i3assist Documentation

Release 0.0.1

Dustin Reed Morado

Nov 05, 2016

CONTENTS

Contents:

CHAPTER

ONE

I3ASSIST

1.1 A python library to facilitate using I3

i3assist is a python library designed to make using I3 slightly easier. It is inspired by my work using pytom and enjoying the flexibilty provided by a clean and object oriented API around averaging and classification jobs.

CHAPTER

TWO

I3ASSIST

2.1 i3assist module

A module to help using I3 for sub-tomogram averaging.

Written By Dustin Reed Morado Last updated 04.11.2016

class `i3assist.`**`Euler`**`(`*phi=0.0*, *theta=0.0*, *psi=0.0*, *unit='deg'*`)`

Bases: `object`

Describes a particle's orientation using ZXZ extrinsic euler angles.

The first rotation (phi) is about the z-axis (z). The second rotation (theta) is about the new x-axis (x'). The third rotation (psi) is about the new z-axis (z'') after the second rotation.

All rotations are of the coordinate axes and not of point coordinates.

Parameters

- **`phi`** (`float`, optional) – First rotation about the z-axis (z).
- **`theta`** (`float`, optional) – Second rotation about the new x-axis (x').
- **`psi`** (`float`, optional) – Third rotation about the new z-axis (z'').
- **`unit`** (`str`, optional) – Whether rotations are in degrees "deg" or radians "rad".

`angles`

`list` of `float` – Array of rotations in order.

Array is in the order phi, theta, psi

Parameters **`value`** (`list` of `float`) – List-type array of the three rotations.

`copy``()`

Returns a copy of the euler object.

Returns Copy of Euler object.

Return type *`i3assist.Euler`*

`degrees``(`*inplace=False*`)`

Converts Euler object to describe rotations in units of degrees.

Parameters **`inplace`** (`boolean`, optional) – If True conversion will be done in-place.

Returns

Copy or self object with rotations in degree units.

Return type *`i3assist.Euler`*

invert(*inplace=False*)

Returns a copy of euler object with inverted rotations.

Again we can simply take the negative of psi, theta, phi to invert the rotatations but it is more theoretically clear to convert the angles to a rotation matrix, transpose it, and convert it back to angles. We also get the added benefit of normalizing the angles.

Parameters **inplace** – If set to True the Euler angles will be changed in-place instead of returning a copy.

Returns

Copy or self object describing the opposite rotation.

Return type *i3assist.Euler*

normalize(*inplace=False*)

Returns a copy of euler object with rotations in old I3 bounds.

Old I3 requires that euler angles be given within the bounds: phi: -180 to 180 theta: 0 to 180 psi: -180 to 180

There are many equivalent euler angle formats and we can normalize the angles using these identities, but these ranges correspond to the ranges given by the default C math arc trig functions, which are used in going from rotation matrix to Euler angles so we just convert to rotation matrix and then back to Euler angles.

Parameters **inplace** – If set to True the Euler angles will be changed in-place instead of returning a copy.

Returns

Copy or self object with rotations normalized to ranges required by old I3.

Return type *i3assist.Euler*

phi

`float` – First rotation about the z-axis (z).

Parameters **value** (`float`) – New phi rotation.

pos_string()

Returns string of angles in old I3 pos format.

Returns Old I3 pos format string of euler angles.

Return type `str`

psi

`float` – Third rotation about the new z-axis (z”).

Parameters **value** (`float`) – New psi rotation.

radians(*inplace=False*)

Converts Euler object to describe rotations in units of radians.

Parameters **inplace** (`boolean`, optional) – If True conversion will be done in-place.

Returns

Copy or self object with rotations in radian units.

Return type *i3assist.Euler*

theta

`float` – Second rotation about the new x-axis (x’).

Parameters **`value`** (`float`) – New theta rotation.

`to_matrix`()

Returns the RotationMatrix object equivalent of Euler object.

Returns

Rotation matrix describing the equivalent euler angles.

Return type *`i3assist.RotationMatrix`*

`trf_string`()

Returns string of angles in new I3 trf format.

Returns New I3 trf format string of rotation matrix

Return type `str`

`unit`

`str` – Describes the unit of rotations as degrees or radians.

Parameters **`value`** (`str`) – "deg" to set unit as degrees or "rad" to set unit as radians.

class `i3assist.`**`GridSearch`**(*theta_max=0.0*, *theta_step=0.0*, *psi_max=0.0*, *psi_step=0.0*, *do_180=False*)

Bases: `object`

Describes the local grid search implemented in new I3.

See the explanation in MRASRCH and the I3 subvolume tutorial for more information on how the grid search is implemented. But overall the grid is defined by four parameters: Nutation (theta) maximum and step and Spin (psi) maximum and step. Finally there is a do_180 parameter to support the old I3 eulerFG scripts, but this is not available in new I3.

Parameters

- **`theta_max`** (`float`, optional) – Maximum half-angle of a cone of nutation about the north pole of the unit sphere.
- **`theta_step`** (`float`, optional) – Angular increment of nutation.
- **`psi_max`** (`float`, optional) – Maximum absolute angle of spin about the orientation axis of the particle. Searched in both directions.
- **`psi_step`** (`float`, optional) – Angular increment of spin.
- **`do_180`** (`boolean`, optional) – If True the spins opposite of the current orientation's facing will also be searched.

`do_180`

`boolean` Whether to search spins opposite particle facing.

Parameters **`value`** (`bool`) – True to search opposite facing spin angles.

`psi_max`

`float` Max absolute angle of spin about the particle z-axis.

Parameters **`value`** (`float`) – Max absolute spin angle in range 0 to 180.

`psi_step`

`float` Angular increment of spin.

Parameters **`value`** (`float`) – Spin angular increment in the range 0 to 180.

`rotations`

`list` of *`i3assist.Euler`* Rotations searched in i3.

Parameters **`params`** (`tuple` of `float` and `boolean`) – A tuple with theta_max, theta_step, psi_max, psi_step, and do_180.

`theta_max`

`float` Max half-angle of nutation about the north pole.

Parameters **`value`** (`float`) – half-angle in the range 0 to 180.

`theta_step`

`float` Angular increment of nutation.

Parameters **`value`** (`float`) – Angular increment in the range 0 to 180.

class `i3assist.`**`RotationMatrix`**(*angles=None*, *unit='deg'*)

Bases: `object`

Describes a particle's orientation using ZXZ passive rotation matrix.

The rotation matrix is the composition of three rotations matrices with the first and third being about the Z-axis and the second about the X-axis.

All rotations are passive (alias) transformations of the coordinate axes and not of point coordinates (active / alibi).

Parameters

- **`angles`** (`list` of `float`, optional) – Euler angles describing the rotation.
- **`unit`** (`str`, optional) – Whether rotations are in degrees "deg" or radians "rad"

`copy`()

Returns a copy of the rotation matrix.

`invert`(*inplace=False*)

Returns rotation matrix describing opposite rotation.

Rotation matrices are orthogonal and hence their transpose is their inverse.

Parameters **`inplace`** (`boolean`, optional) – If True the operation will be done on the object in-place instead of returning a copy.

Returns

Copy or self object of the inverted rotation matrix.

Return type *`i3assist.RotationMatrix`*

`matrix`

`numpy.ndarray` – (3,3) matrix describing the rotation.

Matrix is ordered in the standard format with the first index describing the row and the second index describing the column.

Parameters **`value`** (`numpy.ndarray`) – Euler angles in radians or Rotation matrix.

`pos_string`()

Returns string of angles in old I3 pos format.

Returns Old I3 pos format string of euler angles.

Return type `str`

`to_euler`(*unit='deg'*)

Converts rotation matrix to equivalent euler angles.

Algorithm from Chapter 1 of "Computational Methods for Three-Dimensional Microscopy Reconstruction" Ed. Joachim Frank, Gabor Herman.

Parameters **`unit`** (`str`, optional) – Whether to return the Euler angles in degrees or radians.

Returns Euler angle equivalent of rotation matrix

Return type *`i3assist.Euler`*

`transpose`(*inplace=False*)
Returns rotation matrix describing opposite rotation.

Rotation matrices are orthogonal and hence their transpose is their inverse.

Parameters **`inplace`** (`boolean`, optional) – If True the operation will be done on the object in-place instead of returning a copy.

Returns

Copy or self object of the inverted rotation matrix.

Return type *`i3assist.RotationMatrix`*

`trf_string`()
Returns string of angles in new I3 trf format.

Returns New I3 trf format string of rotation matrix

Return type `str`

class `i3assist.`**`Transform`**(*transform_line*)
Bases: `object`

A single particle transform in new I3.

For more information refer to the subvolume tutorial document for I3.

Parameters **`transformLine`** (`str`) – A string with the transform data.

`add_rotation`(*rotation*, *inplace=False*)
Adds a rotation in addition to particles current orientation.

Parameters

- **`rotation`** (*`i3assist.RotationMatrix`*) – Rotation to add.
- **`inplace`** – If True the operation will be done in-place and modify the transform instead of returning a copy.

Returns Copy or self rotated transform object.

Return type *`i3assist.Transform`*

`add_shift`(*shift_x=0.0*, *shift_y=0.0*, *shift_z=0.0*, *inplace=False*)
Adjusts the particles defined center by an arbitrary vector.

Parameters

- **`shift_x`** (`float`, optional) – Amount to shift in x.
- **`shift_y`** (`float`, optional) – Amount to shift in y.
- **`shift_z`** (`float`, optional) – Amount to shift in z.
- **`inplace`** (`float`, optional) – If True the operation will be done in-place and modify thet transform instead of returning a copy.

Returns Copy or self shifted transform object.

Return type *`i3assist.Transform`*

class_number
`int` Class number that the particle belongs to.

Parameters **value** (`int`) – Particle Class.

coordinates
`list` of `int` Particles integer coordinates in tomogram.

Coordinates are stored here as column arrays to help with using them with the rotation matrices.

Parameters **value** (`list` of `int`) – List with 3 elements for particles coordinates relative to the tomogram map it's extracted from.

copy()
Returns a copy of the transform.

rotation
i3assist.RotationMatrix Particles rotation matrix to orient.

Parameters **value** (`list` of `float`) – Rotation matrix.

scale(*scale_factor*, *inplace=False*)
Scale the transform to handle binning.

Parameters

- **scale_factor** (`float`) – Amount to scale transform by.
- **inplace** – If True the operation will be done in-place and modify the transform instead of returning a copy.

Returns Copy or self scaled transform object.

Return type *i3assist.Transform*

score
`float` Correlation score of particle alignment to reference.

Parameters **value** (`float`) – Correlation score.

shifts
`list` of `float` Particle displacements from coordinates.

Displacements are stored here as column arrays to help with using them with the rotation matrices. Again as for rotations translations are alibi translations of the coordinate system and not the actual points.

Parameters **value** (`list' of :obj`float`) – List with 3 elements for particles shifts relative to the center of the reference.

subset
`str` Subset identifier for particle.

Parameters **value** (`str`) – Subset name around 10 characters.

class i3assist.**TransformList**(*filename='*', *transforms=None*)
Bases: `object`

Describes a full transform file with as a list of Transforms.

For more information refer to the subvolume tutorial document for I3.

Parameters

- **filename** (`str`) – Filename of trf file.
- **transforms** (`list` of *i3assist.Transform*) – List of transforms.

filename
Filename associated with transform list.

Parameters **value** (`str`) – Filename of trf file.

from_file (*filename*)
Loads list of transforms from a trf file.

Parameters **filename** (`str`) – Filename of trf file.

get_by_class (*class_number*)
Gets a subset of a transform list based on the class number.

Parameters **class_number** (`int`) – The class number to search for.

Returns

Subset of self that matches the. class number requested.

Return type *i3assist.TransformList*

get_by_subset (*subset*)
Gets a subset of a transform list based on the subset field.

Parameters **subset** (`str`) – The subset field to search for.

Returns

Subset of self that matches the. subset requested.

Return type *i3assist.TransformList*

scale (*scale_factor*, *inplace=False*)
Scales all of the transforms in a list.

Parameters

- **scale_factor** (`float`) – Amount which to scale the transforms.
- **inplace** – If True, scaling will be done in-place instead of returning a copy.

Returns Scaled transform list.

Return type *i3assist.TransformList*

sort_by_class (*inplace=False*)
Sorts a transform list by class numbers.

Parameters **inplace** – If True the list is not returned and the sorting is done in-place. Otherwise a sorted copy is returned.

Returns Transform list sorted by class.

Return type *i3assist.TransformList*

sort_by_score (*inplace=False*)
Sorts a transform list by correlation coefficient.

Parameters **inplace** – If True the list is not returned and the sorting is done in-place. Otherwise a sorted copy is returned.

Returns Transform list sorted by score.

Return type *i3assist.TransformList*

to_file (*filename*)
Writes out a Transform list to a trf file.

Parameters **`filename`** (`str`) – Name of file to write to.

`transforms`

List of Transform objects.

Parameters **`value`** (`list` of *`i3assist.Transform`*) – List of transforms.

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX