PySnack: ASTROALIGN

Rainer Schödel, IAA-CSIC, 14 Nov 2023

https://astroalign.quatrope.org/en/latest/index.html

Astroalign: A Python module for astronomical image registration

Show affiliations

Beroiz, M. ; Cabral, J. B. ; Sanchez, B.

We present an algorithm implemented in the Astroalign Python module for image registration in astronomy. Our module does not rely on WCS information and instead matches three-point asterisms (triangles) on the images to find the most accurate linear transformation between them. It is especially useful in the context of aligning images prior to stacking or performing difference image analysis. Astroalign can match images of different point-spread functions, seeing, and atmospheric conditions.

ASTROALIGN - what it does

- Python package
- Align images via stellar *asterisms*
- No WCS information necessary

Asterisms

- Works with lists of stars, images or both
- Very useful to
  - align images
  - find lists of corresponding stars

ASTROALIGN - What it does NOT do

ASTROALIGN does not provide any solution for WCS.
… for the latter, see astrometry.net

Astrometry.net

If you have astronomical imaging of the sky with celestial coordinates you do not know—or do not trust—then *Astrometry.net* is for you. Input an image and we'll give you back astrometric calibration meta-data, plus lists of known objects falling inside the field of view.

We have built this astrometric calibration service to create correct, standards-compliant astrometric meta-data for every useful astronomical image ever taken, past and future, in any state of archival disarray. We hope this will help organize, annotate and make searchable all the world's astronomical information.

Do not use ASTROALIGN for precision astrometry, but it can serve as a first step (identifying lists of stars for precision-alignment.

ASTROALIGN - How it works

1. Identify **brightest stars** in the images (max_control_points=50)

2. **Create triangles** from 4 nearest neighbours around each star

3. **Compute invariants** for triangles (L2/L1 and L1/L0, where L0, L1, L2 are the lengths of the triangles)

4. **Find corresponding triangles.** If a match is found, test transformation on the other triangles.

5. For accepted transforms, use **RANSAC* algorithm to re-calculate transformation.**

6. **Align images** using `warp` function from Scikit-IMage's `transformation` module.

**Random Sample Consensus (Fischler & Bolles, 1987)*