image_registration Documentation

Release 0.1

Adam Ginsburg

October 26, 2015

Contents

Github: Image Registration

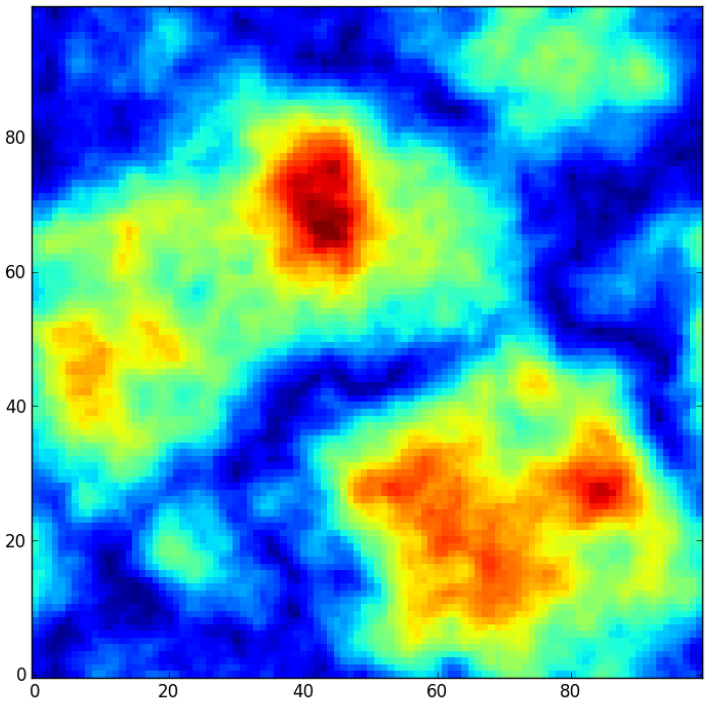
TL;DR version

Use `chi2_shift` to match images.

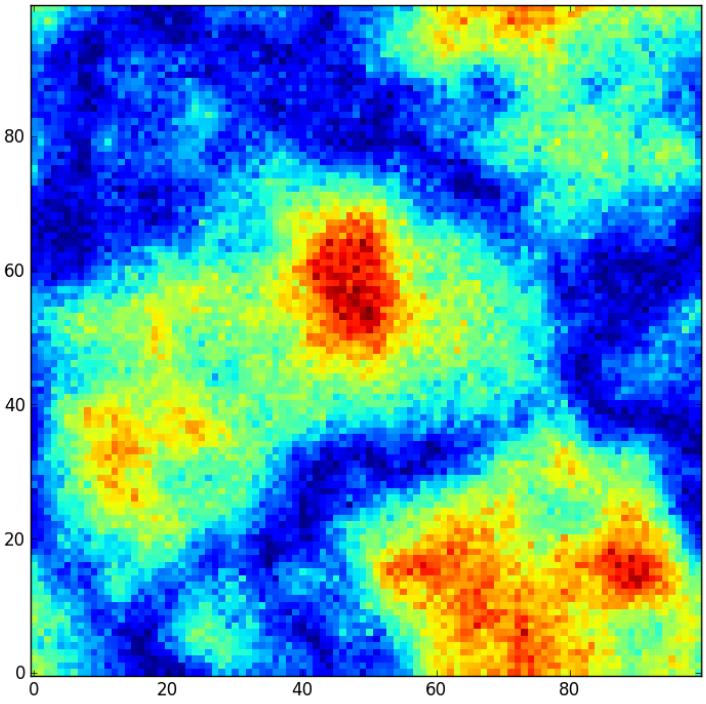
A toolkit for registering images of astronomical images containing primarily extended flux (e.g., nebulae, radio and millimeter maps). [1]

There are related packages scattered throughout the internet that do the same thing, but with different features.

The general goal is to align images that look kind of like these:



[1] Apparently astronomical images look a lot like microscopic images. So maybe this code is good for coaligning bacteria!


0
20
40
60
80
0
20
40
60
80

CHAPTER 1

Module APIs:

`image_registration`

`image_registration.fft_tools` (and a description of the `fourier_tools`)

`image_registration.tests`

The most successful of the methods implemented here is `chi2_shift`

There is an ipython notebook demonstration of the code here and in pdf here

CHAPTER 2

Related Methods

There are many other approaches to performing image registration. Some are summarized here. Note that this package is intended for image registration where the brightness is "extended" or "spread out" - stellar images are best to register by treating the stars as control points.

The methods below have various advantages and deficiencies. The most dangerous approach that should be avoided is that of fitting a gaussian to the peak of a cross-correlation image: this is the only other method that allows for measurements of the errors on the best-fit shift, but it is likely to be systematically wrong. The peak-fitting approach is unstable to fitting cross-correlated structure (which may be "beam-shaped") instead of the cross-correlation shift peak (which may have effectively no shape because it is sub-pixel).

The main advantage of the `chi2_shift` approach is that it can return *statistical errors* on the best-fit shift. It is also fast and efficient for many image types and sizes.

2.1 Programs implementing these methods in various languages:

Varosi + Landsman astrolib correl_optimize : Uses cross-correlation with "reduction" and "magnification" factors for speed and accuracy respectively; this method is relatively slow when using the complete information in the image (the magnification process increases the size of the image directly)

Brian Welsch's cross-cor taylor : Uses the cross-correlation peak to measure the pixel peak of the offset, then does a 2nd order taylor-expansion around that peak to achieve sub-pixel accuracy. Is fast and generally quite accurate, but can be subject to bias.

Manuel Guizar's Efficient Sub-Pixel Registration : A matlab version of the main method implemented in this code. Is fast and accurate. The speed comes from making use of the fourier zoom / fourier scaling property.

Marshall Perrin's sub-pixel image registration : Implements many cross-correlation based methods, with sub-pixel registration based off of centroiding, gaussian fitting, and many variations thereupon. The gaussian approach is also implemented here, but is highly biased and inaccurate in general. As a sidenote, I tried using the "gaussian fit after high-pass filter" approach, but found that it really didn't work - it helped remove SOME of the large-scale junk, but it didn't end up changing the shape of the peak in a helpful way.

Christoph Gohlke's python fft image registration code : Allows for rescaling and rotation.

Related bibliographic items (with no attached code): "Sub-pixel image registration with a maximum likelihood estimator" The method they describe is statistically equivalent to what I have implemented, though my method measures the *error* on the maximum-likelihood offset in addition to the ML offset.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search
- image_registration.FITS_tools
- image_registration.fft_tools
- image_registration
- image_registration.tests