (http://docs.astropy.org/en/stable/api/astropy.table.Table.html#a containing source data or a `SExtractor`-generated text file name.

max_stellarity [float, optional] Maximum stellarity for selecting sources from the catalog.

set_default_filters()

Sets default filters for selecting sources from the raw catalog.

Default selection criteria are: `flux > 0` AND `fwhm > 0` AND `radius > 0` AND `prof-rms-a > 0` AND `prof-rms-b > 0` AND `stellarity <= max_stellarity`.

set_raw_catalog(*rawcat*)

Set/replace raw catalog.

Parameters

rawcat [astropy.table.Table, str] An `Table` (http://docs.astropy.org/en/stable/api/astropy.table.Table.html#astropy.table.Table) containing source data or a `SExtractor`-generated text file name.

class subpixal.catalogs.**SExImageCatalog**(*image=None*, *sexconfig=None*, *max_stellarity=1.0*, *sextractor_cmd='sex'*)

A catalog class specialized for finding sources in images using `SExtractor` and then loading raw `SExtractor` catalogs directly from output text 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, optional] Maximum stellarity for selecting sources from the catalog.

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

execute()

Compute catalog applying masks and selecting only sources that satisfy all set filters.

image

Get image.

segmentation_file

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

sexconfig

Get `SExtractor` configuration file.

1.2 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://hsthelp.stsci.edu))

License *LICENSE*

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

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

execute()

Run resampling algorithm.

fast_add_image(*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(*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.

input_image_names
Get a list of input file names or `None` (https://docs.python.org/3/library/constants.html#None).

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

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

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

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

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

execute()
Run resampling algorithm.

fast_add_image(*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(*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(*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.

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

taskname = 'astrodrizzle'

1.3 Source Cutouts

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

Author Mihai Cara (for help, contact HST Help Desk (https://hsthelp.stsci.edu))

License *LICENSE*

class subpixal.cutout.**Cutout**(*data*, *wcs*, *blc=(0, 0)*, *trc=None*, *dq=None*, *weight=None*, *src_id=0*, *data_units='rate'*, *exptime=1*, *mode='strict'*, *fillval=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 convinience 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)`.

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

blc
Set/Get coordinate of the bottom-left corner.

data
Get image data.

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

dq
Set/Get cutout's data quality.

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

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

exptime
Get/Set exposure time.

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

get_bbox(*wrt='orig'*)
Get a `numpy.ndarray` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.nd
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]

height
Get width of the cutout.

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'`.

mask
Set/Get cutout's mask.

naxis
Get FITS `NAXIS` property of the cutout.

pix2world(**args*, *origin=0*)
Convert _cutout_'s pixel coordinates to world coordinates.

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

src_id
Set/Get source ID.

trc
Set/Get coordinate of the top-right corner.

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

weight
Set/Get cutout's pixel weight.

width
Get width of the cutout.

world2pix(**args*, *origin=0*)
Convert world coordinates to _cutout_'s pixel coordinates.

subpixal.cutout.**create_primary_cutouts**(*catalog*, *segmentation_image*, *imdata*, *imwcs*, *imdq=None*, *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 [SourceCatalog, 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`.

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 array corresponding to `imdata`.

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_weight=None*, *drz_data_units='rate'*, *drz_exptime=1*, *flt_dq=None*, *flt_weight=None*, *flt_data_units='counts'*, *flt_exptime=1*, *pad=2*)

A function for mapping "primary cutouts" (cutouts formed form 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_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_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).

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.

1.4 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://hsthelp.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'*, *sinscl=1.0*, *wcsmap=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.

1.5 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://hsthelp.stsci.edu))

License *LICENSE*

subpixal.cc.**find_displacement**(*ref_image*, *image00*, *image10*, *image01*, *image11*)

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`.

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).

1.6 Centroid Algorithm

Utilities for finding peak in an image.

Author Mihai Cara (for help, contact HST Help Desk (https://hsthelp.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 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.

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.

1.7 Utilities used by `subpixal`

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

Author Mihai Cara (for help, contact HST Help Desk (https://hsthelp.stsci.edu))

License *LICENSE*

`subpixal.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 extention 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.parse_file_name('j1234568q_flt.fits[sci,2]')
('SCI', 2)
```

`subpixal.utils.`**`py2round`**(*x*)

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

1.8 LICENSE

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.

CHAPTER 2

Development Notes

2.1 CHANGELOG

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index