stimage Documentation

Release 0.1

Michael Droettboom

June 23, 2016

CONTENTS

1 Functions 3

2 Indices and tables 11

Python Module Index 13

Index 15

Contents:

CHAPTER ONE

FUNCTIONS

`stsci.stimage.`**`xyxymatch`**(*input*, *ref*, *origin=(0.0, 0.0)*, *mag=(1.0, 1.0)*, *rotation=(0.0, 0.0)*, *ref_origin=(0.0, 0.0)*, *algorithm='tolerance'*, *tolerance=1.0*, *separation=9.0*, *nmatch=30*, *maxratio=10.0*, *nreject=10*)

Match pixels coordinate lists using various methods.

`xyxymatch` matches the *x* and *y* coordinates in the reference coordinate list reference to the corresponding *x* and *y* coordinates in the input coordinate list input to within a user specified tolerance *tolerance*, and returns the matched coordinates in a structured array. The output results are suitable as input to the `geomap` function which compute the actual transformation required to register the corresponding reference and input images.

`xyxymatch` matches the coordinate lists by:

1. computing an initial guess at the linear transformation required to match the input coordinate system to the reference coordinate system
2. applying the computed transformation to the input coordinates
3. sorting the reference and input coordinates and removing points with a minimum separation specified by the parameter separation from both lists
4. matching the two lists using either the "tolerance" or "triangles" algorithm
5. storing the matched list to the output array

The coordinate lists are matched using the algorithm specified by the *algorithm* parameter.

- If *algorithm* is "tolerance", `xyxymatch` searches the sorted transformed input coordinate list for the object closest to the current reference object within the matching tolerance *tolerance*. The major advantage of the "tolerance" algorithm is that it can deal with *x* and *y* scale differences and axis skew in the coordinate transformation. The major disadvantage is that the user must supply tie point information in all but the simplest case of small *x* and *y* shifts between the input and reference coordinate systems.
- If *algorithm* is "triangles", `xyxymatch` contructs a list of triangles using up to *nmatch* reference coordinates and transformed input coordinates, and performs a pattern matching operation on the resulting triangle lists. If the number of coordinates in both lists is less than *nmatch*, the entire list is matched using the "triangles" algorithm directly, otherwise the "triangles" algorithm is used to estimate a new linear transformation, the input coordinate list is transformed using the new transformation, and the entire list is matched using the "tolerance" algorithm. The major advantage of the "triangles" algorithm is that it requires no tie point information from the user. The major disadvantages are that it is sensitive to *x* and *y* scale differences and axis skews between the input and reference coordinate systems and can be computationally expensive.

  The "triangles" algorithm uses a sophisticated pattern matching technique which requires no tie point information from the user. It is expensive computationally and hence is restricted to a maximum of *nmatch* objects from the reference and input coordinate lists.

The "triangles" algorithm first generates a list of all the possible triangles that can be formed from the points in each list. For a list of *nmatch* points, this number is the combinatorial factor `nmatch! / [(nmatch-3)! * 3!]` or `nmatch * (nmatch-1) * (nmatch-2) / 6`. The length of the perimeter, ratio of longest to shortest side, cosine of the angle between the longest and shortest side, the tolerances in the latter two quantities and the direction of the arrangement of the vertices of each triangle are computed and stored in a table. Triangles with vertices closer together than *tolerance* or with a ratio of the longest to shortest side greater than *ratio* are discarded. The remaining triangles are sorted in order of increasing ratio. A sort merge algorithm is used to match the triangles using the ratio and cosine information, the tolerances in these quantities, and the maximum tolerances for both lists. Next the ratios of the perimeters of the matched triangles are compared to the average ratio for the entire list, and triangles which deviate too widely from the mean are discarded. The number of triangles remaining are divided into the number which match in the clockwise sense and the number which match in the counter-clockwise sense. Those in the minority category are eliminated. The rejection step can be repeated up to *nreject* times or until no more rejections occur – whichever comes first. The last step in the algorithm is a voting procedure in which each remaining matched triangle casts three votes, one for eached matched pair of vertices. Points which have fewer than half the maximum number of votes are discarded. The final set of matches are written to the output file.

The "triangles" algorithm functions well when the reference and input coordinate lists have a sufficient number of objects (~50%, in some cases as low as 25%) of their objects in common, any distortions including *x* and *y* scale differences and skew between the two systems are small, and the random errors in the coordinates are small. Increasing the value of the *tolerance* parameter will increase the ability to deal with distortions but will also produce more false matches.

Parameters:

- *input*: Array of input coordinates. (Must be an Nx2 array).
- *ref*: Array of reference coordinates. (Must be an Nx2 array).
- *origin*: The origin of the input coordinate system. Default: (0.0, 0.0)
- *mag*: The scale factor in reference pixels per input pixels. Default: (1.0, 1.0)
- *ref_origin*: The origin of the reference coordinate system. Default: (0.0, 0.0)
- *algorithm*: The matching algorithm. The choices are:
  - `'tolerance'`: A linear transformation is applied to the input coordinate list, the transformed input list and the reference list are sorted, points which are too close together are removed, and the input coordinates which most closely match the reference coordinates within the user-specified tolerance are determined. The tolerance algorithm requires an initial estimate for the linear transformation. This estimate can be derived from a set of tie points, or by setting the parameters *origin*, *mag*, *rotation* and *ref_origin*. Assuming that well-chosen tie points are provided, the tolerance algorithm functions well in the presence of any shifts, axis flips, *x* and *y* scale changes, rotations, and axis skew, between the two coordinate systems. The algorithm is sensitive to higher-order distortion terms in the coordinate transformation.
  - `'triangles'`: A linear transformation is applied to the input coordinate list, the transformed input list and the reference list are sorted, points which are too close together are removed, and the input coordinates are matches to the reference coordinates using a triangle pattern matching technique and the user-specified tolerance parameter. The triangles pattern matching algorithm does not require prior knowledge of the linear transformation, although it will use one if one is supplied. The algorithm functions well in the presence of any shifts, axis flips, magnification and rotation between the two coordinate systems as long as both lists have a reasonable number of objects in common and the errors in the computed coordinates are small. However, since the algorithm depends on comparisons of similar triangles, it is sensitive to differences in the *x* and *y* coordinate scales, any skew between the *x* and *y* axes, and higher order distortion terms in the coordinate transformation.

- *tolerance*: The matching tolerance in pixels. Default: 1.0
- *separation*: The minimum separation for objects in the input and reference coordinate lists. Objects closer together than *separation* pixels are removed from the input and reference coordinate lists prior to matching. Default: 9.0
- *nmatch*: The maximum number of reference and input coordinates used by the `'triangles'` pattern matching algorithm. If either list contains more coordinates than *nmatch*, the lists are subsampled. *nmatch* should be kept small as the computation and memory requirements of the triangles algorithm depend on a high power of lengths of the respective lists. Default: 30
- *maxratio*: The maximum ratio of the longest to shortest side of the triangles generated by the triangles pattern matching algorithm. Triangles with computed longest to shortest side. Ratios greater than *ratio* are rejected from the pattern matching algorithm. *ratio* should never be set higher than 10.0 but may be set as low as 5.0. Default: 10.0
- *nreject*: The maximum number of rejection iterations for the `'triangles'` pattern matching algorithm. Default: 10

Returns: A structured array containing the output information. It has the following columns:

- *input_x*
- *input_y*
- *input_idx*
- *ref_x*
- *ref_y*
- *ref_idx*

`stsci.stimage.`**`geomap`**(*input*, *ref*, *bbox=None*, *fit_geometry='general'*, *function='polynomial'*, *xxorder=2*, *xyorder=2*, *yxorder=2*, *yyorder=2*, *xxterms='half'*, *yxterms='half'*, *maxiter=0*, *reject=0.0*)

`geomap` computes the transformation required to map the reference coordinate system to the input coordinate system.

The coordinates of points in common to the two systems are listed in the same location in the *input* and *ref* arrays.

The computed transformation has the form shown below:

```
xin = f (xref, yref)
yin = g (xref, yref)
```

If, on the other hand, the user wishes to transform coordinates from the input image coordinate system to the reference coordinate system then he or she must reverse the roles of the reference and input coordinates as defined above, and compute the inverse transformation.

The functions *f* and *g* are either a power series polynomial or a Legendre or Chebyshev polynomial surface of order *xxorder* and *xyorder* in *x* and *yxorder* and *yyorder* in *y*.

Several polynomial cross terms options are available. Options "none", "half", and "full" are illustrated below for a quadratic polynomial in *x* and *y*:

```
xxterms = "none", xyterms = "none"
xxorder = 3, xyorder = 3, yxorder = 3, yyorder = 3

xin = a11 + a21 * xref + a12 * yref +
      a31 * xref ** 2 + a13 * yref ** 2
yin = a11' + a21' * xref + a12' * yref +
      a31' * xref ** 2 + a13' * yref ** 2
```

```
xxterms = "half", xyterms = "half"
xxorder = 3, xyorder = 3, yxorder = 3, yyorder = 3

xin = a11 + a21 * xref + a12 * yref +
      a31 * xref ** 2 + a22 * xref * yref + a13 * yref ** 2
yin = a11' + a21' * xref + a12' * yref +
      a31' * xref ** 2 + a22' * xref * yref + a13' * yref ** 2

xxterms = "full", xyterms = "full"
xxorder = 3, xyorder = 3, yxorder = 3, yyorder = 3

xin = a11 + a21 * xref + a31 * xref ** 2 +
      a12 * yref + a22 * xref * yref +  a32 * xref ** 2 * yref +
      a13 * yref ** 2 + a23 * xref *  yref ** 2 +
      a33 * xref ** 2 * yref ** 2
yin = a11' + a21' * xref + a31' * xref ** 2 +
      a12' * yref + a22' * xref * yref +  a32' * xref ** 2 * yref +
      a13' * yref ** 2 + a23' * xref *  yref ** 2 +
      a33' * xref ** 2 * yref ** 2
```

If the *fit_geometry* parameter is anything other than "general", the order parameters assume the value 2 and the cross terms switches assume the value "none", regardless of the values set by the user.

Automatic pixel rejection may be enabled by setting *maxiter* > 0 and *reject* to some number greater than 0.

bbox defines the region of validity of the fit in the reference coordinate system and may be set by the user. These parameters can be used to reject out-of-range data before the actual fitting is done.

The transformation computed by the "general" fitting geometry is arbitrary and does not correspond to a physically meaningful model. However, the computed coefficients for the linear term can be given a simple geometrical geometric interpretation for all the fitting geometries as shown below:

```
fitting geometry = general (linear term)
    xin = a + b * xref + c * yref
    yin = d + e * xref + f * yref

fitting geometry = shift
    xin = a + xref
    yin = d + yref

fitting geometry = xyscale
    xin = a + b * xref
    yin = d + f * yref

fitting geometry = rotate
    xin = a + b * xref + c * yref
    yin = d + e * xref + f * yref
    b * f - c * e = +/-1
    b = f, c = -e or b = -f, c = e

fitting geometry = rscale
    xin = a + b * xref + c * yref
    yin = d + e * xref + f * yref
    b * f - c * e = +/- const
    b = f, c = -e or b = -f, c = e

fitting geometry = rxyscale
    xin = a + b * xref + c * yref
    yin = d + e * xref + f * yref
```

```
        b * f - c * e = +/- const
```

The coefficients can be interpreted as follows. `xref0`, `yref0`, `xin0`, `yin0` are the origins in the reference and input frames respectively. Orientation and skew are the rotation of the *x* and *y* axes and their deviation from perpendicularity respectively. `xmag` and `ymag` are the scaling factors in *x* and *y* and are assumed to be positive:

```
    general (linear term)
        xrotation = rotation - skew / 2
        yrotation = rotation + skew / 2
        b = xmag * cos (xrotation)
        c = ymag * sin (yrotation)
        e = -xmag * sin (xrotation)
        f = ymag * cos (yrotation)
        a = xin0 - b * xref0 - c * yref0 = xshift
        d = yin0 - e * xref0 - f * yref0 = yshift

    shift
        xrotation = 0.0,  yrotation = 0.0
        xmag = ymag = 1.0
        b = 1.0
        c = 0.0
        e = 0.0
        f = 1.0
        a = xin0 - xref0 = xshift
        d = yin0 - yref0 = yshift

    xyscale
        xrotation 0.0 / 180.0 yrotation = 0.0
        b = + /- xmag
        c = 0.0
        e = 0.0
        f = ymag
        a = xin0 - b * xref0 = xshift
        d = yin0 - f * yref0 = yshift

    rscale
        xrotation = rotation + 0 / 180, yrotation = rotation
        mag = xmag = ymag
        const = mag * mag
        b = mag * cos (xrotation)
        c = mag * sin (yrotation)
        e = -mag * sin (xrotation)
        f = mag * cos (yrotation)
        a = xin0 - b * xref0 - c * yref0 = xshift
        d = yin0 - e * xref0 - f * yref0 = yshift

    rxyscale
        xrotation = rotation + 0 / 180, yrotation = rotation
        const = xmag * ymag
        b = xmag * cos (xrotation)
        c = ymag * sin (yrotation)
        e = -xmag * sin (xrotation)
        f = ymag * cos (yrotation)
        a = xin0 - b * xref0 - c * yref0 = xshift
        d = yin0 - e * xref0 - f * yref0 = yshift
```

Parameters:

- *input*: Array of input coordinates. (Must be an Nx2 array).

- *ref*: Array of reference coordinates. (Must be an Nx2 array).
- *bbox*: The range of reference coordinates over which the computed coordinate transformation is valid. Must be convertible to a 4-length or 2x2 array of the form:

```
[xmin, ymin, xmax, ymax]
```

or:

```
[[xmin, ymin], [xmax, ymax]]
```

If the user is working in pixel units, these limits should normally be set to the values of the column and row limits of the reference image, e.g `[1.0, 1.0, 512.0, 512.0]` for a 512 x 512 image. The minimum and maximum values in *ref* input are used if *bbox* is `None` or any of its members are NaN.

- *fit_geometry*: The fitting geometry to be used. The options are the following:
  - "shift": *x* and *y* shifts are only fit.
  - "xyscale": *x* and *y* shifts and *x* and *y* magnification factors are fit. Axis flips are allowed for.
  - "rotate": *x* and *y* shifts and a rotation angle are fit. Axis flips are allowed for.
  - "rscale": *x* and *x* shifts, a magnification factor assumed to be the same in *x* and *y*, and a rotation angle are fit. Axis flips are allowed for.
  - "rxyscale": *x* and *y* shifts, *x* and *y* magnifications factors, and a rotation angle are fit. Axis flips are allowed for.
  - "general" (default): A polynomial of arbitrary order in *x* and *y* is fit. A linear term and a distortion term are computed separately. The linear term includes an *x* and *y* shift, an *x* and *y* scale factor, a rotation and a skew. Axis flips are also allowed for in the linear portion of the fit. The distortion term consists of a polynomial fit to the residuals of the linear term. By default the distortion term is set to zero.

  For all the fitting geometries except "general", no distortion term is fit, i.e. the *x* and *y* polynomial orders are assumed to be 2 and the cross term switches (*xyterms* and *yxterms*) are assumed to be "none", regardless of the values of the *xxorder*, *xyorder*, *xxterms*, *yxorder*, *yyorder* and *yxterms* parameters set by the user.
- *function*: The type of analytic surface to be fit. The options are the following.
  - "legendre": Legendre polynomials in *x* and *y*.
  - "chebyshev": Chebyshev polynomials in *x* and *y*.
  - "polynomial" (default): Power series in *x* and *y*.
- *xxorder*, *xyorder*, *yxorder*, *yyorder*: The order of the polynomials in *x* and *y* for the *x* and *y* fits respectively. The default order and cross term settings define the linear term in *x* and *y*, where the 6 coefficients can be interpreted in terms of an *x* and *y* shift, an *x* and *y* scale change, and rotations of the *x* and *y* axes. The "shift", "xyscale", "rotation", "rscale", and "rxyscale", fitting geometries assume that the polynomial order parameters are 2 regardless of the values set by the user. If any of the order parameters are higher than 2 and *fit_geometry* is "general", then a distortion surface is fit to the residuals from the linear portion of the fit.
- *xxterms*, *yxterms*: The options are:
  - "none": The individual polynomial terms contain powers of *x* or powers of *y* but not powers of both.
  - "half" (default): The individual polynomial terms contain powers of *x* and powers of *y*, whose maximum combined power is `max(xxorder - 1, xyorder - 1)` for the *x* fit and `max(yxorder - 1, yyorder - 1)` for the *y* fit.

–"full": The individual polynomial terms contain powers of *x* and powers of *y*, whose maximum combined power is `max(xxorder - 1 + xyorder - 1)` for the *x* fit and `max(yxorder - 1 + yyorder - 1)` for the *y* fit.

The "shift", "xyscale", "rotation", "rscale", and "rxyscale" fitting geometries, assume that the cross term switches are set to "none" regardless of the values set by the user. If either of the cross terms parameters are set to "half" or "full" and *fit_geometry* is "general" then a distortion surface is fit to the residuals from the linear portion of the fit.

•*maxiter* = 0: The maximum number of rejection iterations. The default is no rejection.

•*reject* = 3.0: The rejection limit in units of sigma.

Returns: A 2-tuple with the following parts:

•`GeomapResults` object, with the following attributes:

–*fit_geometry* str: The same value as *fit_geometry* passed to `geomap`.

–*function* str: The same value as *function* passed to `geomap`.

–*rms* (x, y) tuple: The root-mean-square of the residuals.

–*mean_ref* (x, y) tuple: The mean value of the reference coordinates.

–*mean_input* (x, y) tuple: The mean value of the input coordinates.

–*shift* (x, y) tuple: The translation of the fit.

–*mag* (x, y) tuple: The scale of the fit.

–*rotation* (x, y) tuple: The rotation of the fit.

–*xcoeff* double array: The first-order *x* coefficients of the fit.

–*ycoeff* double array: The first-order *y* coefficients of the fit.

–*x2coeff* double array: The second-order *x* coefficients of the fit.

–*y2coeff* double array: The second-order *y* coefficients of the fit.

•A Numpy structured array with the following columns:

–*input_x*

–*input_y*

–*ref_x*

–*ref_y*

–*fit_x*

–*fit_y*

–*resid_x*

–*resid_y*

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

stsci.stimage, 3

G

geomap() (in module stsci.stimage), 5

S

stsci.stimage (module), 3

X

xyxymatch() (in module stsci.stimage), 3