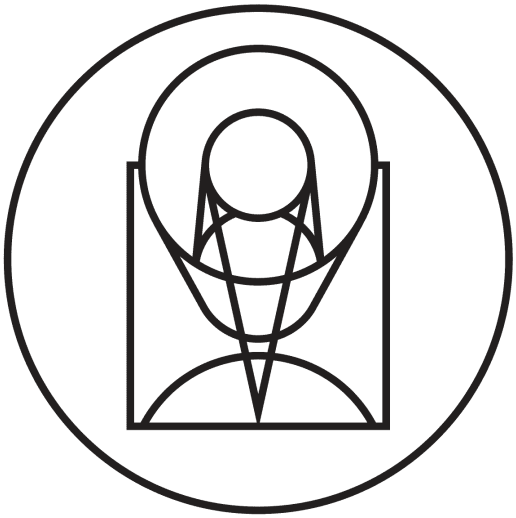

tweakwcs Documentation

Release 0.6.3

Mihai Cara

May 14, 2020

Contents

`tweakwcs` is a package that provides core algorithms for computing and applying corrections to `WCS` objects such as to minimize mismatch between image and reference catalogs. Currently only aligning images with `FITS WCS` and `JWST gWCS` are supported.

CHAPTER 1

Content

1.1 imalign

A module that provides functions for "aligning" images: specifically, it provides functions for computing corrections to image WCS so that image catalogs "align" to the reference catalog *on the sky*.

Authors Mihai Cara

License *LICENSE*

tweakwcs.imalign.**fit_wcs**(*refcat*, *imcat*, *tpwcs*, *fitgeom='general'*, *nclip=3*, *sigma=(3.0*, *'rmse')*)

"Tweak" **a single** image's WCS by fitting image catalog to a reference catalog. This is a simplified version of *align_wcs* that does not perform matching and is limited to the fitting part.

Note: Both reference and image catalogs must have been **matched** *prior to* calling fit_wcs(). This means that the lengths of both refcat and imcat catalogs must be equal *and* that coordinates with the same indices in both catalogs correspond to the same source.

Warning: If tpwcs.meta dictionary contains 'catalog' keyword, it will be ignored.

Parameters

refcat: astropy.table.Table A reference source catalog. The catalog must contain 'RA' and 'DEC' columns which indicate reference source world coordinates (in degrees). An optional column in the catalog is the 'weight' column, which when present, will be used in fitting. See Notes section for further details.

imcat: astropy.table.Table Source catalog associated with an image whose WCS needs to be aligned by fitting a linear transformation to `imcat` source positions so as to align them to the same sources from the `refcat` catalog. Must contain `'x'` and `'y'` columns which indicate source coordinates (in pixels) in the associated image. An optional column in the catalog is the `'weight'` column, which when present, will be used in fitting. See `Notes` section for further details.

tpwcs: TPWCS A `WCS` associated with the image from which the catalog was derived. This `TPWCS`-subclassed WCS corrector object must also define a tangent plane that will be used for fitting the two catalogs' sources and in which WCS corrections will be applied.

fitgeom: {'shift', 'rscale', 'general'}, optional 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. The 'general' fit geometry allows for independent scale and rotation for each axis.

nclip: int, None, optional Number (a non-negative integer) of clipping iterations in fit. Clipping will be turned off if `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

sigma: float, tuple of the form (float, str), optional When a tuple is provided, first value (a positive number) indicates the number of "fit error estimates" to use for clipping. The second value (a string) indicates the statistic to be used for "fit error estimate". Currently the following values are supported: `'rmse'`, `'mae'`, and `'std'` - see *`iter_linear_fit`* for more details.

When `sigma` is a single number, it must be a positive number and the default error estimate `'rmse'` is assumed.

This parameter is ignored when `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

Returns

twwcs: TPWCS "Tweaked" (aligned) `WCS` that contains tangent-plane corrections so that reference and image catalog sources better align in the tangent plane and therefore on the sky as well.

Notes

When fitting image sources to reference catalog sources, we can specify which sources have higher weights. This can be done by assigning a "weight" to each source by specifying these values in the optional `'weight'` column of either the reference catalog, image catalog, or both.

When weights are not provided, all sources are weighed equally. When only either image or reference catalog weights are provided, the sources will be weighted with the specified weights. When *both* image *and* reference catalogs specify weights for the same sources, the two weights will be combined into a single weight as:

$$1/w = 1/w_i + 1/w_r$$

Warning: Keep in mind that when a group catalog is created from individual catalogs, weights of the group catalog are created by *concatenating* weights of individual catalogs. Therefore, for the weighting of groups of catalogs to work correctly, the weights of individual catalogs should be scaled in such a way that when individual catalogs are combined into a single "group catalog", weights preserve their relative values.

For example, let's say a group is formed from two individual catalogs. Let's say first catalog contains four sources with equal weights `[1,1,1,1]` and the second catalog contains two sources with weights `[1,1]` then the group's catalogs sources will also have equal weights `[1,1,1,1,1,1]`. However, if each individual catalog's weights were normalized such that sum of all weights is 1, then group's sources will be weighed unequally: `[0.25,0.25,0.25,0.25,0.5,0.5]`.

Upon **successful** completion, this function will set the `'fit_info'` key value of the `meta` attribute of the returned `TPWCS` object. `'fit_info'` is a dictionary with the following items:

- **'shift'**: A `numpy.ndarray` with two components of the computed shift.
- **'matrix'**: A `2x2` `numpy.ndarray` with the computed generalized rotation matrix.
- **'proper_rot'**: Rotation angle (degree) as if the rotation is proper.
- **'rot'**: A tuple of `(rotx, roty)` - the rotation angles with regard to the `X` and `Y` axes.
- **'<rot>'**: *Arithmetic mean* of the angles of rotation around `X` and `Y` axes.
- **'scale'**: A tuple of `(sx, sy)` - scale change in the direction of the `X` and `Y` axes.
- **'<scale>'**: *Geometric mean* of scales `sx` and `sy`.
- **'skew'**: Computed skew.
- **'proper'**: a boolean indicating whether the rotation is proper.
- **'fitgeom'**: Fit geometry (allowed transformations) used for fitting data (to minimize residuals). This is copy of the input argument `fitgeom`.
- **'center'**: Center of rotation in the *tangent plane* of the computed linear transformations.
- **'fitmask'**: A boolean array indicating which source positions where used for fitting (`True` (https://docs.python.org/3/library/constants.html#True)) and which were clipped out (`False` (https://docs.python.org/3/library/constants.html#False)). **NOTE:** For weighted fits, positions with zero weights are automatically excluded from the fits.
- **'eff_nclip'**: Effective number of clipping iterations
- **'rmse'**: fit Root-Mean-Square Error in *tangent plane* coordinates of corrected image source positions from reference source positions.
- **'mae'**: fit Mean Absolute Error in *tangent plane* coordinates of corrected image source positions from reference source positions.
- **'std'**: Norm of the standard deviation of the residuals in *tangent plane* along each axis.
- **'resids'**: An array of residuals of the fit in the *tangent plane*.

NOTE: Only the residuals for the "valid" points are reported here. Therefore the length of this array may be smaller than the length of input arrays of positions.

- **'fit_RA'**: first (corrected) world coordinate of input source positions used in fitting.
- **'fit_DEC'**: second (corrected) world coordinate of input source positions used in fitting.
- **'status'**: Alignment status. Currently two possible status are possible `'SUCCESS'` or `'FAILED: reason for failure'`. When alignment failed, the reason for failure is provided after alignment status.

`tweakwcs.imalign.`**`align_wcs`**(*wcscat*, *refcat=None*, *enforce_user_order=True*, *expand_refcat=False*, *minobj=None*, *match=<tweakwcs.matchutils.TPMatch object at 0x7f96decad340>*, *fitgeom='general'*, *nclip=3*, *sigma=(3.0, 'rmse')*)

Align (groups of) image catalogs by adjusting the parameters of their WCS based on fits between matched sources in these catalogs and a reference catalog which may be automatically created from one of the input `wcscat` catalogs.

> **Warning:** This function modifies the `wcs` attribute of each item in the input `wcscat` list!

Upon completion, this function will add a field `'fit_info'` to the `meta` attribute of the input WCS correctors (except of the one chosen as a reference catalog when `refcat` is `None` (https://docs.python.org/3/library/constants.html#None)) containing a dictionary describing matching and fit results. For a description of the items in this dictionary, see *tweakwcs.wcsimage.WCSGroupCatalog.align_to_ref()*. In addition to the status set by *align_to_ref()*, this function may set status to `'REFERENCE'` for an input image used as a reference image when a reference catalog is not provided. In this case no other fields in the `'fit_info'` will be present because a reference image is not being aligned. When alignment failed, the reason for failure is provided after alignment status.

> **Warning:** Unless status in `'fit_info'` is `'SUCCESS'`, there is no guarantee that other fields in `'fit_info'` are present or valid. Therefore, it is advisable verify that status is `'SUCCESS'` before attempting to access other items, for example:
>
> ```
> >>> fit_info = wcscat[0].meta.get('fit_info')  # noqa
> >>> if fit_info['status'] == 'SUCCESS':
> ...     print("shifts: [{}, {}]".format(*fit_info['shift']))
> ... else:
> ...     print("tweak info not available for this image")
> ```

Parameters

wcscat: tweakwcs.tpwcs.TPWCS, list of tweakwcs.tpwcs.TPWCS
A list of all *TPWCS*-derived WCS correctors whose `meta` dictionary **must** contain `'catalog'` item with a non-empty table value of type `astropy.table.Table`

(https://docs.astropy.org/en/stable/api/astropy.table.Table.html#astropy.table.Table). This catalog must contain `'x'` and `'y'` columns which indicate source coordinates (in pixels) in the associated image. An optional column in the catalog is the `'weight'` column, which when present, will be used in fitting. See `Notes` section for further details. In addition to `'catalog'`, the following items in the `meta` dictionary are recognized/supported: `'name'` and `'group_id'`. `'name'` is catalog's name and it used to identify catalog during logging. If `'name'` value is `None` (https://docs.python.org/3/library/constants.html#None) or not present at all in the `meta` of a catalog, the name of that catalog will reported as `'Unknown'`. Group ID that may be used for identifying catalogs that need to be aligned together. `group_id` must be hashable. If `'group_id'` is `None` (https://docs.python.org/3/library/constants.html#None) or not provided, each input WCS/catalog will be aligned individually.

Note: Upon completion this function will add `'fit_info'` item (a dictionary) to input object's `meta` dictionary. See **Notes** section for more details.

> **Warning:** This function modifies the WCS of `TPWCS` objects by calling their *`set_correction()`* method.

refcat: astropy.table.Table, optional A reference source catalog. The catalog must contain `'RA'` and `'DEC'` columns which indicate reference source world coordinates (in degrees). An optional column in the catalog is the `'weight'` column, which when present, will be used in fitting. See `Notes` section for further details.

enforce_user_order: bool, optional Specifies whether images should be aligned in the order specified in the `file` input parameter or `align` should optimize the order of alignment by intersection area of the images. Default value (`True` (https://docs.python.org/3/library/constants.html#True)) will align images in the user specified order, except when some images cannot be aligned in which case `align` will optimize the image alignment order. Alignment order optimization is available *only* when `expand_refcat` is `True` (https://docs.python.org/3/library/constants.html#True).

expand_refcat: bool, optional Specifies whether to add new sources from just matched images to the reference catalog to allow next image to be matched against an expanded reference catalog. By delault, the reference catalog is not being expanded.

If `refcat` is not `None` (https://docs.python.org/3/library/constants.html#None) and contains an `'id'` column, then sources being added to the reference catalog will be assigned consecutive IDs that continue maximum ID in the `refcat`.

If one desires to uniquely associate source in the expanded catalog to their original catalogs, it is recommended that one assign unique IDs to all sources in all input catalogs **and** in the reference catalog in a separate column such as `'uuid'`.

minobj: int, None, optional Minimum number of identified objects from each input image to use in matching objects from other images. If the default `None` (https://docs.python.org/3/library/constants.html#None) value is used then `align` will automatically determmine the minimum number of sources from the value of the `fitgeom` parameter.

match: MatchCatalogs, function, None, optional A callable that takes two arguments: a reference catalog and an image catalog. Both catalogs will have columns `'TPx'` and `'TPy'` that represent the source coordinates in some common (to both catalogs) coordinate system.

fitgeom: {'shift', 'rscale', 'general'}, optional 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. The 'general' fit geometry allows for independent scale and rotation for each axis.

nclip: int, None, optional Number (a non-negative integer) of clipping iterations in fit. Clipping will be turned off if `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

sigma: float, tuple of the form (float, str), optional When a tuple is provided, first value (a positive number) indicates the number of "fit error estimates" to use for clipping. The second value (a string) indicates the statistic to be used for "fit error estimate". Currently the following values are supported: `'rmse'`, `'mae'`, and `'std'` - see *iter_linear_fit* for more details.

When `sigma` is a single number, it must be a positive number and the default error estimate `'rmse'` is assumed.

This parameter is ignored when `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

Returns

eff_refcat: astropy.table.Table Effective reference catalog used for aligning all images. Depending on the values of the input parameters `refcat`, `enforce_user_order`, and `expand_refcat`, effective reference catalog may be one of the input image catalogs, the original `refcat` catalog, an expanded `refcat` with a combination of source positions from all input images.

Notes

1. Weights:

When fitting image sources to reference catalog sources, we can specify which sources have higher weights. This can be done by assigning a "weight" to each source by specifying these values in the optional `'weight'` column of either the reference catalog, image catalog, or both.

When weights are not provided, all sources are weighed equally. When only either image or reference catalog weights are provided, the sources will be weighted with the specified weights. When *both* image *and* reference catalogs specify weights for the same sources, the two weights will be combined

into a single weight as:

$$1/w = 1/w_i + 1/w_r$$

Warning: Keep in mind that when a group catalog is created from individual catalogs, weights of the group catalog are created by *concatenating* weights of individual catalogs. Therefore, for the weighting of groups of catalogs to work correctly, the weights of individual catalogs should be scaled in such a way that when individual catalogs are combined into a single "group catalog", weights preserve their relative values.

For example, let's say a group is formed from two individual catalogs. Let's say first catalog contains four sources with equal weights `[1,1,1,1]` and the second catalog contains two sources with weights `[1,1]` then the group's catalogs sources will also have equal weights `[1,1,1,1,1,1]`. However, if each individual catalog's weights were normalized such that sum of all weights is 1, then group's sources will be weighed unequally: `[0.25,0.25,0.25,0.25,0.5,0.5]`.

Warning: When image catalogs contain optional `'weight'` column, then all image catalogs in a group must contain this column.

2. `'fit_info'`:

Upon completion, this function will add `'fit_info'` item (itself a dictionary) to input object's `meta` dictionary. If input objects are *TPWCS* WCS correctors, then `TPWCS.meta['fit_info']` will be set to a dictionary containing fit information.

Note: For *TPWCS* that are aligned in a group, the `'matrix'` and `'shift'` items in the `'fit_info'` dictionary may differ from the values of the same items in `TPWCS.meta` dictionary. This is normal since WCS corrections (in *TPWCS*) are applied in the image's WCS plane while fit may be performed in a slightly different tangent plane.

1.2 tpwcs

This module provides support for manipulating tangent-plane corrections of `WCS`.

Authors Mihai Cara

License *LICENSE*

class `tweakwcs.tpwcs.`**TPWCS**(*wcs*, *meta=None*)

A class that provides common interface for manipulating WCS information and for managing tangent-plane corrections.

Parameters

wcs: ''WCS object'' A `WCS` object supported by a child class.

meta: dict, None, optional Dictionary that will be merged to the object's `meta` fields.

bounding_box
Get the bounding box (if any) of the underlying image for which the original WCS is defined.

copy(*self*)
Returns a deep copy of this object.

det_to_tanp(*self*, *x*, *y*)
Convert detector (pixel) coordinates to tangent plane coordinates.

det_to_world(*self*, *x*, *y*)
Convert pixel coordinates to sky coordinates using full (i.e., including distortions) transformations.

meta

original_wcs
Get original WCS object.

set_correction(*self, matrix=[[1, 0], [0, 1]], shift=[0, 0], ref_tpwcs=None, meta=None, **kwargs*)
Sets a tangent-plane correction of the WCS object according to the provided liniar parameters. In addition, this function updates the `meta` attribute of the *TPWCS*-derived object with the values of keyword arguments except for the argument `meta` which is *merged* with the *attribute* `meta`.

Parameters

matrix: list, numpy.ndarray, optional A `2x2` array or list of lists coefficients representing scale, rotation, and/or skew transformations.

shift: list, numpy.ndarray, optional A list of two coordinate shifts to be applied to coordinates *after* `matrix` transformations are applied.

ref_tpwcs: TPWCS, None, optional A reference WCS of the type `TPWCS` that provides the tangent plane in which corrections (`matrix` and `shift`) were defined. When not provided (i.e., set to `None` (https://docs.python.org/3/library/constants.html#None)), it is assumed that the transformations are being applied directly to *this* image WCS' tangent plane.

meta: dict, None, optional Dictionary that will be merged to the object's `meta` fields.

****kwargs: optional parameters** Optional parameters for the WCS corrector.

tanp_center_pixel_scale
Estimate pixel scale in the tangent plane near a location in the detector's coordinate system corresponding to the origin of the tangent plane.

Returns

pscale: float Pixel scale [in units used in the tangent plane coordinate system] in the tangent plane near a location in the detector's plane corresponding to the origin of the tangent plane.

tanp_pixel_scale(*self*, *x*, *y*)
Estimate pixel scale in the tangent plane near a location in the detector's coordinate system given by parameters `x` and `y`.

Parameters

x: int, float X-coordinate of the pixel in the detector's coordinate system near which pixel scale in the tangent plane needs to be estimated.

y: int, float Y-coordinate of the pixel in the detector's coordinate system near which pixel scale in the tangent plane needs to be estimated.

Returns

pscale: float Pixel scale [in units used in the tangent plane coordinate system] in the tangent plane near a location specified by detector coordinates `x` and `y`.

tanp_to_det(*self*, *x*, *y*)
Convert tangent plane coordinates to detector (pixel) coordinates.

tanp_to_world(*self*, *x*, *y*)
Convert tangent plane coordinates to world coordinates.

wcs
Get current WCS object.

world_to_det(*self*, *ra*, *dec*)
Convert sky coordinates to image's pixel coordinates using full (i.e., including distortions) transformations.

world_to_tanp(*self*, *ra*, *dec*)
Convert tangent plane coordinates to detector (pixel) coordinates.

class tweakwcs.tpwcs.**JWSTgWCS**(*wcs*, *wcsinfo*, *meta=None*)
A class for holding `JWST gWCS` information and for managing tangent-plane corrections.

Parameters

wcs: GWCS A `GWCS` object.

wcsinfo: dict A dictionary containing reference angles of `JWST` instrument provided in the `wcsinfo` property of `JWST` meta data.

This dictionary **must contain** the following keys and values:

'v2_ref': float V2 position of the reference point in arc seconds.

'v3_ref': float V3 position of the reference point in arc seconds.

'roll_ref': float Roll angle in degrees.

meta: dict, None, optional Dictionary that will be merged to the object's `meta` fields.

bounding_box
Get the bounding box (if any) of the underlying image for which the original WCS is defined.

det_to_tanp(*self*, *x*, *y*)
Convert detector (pixel) coordinates to tangent plane coordinates.

det_to_world(*self*, *x*, *y*)
Convert pixel coordinates to sky coordinates using full (i.e., including distortions) transformations.

ref_angles
Return a `wcsinfo`-like dictionary of main WCS parameters.

set_correction(*self, matrix=[[1, 0], [0, 1]], shift=[0, 0], ref_tpwcs=None, meta=None, **kwargs*)
Sets a tangent-plane correction of the GWCS object according to the provided liniar parameters. In addition, this function updates the `meta` attribute of the *JWSTgWCS* object with the values of keyword arguments except for the argument `meta` which is *merged* with the *attribute* `meta`.

Parameters

matrix: list, numpy.ndarray A `2x2` array or list of lists coefficients representing scale, rotation, and/or skew transformations.

shift: list, numpy.ndarray A list of two coordinate shifts to be applied to coordinates *after* `matrix` transformations are applied.

ref_tpwcs: TPWCS, None, optional A reference WCS of the type `TPWCS` that provides the tangent plane in which corrections (`matrix` and `shift`) were defined. When not provided (i.e., set to `None` (https://docs.python.org/3/library/constants.html#None)), it is assumed that the transformations are being applied directly to *this* image WCS' tangent plane.

meta: dict, None, optional Dictionary that will be merged to the object's `meta` fields.

****kwargs: optional parameters** Optional parameters for the WCS corrector. *JWSTgWCS* ignores these arguments (except for storing them in the `meta` attribute).

tanp_to_det(*self*, *x*, *y*)
Convert tangent plane coordinates to detector (pixel) coordinates.

tanp_to_world(*self*, *x*, *y*)
Convert tangent plane coordinates to world coordinates.

world_to_det(*self*, *ra*, *dec*)
Convert sky coordinates to image's pixel coordinates using full (i.e., including distortions) transformations.

world_to_tanp(*self*, *ra*, *dec*)
Convert tangent plane coordinates to detector (pixel) coordinates.

class tweakwcs.tpwcs.**FITSWCS**(*wcs*, *meta=None*)
A class for holding `FITS WCS` information and for managing tangent-plane corrections.

Note: Currently only WCS objects that have `CPDIS`, `DET2IM`, and `SIP` distortions *before* the

application of the CD or PC matrix are supported.

Parameters

wcs: astropy.wcs.WCS An astropy.wcs.WCS (https://docs.astropy.org/en/stable/api/astropy.wcs.WCS.html#astropy.wcs.WCS) object.

bounding_box
Get the bounding box (if any) of the underlying image for which the original WCS is defined.

det_to_tanp(*self*, *x*, *y*)
Convert detector (pixel) coordinates to tangent plane coordinates.

det_to_world(*self*, *x*, *y*)
Convert pixel coordinates to sky coordinates using full (i.e., including distortions) transformations.

set_correction(*self, matrix=[[1, 0], [0, 1]], shift=[0, 0], ref_tpwcs=None, meta=None, **kwargs*)
Computes a corrected (aligned) wcs based on the provided linear transformation. In addition, this function updates the meta attribute of the *FITSWCS* object with the the values of keyword arguments except for the argument meta which is *merged* with the *attribute* meta.

Parameters

matrix: list, numpy.ndarray A 2x2 array or list of lists coefficients representing scale, rotation, and/or skew transformations.

shift: list, numpy.ndarray A list of two coordinate shifts to be applied to coordinates *after* matrix transformations are applied.

ref_tpwcs: TPWCS, None, optional A reference WCS of the type TPWCS that provides the tangent plane in which corrections (matrix and shift) were defined. When not provided (i.e., set to None (https://docs.python.org/3/library/constants.html#None)), it is assumed that the transformations are being applied directly to *this* image WCS' tangent plane.

meta: dict, None, optional Dictionary that will be merged to the object's meta fields.

****kwargs: optional parameters** Optional parameters for the WCS corrector. *FITSWCS* ignores these arguments (except for storing them in the meta attribute).

tanp_to_det(*self*, *x*, *y*)
Convert tangent plane coordinates to detector (pixel) coordinates.

tanp_to_world(*self*, *x*, *y*)
Convert tangent plane coordinates to world coordinates.

world_to_det(*self*, *ra*, *dec*)
Convert sky coordinates to image's pixel coordinates using full (i.e., including distortions) transformations.

world_to_tanp(*self*, *ra*, *dec*)
Convert tangent plane coordinates to detector (pixel) coordinates.

1.3 matchutils

A module that provides algorithms matching catalogs and for initial estimation of shifts based on 2D histograms.

License *LICENSE*

class tweakwcs.matchutils.**MatchCatalogs**
A class that provides common interface for matching catalogs.

class tweakwcs.matchutils.**TPMatch**(*searchrad=3.0*, *separation=0.5*, *use2dhist=True*, *xoffset=0.0*, *yoffset=0.0*, *tolerance=1.0*)
Catalog source matching in tangent plane. Uses `xyxymatch` algorithm to cross-match sources between this catalog and a reference catalog.

Note: The tangent plane is a plane tangent to the celestial sphere and it must be not distorted, that is, if *image* coordinates are distorted, then distortion correction must be applied to them before tangent plane coordinates are computed. Alternatively, one can think that undistorted world coordinates are projected from the sphere onto the tangent plane.

Parameters

searchrad: float, optional The search radius for a match (in units of the tangent plane).

separation: float, optional The minimum separation in the tangent plane (in units of the tangent plane) for sources in the image and reference catalogs in order to be considered to be disctinct sources. Objects closer together than `separation` distance are removed from the image and reference coordinate catalogs prior to matching. This parameter gets passed directly to `xyxymatch()` for use in matching the object lists from each image with the reference catalog's object list.

use2dhist: bool, optional Use 2D histogram to find initial offset?

xoffset: float, optional Initial estimate for the offset in X (in units of the tangent plane) between the sources in the image and the reference catalogs. This offset will be used for all input images provided. This parameter is ignored when `use2dhist` is `True` (https://docs.python.org/3/library/constants.html#True).

yoffset: float, optional Initial estimate for the offset in Y (in units of the tangent plane) between the sources in the image and the reference catalogs. This off-

set will be used for all input images provided. This parameter is ignored when `use2dhist` is `True` (https://docs.python.org/3/library/constants.html#True).

tolerance: float, optional The matching tolerance (in units of the tangent plane) 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.

1.4 wcsimage

This module provides support for working with image footprints on the sky and source catalogs.

Authors Mihai Cara

License *LICENSE*

`tweakwcs.wcsimage.`**`convex_hull`**(*x*, *y*, *wcs=None*)

Computes the convex hull of a set of 2D points.

Implements Andrew's monotone chain algorithm (http://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/
The algorithm has O(n log n) complexity.

Credit: http://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain

Parameters

points: list of tuples An iterable sequence of (x, y) pairs representing the points.

Returns

Output: list A list of vertices of the convex hull in counter-clockwise order, starting from the vertex with the lexicographically smallest coordinates.

class `tweakwcs.wcsimage.`**`RefCatalog`**(*catalog*, *name=None*, *footprint_tol=1.0*)

An object that holds a reference catalog and provides tools for coordinate convertions using reference WCS as well as catalog manipulation and expansion.

Parameters

catalog: astropy.table.Table Reference catalog.

Note: Reference catalogs (`Table` (https://docs.astropy.org/en/stable/api/astropy.table.Table.html#astro *must* contain *both* `'RA'` and `'DEC'` columns.

---

name: str, None, optional Name of the reference catalog.

footprint_tol: float, optional Matching tolerance in arcsec. This is used to estimate catalog's footprint when catalog contains only one or two sources.

`calc_bounding_polygon`(*self*)

Calculate bounding polygon of the sources in the catalog.

calc_tanp_xy (*self*, *tanplane_wcs*)
Compute x- and y-positions of the sources from the reference catalog in the tangent plane provided by the `tanplane_wcs`. This creates the following columns in the catalog's table: `'TPx'` and `'TPy'`.

Parameters

tanplane_wcs: ImageGWCS A `ImageGWCS` object that will provide transformations to the tangent plane to which sources of this catalog a should be "projected".

catalog
Get/set image's catalog.

expand_catalog (*self*, *catalog*)
Expand current reference catalog with sources from another catalog.

If current catalog is empty, then the catalog being added will become the new reference catalog. In this case if the `catalog` does have `id` column, those ID values will be preserved. If the `catalog` does not contain an ID column, then the new IDs will be assigned in increasing order starting with 1.

If the existing reference catalog is not empty, then the IDs from the `catalog` being added will be discarded and new IDs will be assigned in the increasing order such as to continue the numbering of existing source positions in the reference catalog.

Parameters

catalog: astropy.table.Table A catalog of new sources to be added to the existing reference catalog. *`catalog` must* contain *both* `'RA'` and `'DEC'` columns.

intersection (*self*, *wcsim*)
Compute intersection of this *`WCSImageCatalog`* object and another *`WCSImageCatalog`*, *`WCSGroupCatalog`*, *`RefCatalog`*, or `SphericalPolygon` object.

Parameters

wcsim: WCSImageCatalog, WCSGroupCatalog, RefCatalog, SphericalPolygon Another object that should be intersected with this *`WCSImageCatalog`*.

Returns

polygon: SphericalPolygon A `SphericalPolygon` that is the intersection of this *`WCSImageCatalog`* and `wcsim`.

intersection_area (*self*, *wcsim*)
Calculate the area of the intersection polygon.

name
Get/set *`WCSImageCatalog`* object's name.

poly_area
Area of the bounding polygon (in srad).

polygon
Get image's (or catalog's) bounding spherical polygon.

class tweakwcs.wcsimage.**WCSImageCatalog**(*catalog*, *tpwcs*, *name=None*, *group_id=None*, *meta={}*)

A class that holds information pertinent to an image WCS and a source catalog of the sources found in that image.

Warning: If `tpwcs.meta` dictionary contains any of the following keywords `'catalog'`, `'name'`, or `'group_id'`, they will be ignored without warning.

Parameters

catalog: astropy.table.Table Source catalog associated with an image. Must contain `'x'` and `'y'` columns which indicate source coordinates (in pixels) in the associated image.

tpwcs: TPWCS `TPWCS`-derived tangent-plane WCS corrector object associated with the image from which the catalog was derived.

name: str, None, optional Image catalog's name. This is used to identify catalog during logging. If `name` is `None` (https://docs.python.org/3/library/constants.html#None), the `name` of this `WCSImageCatalog` object will be set to `'Unknown'`.

group_id: hashable, None, optional Group ID that may be used for identifying catalogs that need to be aligned together. `group_id` must be hashable.

meta: dict, optional Additional information about image, catalog, and/or WCS to be stored (for convenience) within *WCSImageCatalog* object.

bb_radec

Get a 2xN `numpy.ndarray` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#nump of RA and DEC of the vertices of the bounding polygon.

calc_bounding_polygon(*self*)

Calculate bounding polygon of the image or of the sources in the catalog (if catalog was set).

catalog

Get/set image's catalog.

det_to_tanp(*self*, *x*, *y*)

Convert detector (pixel) coordinates to tangent plane coordinates.

det_to_world(*self*, *x*, *y*)

Convert pixel coordinates to sky coordinates using full (i.e., including distortions) transformations.

fit_info

Get fit information - a dictionary. This class sets only the `'status'` field but fitting routines may set additional fields.

fit_status

Get/Set fit status. This property is a shortcut to the `'status'` key value in the `fit_info`

dictionary. When the *WCSImageCatalog* object is created, `fit_status` is initially set to `'SKIPPED'`. Alignment tools are reponsible for updating catalog's fit status.

group_id
Get/set *WCSImageCatalog* object's group ID that may be used for identifying catalogs that need to be aligned together. `group_id` must be hashable.

intersection(*self*, *wcsim*)
Compute intersection of this *WCSImageCatalog* object and another *WCSImageCatalog*, *WCSGroupCatalog*, or `SphericalPolygon` object.

Parameters

wcsim: WCSImageCatalog, WCSGroupCatalog, SphericalPolygon Another object that should be intersected with this *WCSImageCatalog*.

Returns

polygon: SphericalPolygon A `SphericalPolygon` that is the intersection of this *WCSImageCatalog* and `wcsim`.

intersection_area(*self*, *wcsim*)
Calculate the area of the intersection polygon.

name
Get/set catalog's name. This is used to identify catalog during logging. Upon setting, the value will be converted to a `str` (https://docs.python.org/3/library/stdtypes.html#str). When setting to `None` (https://docs.python.org/3/library/constants.html#None), the `name` will be set to `'Unknown'`.

polygon
Get image's (or catalog's) bounding spherical polygon.

tanp_to_det(*self*, *x*, *y*)
Convert tangent plane coordinates to detector (pixel) coordinates.

tanp_to_world(*self*, *x*, *y*)
Convert tangent plane coordinates to world coordinates.

tpwcs
Get `TPWCS` WCS.

world_to_det(*self*, *ra*, *dec*)
Convert sky coordinates to image's pixel coordinates using full (i.e., including distortions) transformations.

world_to_tanp(*self*, *ra*, *dec*)
Convert tangent plane coordinates to detector (pixel) coordinates.

class tweakwcs.wcsimage.**WCSGroupCatalog**(*images*, *name=None*)
A class that holds together *WCSImageCatalog* image catalog objects whose relative positions are fixed and whose source catalogs should be fitted together to a reference catalog.

Parameters

images: list of WCSImageCatalog A list of *WCSImageCatalog* image catalogs.

name: str, None, optional Name of the group.

align_to_ref(*self*, *refcat*, *match=None*, *minobj=None*, *fitgeom='rscale'*, *nclip=3*, *sigma=(3.0, 'rmse')*)

Matches sources from the image catalog to the sources in the reference catalog, finds the affine transformation between matched sources, and adjusts images' WCS according to this fit.

Upon successful return, this function will also set the following fields of the `fit_info` attribute of each member *`WCSImageCatalog`* object:

- **'fitgeom'**: the value of the `fitgeom` argument
- **'eff_minobj'**: effective value of the `minobj` parameter
- **'matrix'**: computed rotation matrix
- **'shift'**: shift (offset) along X- and Y-axis
- **'rot'**: A tuple of `(rotx, roty)` - the rotation angles with regard to the `X` and `Y` axes.
- **'<rot>'**: *Arithmetic mean* of the angles of rotation around `X` and `Y` axes.
- **'proper_rot'**: rotation angle as if rotation is a proper rotation.
- **'proper'**: Indicates whether the rotation is a proper rotation (boolean)
- **'scale'**: A tuple of `(sx, sy)` - scale change in the direction of the `X` and `Y` axes.
- **'<scale>'**: *Geometric mean* of scales `sx` and `sy`.
- **'skew'**: Computed skew - an angle in the range [-180, 180).
- **'center'**: Center of rotation in the *tangent plane* of the computed linear transformations.
- **'fitmask'**: boolean array indicating (with `True` (https://docs.python.org/3/library/constants.html#True)) sources **used** for fitting
- **'nmatches'** [when `match` is not `None` (https://docs.python.org/3/library/constants.html#None)]: number of matched sources
- **'matched_ref_idx'** [when `match` is not `None` (https://docs.python.org/3/library/constants.html#None)]: indices of the matched sources in the reference catalog
- **'matched_input_idx'** [when `match` is not `None` (https://docs.python.org/3/library/constants.html#None)]: indices of the matched sources in the "input" catalog (the catalog from image to be aligned)
- **'fit_ref_idx'**: indices of the sources from the reference catalog used for fitting (a subset of 'matched_ref_idx' indices, when `match` is not `None` (https://docs.python.org/3/library/constants.html#None), left after clipping iterations performed during fitting)
- **'fit_input_idx'**: indices of the sources from the "input" (image) catalog used for fitting (a subset of 'matched_input_idx' indices, when `match` is not `None` (https://docs.python.org/3/library/constants.html#None), left after clipping iterations performed during fitting)
- **'rmse'**: fit Root-Mean-Square Error in *tangent plane* coordinates of corrected image source positions from reference source positions.

- **'mae'**: fit Mean Absolute Error in *tangent plane* coordinates of corrected image source positions from reference source positions.
- **'std'**: Norm of the STandard Deviation of the residuals in *tangent plane* along each axis.
- **'fit_RA'**: first (corrected) world coordinate of input source positions used in fitting.
- **'fit_DEC'**: second (corrected) world coordinate of input source positions used in fitting.
- **'status'**: Alignment status. Currently two possible status are possible `'SUCCESS'` or `'FAILED: reason for failure'`. When alignment failed, the reason for failure is provided after alignment status.

Note: A `'SUCCESS'` status does not indicate a "good" alignment. It simply indicates that alignment algortithm has completed without errors. Use other fields to evaluate alignment: fit `RMSE` and `MAE` values, number of matched sources, etc.

Parameters

refcat: RefCatalog A *`RefCatalog`* object that contains a catalog of reference sources as well as a valid reference WCS.

match: MatchCatalogs, function, None, optional A callable that takes two arguments: a reference catalog and an image catalog.

minobj: int, None, optional Minimum number of identified objects from each input image to use in matching objects from other images. If the default `None` (https://docs.python.org/3/library/constants.html#None) value is used then `align` will automatically determnine the minimum number of sources from the value of the `fitgeom` parameter. This parameter is used to interrupt alignment process (catalog fitting, `WCS` "tweaking") when the number of matched sources is smaller than the value of `minobj` in which case this method will return `False` (https://docs.python.org/3/library/constants.html#False).

fitgeom: {'shift', 'rscale', 'general'}, optional 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. The 'general' fit geometry allows for independent scale and rotation for each axis. This parameter is ignored if `match` is `False` (https://docs.python.org/3/library/constants.html#False).

nclip: int, None, optional Number (a non-negative integer) of clipping iterations in fit. Clipping will be turned off if `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

This parameter is ignored if `match` is `False` (https://docs.python.org/3/library/constants.html#False).

sigma: float, tuple of the form (float, str), optional When a tuple is provided, first value (a positive number) indicates the number of "fit error estimates" to use for clipping. The second value (a string) indicates the statistic to be used

for "fit error estimate". Currently the following values are supported: `'rmse'`, `'mae'`, and `'std'` - see *iter_linear_fit* for more details.

When `sigma` is a single number, it must be a positive number and the default error estimate `'rmse'` is assumed.

This parameter is ignored when `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0 or when `match` is `False` (https://docs.python.org/3/library/constants.html#False).

Returns

bool Returns `True` (https://docs.python.org/3/library/constants.html#True) if the number of matched sources is larger or equal to `minobj` and all steps have been performed, including catalog fitting and `WCS` "tweaking". If the number of matched sources is smaller than `minobj`, this function will return `False` (https://docs.python.org/3/library/constants.html#False).

apply_affine_to_wcs (*self*, *tanplane_wcs*, *matrix*, *shift*, *meta=None*)
Applies a general affine transformation to the WCS.

calc_tanp_xy (*self*, *tanplane_wcs*)
Compute x- and y-positions of the sources from the image catalog in the tangent plane. This creates the following columns in the catalog's table: `'TPx'` and `'TPy'`.

Parameters

tanplane_wcs: ImageGWCS A `ImageGWCS` object that will provide transformations to the tangent plane to which sources of this catalog a should be "projected".

catalog
Get/set image's catalog.

create_group_catalog (*self*)
Combine member's image catalogs into a single group's catalog.

Returns

group_catalog: astropy.table.Table Combined group catalog.

fit2ref (*self*, *refcat*, *tanplane_wcs*, *fitgeom='general'*, *nclip=3*, *sigma=(3.0, 'rmse')*)
Perform linear fit of this group's combined catalog to the reference catalog. When either/both group's catalog or/and the reference catalog contain `'weight'` column, weigted fitting will be performed. See `Notes` section for further details.

Parameters

refcat: RefCatalog A *RefCatalog* object that contains a catalog of reference sources.

tanplane_wcs: ImageGWCS A `ImageGWCS` object that will provide transformations to the tangent plane to which sources of this catalog a should be "projected".

fitgeom: {'shift', 'rscale', 'general'}, optional 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. The 'general' fit geometry allows for independent scale and rotation for each axis.

nclip: int, None, optional Number (a non-negative integer) of clipping iterations in fit. Clipping will be turned off if `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

sigma: float, tuple of the form (float, str), optional When a tuple is provided, first value (a positive number) indicates the number of "fit error estimates" to use for clipping. The second value (a string) indicates the statistic to be used for "fit error estimate". Currently the following values are supported: `'rmse'`, `'mae'`, and `'std'` - see *iter_linear_fit* for more details.

When `sigma` is a single number, it must be a positive number and the default error estimate `'rmse'` is assumed.

This parameter is ignored when `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

Notes

When fitting image sources to reference catalog sources, we can specify which sources have higher weights. This can be done by assigning a "weight" to each source by specifying these values in the optional `'weight'` column of either the reference catalog, image catalog, or both.

When weights are not provided, all sources are weighed equally. When only one of image or reference catalog weights are provided, the sources will be weighted with the specified weights. When *both* image *and* reference catalogs specify weights for the same sources, the two weights will be combined into a single weight as:

$$1/w = 1/w_i + 1/w_r$$

Warning: Keep in mind that when a group catalog is created from individual catalogs, weights of the group catalog are created by *concatenating* weights of individual catalogs. Therefore, for the weighting of groups of catalogs to work correctly, the weights of individual catalogs should be scaled in such a way that when individual catalogs are combined into a single "group catalog", weights preserve their relative values.

For example, let's say a group is formed from two individual catalogs. Let's say first catalog contains four sources with equal weights `[1,1,1,1]` and the second catalog contains two sources with weights `[1,1]` then the group's catalogs sources will also have equal weights `[1,1,1,1,1,1]`. However, if each individual catalog's weights were normalized such that sum of all weights is 1, then group's sources will be weighed unequally: `[0.25,0.25,0.25,0.25,0.5,0.5]`.

get_matched_cat (*self*)
Retrieve only those sources from the catalog that **have been** matched to the sources in the reference catalog.

get_unmatched_cat (*self*)
Retrieve only those sources from the catalog that have **not** been matched to the sources in the reference catalog.

intersection (*self*, *wcsim*)
Compute intersection of this *WCSGroupCatalog* object and another *WCSImageCatalog*, *WCSGroupCatalog*, or `SphericalPolygon` object.

Parameters

wcsim: WCSImageCatalog, WCSGroupCatalog, SphericalPolygon Another object that should be intersected with this *WCSGroupCatalog*.

Returns

polygon: SphericalPolygon A `SphericalPolygon` that is the intersection of this *WCSGroupCatalog* and `wcsim`.

intersection_area (*self*, *wcsim*)
Calculate the area of the intersection polygon.

match2ref (*self*, *refcat*, *match=None*)

Uses xyxymatch to cross-match sources between this catalog and a reference catalog.

Parameters

refcat: RefCatalog A *RefCatalog* object that contains a catalog of reference sources as well as a valid reference WCS.

match: MatchCatalogs, function, None, optional A callable that takes two arguments: a reference catalog and an image catalog. Both catalogs will have columns `'TPx'` and `'TPy'` that represent the source coordinates in some common (to both catalogs) coordinate system.

name
Get/set *WCSImageCatalog* object's name.

polygon
Get image's (or catalog's) bounding spherical polygon.

recalc_catalog_radec (*self*)
Recalculate RA and DEC of the sources in the catalog.

update_bounding_polygon (*self*)
Recompute bounding polygons of the member images.

1.5 linearfit

A module that provides algorithms for performing linear fit between sets of 2D points.

Authors Mihai Cara, Warren Hack

License *LICENSE*

`tweakwcs.linearfit.`**`iter_linear_fit`**(*xy*, *uv*, *wxy=None*, *wuv=None*, *fitgeom='general'*, *center=None*, *nclip=3*, *sigma=(3.0, 'rmse')*, *clip_accum=False*)

Compute linear transformation parameters that "best" (in the sense of minimizing residuals) transform `uv` source position to `xy` sources iteratively using sigma-clipping.

More precisely, this functions attempts to find a `2x2` matrix `F` and a shift vector `s` that minimize the residuals between the *transformed* reference source coordinates `uv`

$$\mathbf{xy}'_k = \mathbf{F} \cdot (\mathbf{uv}_k - \mathbf{c}) + \mathbf{s} + \mathbf{c} \tag{1.1}$$

and the "observed" source positions `xy`:

$$\epsilon^2 = \Sigma_k w_k \|\mathbf{xy}_k - \mathbf{xy}'_k\|^2. \tag{1.2}$$

In the above equations, $\mathbf{F}$ is a `2x2` matrix while $\mathbf{xy}_k$ and $\mathbf{uv}_k$ are the position coordinates of the `k`-th source (row in input `xy` and `uv` arrays).

One of the two catalogs (`xy` or `uv`) contains what we refer to as "image" source positions and the other one as "reference" source positions. The meaning assigned to `xy` and `uv` parameters are up to the caller of this function.

Parameters

xy: numpy.ndarray A `(N, 2)`-shaped array of source positions (one 2-coordinate position per line).

uv: numpy.ndarray A `(N, 2)`-shaped array of source positions (one 2-coordinate position per line). This array *must have* the same length (shape) as the `xy` array.

wxy: numpy.ndarray, None, optional A 1-dimensional array of weights of the same length (`N`) as `xy` array indicating how much a given coordinate should be weighted in the fit. If not provided or set to `None` (https://docs.python.org/3/library/constants.html#None), all positions will be contribute equally to the fit if `wuv` is also set to `None` (https://docs.python.org/3/library/constants.html#None). See `Notes` section for more details.

wuv: numpy.ndarray, None, optional A 1-dimensional array of weights of the same length (`N`) as `xy` array indicating how much a given coordinate should be weighted in the fit. If not provided or set to `None` (https://docs.python.org/3/library/constants.html#None), all positions will be contribute equally to the fit if `wxy` is also set to `None` (https://docs.python.org/3/library/constants.html#None). See `Notes` section for more details.

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

center: tuple, list, numpy.ndarray, None, optional A list-like container with two `X`- and `Y`-positions of the center (origin) of rotations in the `uv` and `xy` coordinate frames. If not provided, `center` is estimated as a (weighted) mean position in the `uv` frame.

nclip: int, None, optional Number (a non-negative integer) of clipping iterations in fit. Clipping will be turned off if `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

sigma: float, tuple of the form (float, str), optional When a tuple is provided, first value (a positive number) indicates the number of "fit error estimates" to use for clipping. The second value (a string) indicates the statistic to be used for "fit error estimate". Currently the following values are supported: `'rmse'`, `'mae'`, and `'std'` - see `Notes` section for more details.

When `sigma` is a single number, it must be a positive number and the default error estimate `'rmse'` is assumed.

This parameter is ignored when `nclip` is either `None` (https://docs.python.org/3/library/constants.html#None) or 0.

clip_accum: bool, optional Indicates whether or not to reset the list of "bad" (clipped out) sources after each clipping iteration. When set to `True` (https://docs.python.org/3/library/constants.html#True) the list only grows with each iteration as "bad" positions never re-enter the pool of available position for the fit. By default the list of "bad" source positions is purged at each iteration.

Returns

fit: dict

- `'shift'`: A `numpy.ndarray` with two components of the computed shift.
- `'shift_ld'`: A `numpy.ndarray` with two components of the computed shift of type `numpy.longdouble`.
- `'matrix'`: A `2x2` `numpy.ndarray` with the computed generalized rotation matrix.
- `'matrix_ld'`: A `2x2` `numpy.ndarray` with the computed generalized rotation matrix of type `numpy.longdouble`.
- `'proper_rot'`: Rotation angle (degree) as if the rotation is proper.
- `'rot'`: A tuple of `(rotx, roty)` - the rotation angles with regard to the `X` and `Y` axes.
- `'<rot>'`: *Arithmetic mean* of the angles of rotation around `X` and `Y` axes.
- `'scale'`: A tuple of `(sx, sy)` - scale change in the direction of the `X` and `Y` axes.

- `'<scale>'`: *Geometric mean* of scales `sx` and `sy`.
- `'skew'`: Computed skew.
- `'proper'`: a boolean indicating whether the rotation is proper.
- `'fitgeom'`: Fit geometry (allowed transformations) used for fitting data (to minimize residuals). This is copy of the input argument `fitgeom`.
- `'center'`: Center of rotation
- `'center_ld'`: Center of rotation as a `numpy.longdouble`.
- `'fitmask'`: A boolean array indicating which source positions where used for fitting (`True` (https://docs.python.org/3/library/constants.html#True)) and which were clipped out (`False` (https://docs.python.org/3/library/constants.html#False)). **NOTE** For weighted fits, positions with zero weights are automatically excluded from the fits.
- `'eff_nclip'`: Effective number of clipping iterations
- `'rmse'`: Root-Mean-Square Error
- `'mae'`: Mean Absolute Error
- `'std'`: Standard Deviation of the residuals
- `'resids'`: An array of residuals of the fit. **NOTE:** Only the residuals for the "valid" points are reported here. Therefore the length of this array may be smaller than the length of input arrays of positions.

Notes

Weights

Weights can be provided for both "image" source positions and "reference" source positions. When no weights are given, all positions are weighted equally. When only one set of positions have weights (i.e., either `wxy` or `wuv` is not `None` (https://docs.python.org/3/library/constants.html#None)) then weights in (1.2) are set to be equal to the provided set of weights. When weights for *both* "image" source positions and "reference" source positions are provided, then the combined weight that is used in (1.2) is computed as:

$$1/w = 1/w_{xy} + 1/w_{uv}.$$

Statistics for clipping

Several statistics are available for clipping iterations and all of them are reported in the returned `fit` dictionary regardless of the setting in `sigma`:

$$\mathrm{RMSE} = \sqrt{\Sigma_k w_k \|\mathbf{r}_k\|^2}$$

$$\mathrm{MAE} = \sqrt{\Sigma_k w_k \|\mathbf{r}_k\|}$$

$$\mathrm{STD} = \sqrt{\Sigma_k w_k \|\mathbf{r}_k - \bar{\mathbf{r}}\|^2}/(1 - V_2)$$

where $\mathbf{r}_k = \mathbf{xy}_k - \mathbf{xy}'_k$, $\Sigma_k w_k = 1$, and $V_2 = \Sigma_k w_k^2$.

`tweakwcs.linearfit.`**`build_fit_matrix`**(*rot*, *scale=1*)

Create an affine transformation matrix (2x2) from the provided rotation angle(s) and scale(s):

$$M = \begin{bmatrix} s_x \cos(\theta_x) & s_y \sin(\theta_y) \\ -s_x \sin(\theta_x) & s_y \cos(\theta_y) \end{bmatrix}$$

Parameters

rot: tuple, float, optional Rotation angle in degrees. Two values (one for each axis) can be provided as a tuple.

scale: tuple, float, optional Scale of the liniar transformation. Two values (one for each axis) can be provided as a tuple.

Returns

matrix: numpy.ndarray A 2x2 `numpy.ndarray` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray) containing coefficients of a liniear transformation.

1.6 wcsutils

A module that provides utility functions for WCS transformations.

Authors Mihai Cara (contact: help@stsci.edu)

`tweakwcs.wcsutils.`**`planar_rot_3d`**(*angle*, *axis*)

Create a 3D rotation matrix that performs a rotation *in a plane* perpendicular to the specified `axis`.

1.7 linalg

This module provides general purpose and/or specialized linear algebra routines.

Authors Mihai Cara (contact: help@stsci.edu)

License *LICENSE*

`tweakwcs.linalg.`**`inv`**(*m*)

This function computes inverse matrix using Gauss-Jordan elimination with full pivoting. Computations are performed using `numpy.longdouble` precision. On systems on which `numpy.longdouble` is equivalent to `numpy.double` this function reverts to `numpy.linalg.inv` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.inv.html#numpy.linalg.inv) for performance reasons.

Parameters

m: numpy.ndarray A 2D *square* matrix of type `numpy.ndarray` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray).

Returns

invm: numpy.ndarray Inverse matrix of the input matrix `m`: a 2D *square* `numpy.ndarray` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html#numpy.ndarray) of type `numpy.longdouble` on systems on which it is more accurate than `numpy.double`.

1.8 LICENSE

CHAPTER 2

Development Notes

2.1 Release Notes

2.1.1 0.6.4 (14-May-2020)

- Bug fix: Unable to initialize `JWSTgWCS` tangent-plane corrector from an already corrected WCS. [#122]
- Fix a bug in how corrections are applied to a previously corrected JWST WCS. [#120]
- Do not attempt to extract center of linear transformation when not available in `'fit_info'`. [#119]

2.1.2 0.6.3 (14-April-2020)

- Fixed a bug due to which reprojection transformation for JWST gWCS was computed at wrong location in the tangent plane. [#118]

2.1.3 0.6.2 (07-April-2020)

- When WCS has valid bounding box, estimate scale at the center of the bounding box. [#117]
- Adjust the point at which tangent plane-to-tangent plane transformation is computed by 1/2 pixels for JWST corrections. This correction should have no measurable impact on computed corrections. [#115]

2.1.4 0.6.1 (09-March-2020)

- Fixed a bug in applying JWST correction for the case when alignment is performed twice on the same image. Due to this bug the inverse transformation was not updated. [#112]

2.1.5 0.6.0 (25-February-2020)

- Fix a possible crash when aligning FITS WCS images due to an unusual way `stwcs.wcsutil.all_world2pix` handles (or not) scalar arguments. [#110]
- Modified the angle at which the reported rotation angles are reported. Now rotation angles have the range `[-180, 180]` degrees. [#109]
- Added support FITS WCS that use `PC` matrix instead of the `CD` matrix used in HSTs WCS. [#108]
- Bug fix for alignment of multi-chip FITS images: correction of how transformations from the reference tangent plane are converted to individual images' tangent planes. [#106]
- Significant re-organization of the `fit_info` dictionary. `rot` now becomes `proper_rot` and `rotxy` now becomes `rot` containing only `rotx` and `roty`. Also, `scale` now is a tuple of only two scales `sx` and `sy`. The geometric mean scale is now a separate field `'<scale>'` as well as the arithmetic mean of rotation angles (`'<rot>'`). Finally, `'offset'` in the fit functions from the `linearfit` module was renamed to `'shift'` in order to match the same field returned by functions from the `imalign` module. [#105]
- Linear fit functions now return the fit matrix `F` instead of its transpose. [#100]
- Linear fit functions (in the `linearfit` module) use `longdouble` for internal computations. [#100]
- Re-designed the `JWSTgWCS` corrector class to rely exclusively on basic models available in `astropy` and `gwcs` instead of the `TPCorr` class provided by the `jwst` pipeline. This eliminates the need to install the `jwst` pipeline in order to align `JWST` images. [#96, #98]

2.1.6 0.5.3 (15-November-2019)

- Added logic to allow some input catalogs to be empty and to allow the alignment to proceed as long as there are at least two non-empty (image or group) input catalogs. [#94]

2.1.7 0.5.2 (26-July-2019)

- Fixed a deprecation issue in logging and added logic to compute image group's catalog name using a common prefix (if exists) of the names of constituent images. [#92]
- Package version is now handled by `setuptools_scm`. [#93]

2.1.8 0.5.1 (08-May-2019)

- Fixed a bug in the "2dhist" algorithm resulting in a crash when 2D histogram has multiple maxima of the same value and no other value larger than one. [#90]

2.1.9 0.5.0 (22-April-2019)

- Fixed a bug due to which a warning log message "Failed to align catalog..." would be issued for successful alignments. [#84]
- Fixed a bug in creation of WCS image groups with empty catalogs. [#84]
- Fixed a bug in `match2ref` when it was run in a non-matching mode (`match=None`) dute to which it was impossible to detect the case when reference catalog has a different length from a supposedly matched WCS group catalog. [#84]
- Fixed a bug in computation of the bounding polygon of a reference catalog containing only two sources. [#84]
- Fixed a bug in `convex_hull()` resulting in incorrect type being returned in case of empty input coordinate lists or whne only one point is provided. [#84]
- Implemented a more robust estimate of the maximum type supported by `numpy.linalg.inv`. [#82]
- Renamed `wcsutils.planar_rot_3D` to `wcsutils.planar_rot_3d`. [#75]
- Renamed `wcsutils.cartesian2spherical` to `wcsutils.cartesian_to_spherical` and `wcsutils.spherical2cartesian` to `wcsutils.spherical_to_cartesian`. [#71]
- Improved "2dhist" algorithm that performs simple catalog pre-alignment used for source matching. [#69]
- Changed the default value of the `searchrad` parameter in `matchutils.TPMatch` to 3. [#69]

2.1.10 0.4.5 (14-March-2019)

- Fixed incorrect pointer type introduced in previous release [#67].

2.1.11 0.4.4 (13-March-2019)

- Fixed VS2017 compiler error, `"void *": unknown size`. [#62, #63, #64]

2.1.12 0.4.3 (13-March-2019)

- Package maintenance release.

2.1.13 0.4.2 (21-February-2019)

- Fixed a bug due to which the fitting code would crash is `wuv` were provided but `wxy` were set to `None`. [#60]

2.1.14 0.4.1 (14-February-2019)

- Code cleanup: removed debug print statements. [#59]

2.1.15 0.4.0 (08-February-2019)

- Matched indices, linear fit results and fit residuals are now set in the input "WCS catalogs" `meta['fit_info']` instead of `meta['tweakwcs_info']`. [#57]
- Updated example notebook to reflect changes to API. [#57]
- Allow `TPWCS` classes to set `meta` during object instantiation. This allows attaching, for example, a source catalog to the tangent-plane WCS corrector object. [#57]
- `align_wcs` no longer supports `NDData` input. Instead catalogs can be provided directly in the `meta` attribute of `TPWCS`-derived WCS "correctors". This fundamentally transfers the responsibility of instantiating the correct tangent-plane WCS to the caller. This, in turn, will allow future WCS to be supported by providing a custom `TPWCS`-derived corrector defined externally to `tweakwcs` package. Second benefit is that image data no longer need to be kept in memory in `NDData` objects as image data are not needed for image alignment once catalogs have been created. [#57]
- Renamed `tweak_wcs` to `fit_wcs` and `tweak_image_wcs` to `align_wcs`. [#57]
- Fixed a bug due to which the code might crash due to an undefined `ra` variable, see issue #55. [#56]
- `tweak_image_wcs()` now returns effective reference catalog used for image alignment. [#54]
- Modified how IDs are assigned to the reference catalog source positions when `expand_refcat` is `True` (https://docs.python.org/3/library/constants.html#True): instead of having all sources numbered consecutively starting with 1, now the code will attempt to preserve the original IDs (if any) of the input reference catalog (`refcat`) or an input image used as a reference catalog and consecutively number only the sources being added to the `refcat`. [#54]
- Modified the clipping algorithm to start with all valid sources at each iteration. In other words, clippings do not accumulate by default. Old behavior can be replicated by setting `clip_accum` to `True` (https://docs.python.org/3/library/constants.html#True). [#53]
- Cleaned-up `iter_linear_fit` interface as well as simplified the `fit` dictionary returned by `iter_linear_fit`. [#53]
- Added option to specify statistics used for clipping. [#51, #52]

2.1.16 0.3.3 (21-January-2019)

- Corrected a bug in the non-weighted `rscale` fit. [#49]

- Corrected a bug in the computation of `RMSE` for the "general" fit. [#47]
- Added computation of `MAE` of the fit (in addition to `RMSE`), see [Mean Absolute Error](https://en.wikipedia.org/wiki/Mean_absolute_error). [#47]
- Renamed `RMSD` to `RMSE` (Root-Mean-Square Error). [#47]

2.1.17 0.3.2 (15-January-2019)

- Fixed the formula for computing `RMSD` of non-weighted fit. [#46]

2.1.18 0.3.1 (14-January-2019)

- Fixed Read-The-Docs build failure. [#45]

2.1.19 0.3.0 (14-January-2019)

- Implemented higher-accuracy matrix inversion. [#42]
- Bug fix related to not switching to using `bounding_box` instead of `pixel_shape`. [#41]
- Added support for optional `'weight'` column in catalogs indicating the weight of each source in fitting linear transformations. [#41]
- Add support for weights to the linear fitting routines. [#40]
- Replaced the use of `RMS` for each axis with a single `RMSD` value, see [Root-Mean-Square Deviation](https://en.wikipedia.org/wiki/Root-mean-square_deviation). [#40]
- Rely on `pixel_bounds` [see APE 14](https://github.com/astropy/astropy-APEs/blob/master/APE14.rst) when available for computation of image's bounding box. [#39]
- Fix a bug in the computation of the world coordinates of the fitted (*aligned*) sources. [#36]

2.1.20 0.2.0 (20-December-2018)

- Fix swapped reported reference and input indices of sources used for fitting. [#34]
- Fix for non-initialized C arrays. [#34]
- Changelog correction. [#33]

2.1.21 0.1.1 (11-December-2018)

- Fixeded a bug due to which `'fit_ref_idx'` and `'fit_input_idx'` fields in the `fit` dictionary were never updated. [#31]
- `jwst` (pipeline) package is no longer a hard dependency. [#30]
- Removed unnecessary install dependencies. [#30]

- Documentation improvements. [#30, #32]
- Corrected 'RA', 'DEC' units used to compute bounding polygon for the reference catalog. [#30]
- Updated `C` code to avoid `numpy` deprecation warnings. [#30]

2.1.22 0.1.0 (08-December-2018)

- Added support for aligning FITS WCS. [#15, #16]
- Added keywords to `meta` attributes of the `TPWCS` and `NDData` to allow easy access to the match and fit information. [#20, #21, #28]
- Package and setup re-design. Support for `readthedocs`. [#23]
- Documentation improvements. [#17, #18]
- Numerous other bug fixes, code clean-up, documentation improvements and enhancements. [#2, #3, #4, #5, #6, #7, #8, #9, #10, #11, #12, #13, #14, #19, #22, #24, #25, #26, #27, #28, #29]

2.1.23 0.0.1 (25-April-2018)

Initial release. [#1]

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

T

U

W