User manual for Astrometry.net

Jonathan Gagné

Astrometry.net is a software suite written in C and Python, which can find astrometric solutions as much on scientific images (for example in the .fits format) than on those of amateur astronomers. In the particular case which should be of our interest, it can build a standard header, containing the coordinate transformation between the physical (X,Y) coordinates on the image and the equatorial coordinate system (RA,DEC) in J2000. The software can even compute SIP distortion polynomials, which are another standard in astronomy (they are understood by ds9, IRAF and the IDL astrolib library).

The beauty of this suite does not reside in the fact it has been written in Python, but in the fact it's robus and possesses prophetic gifts : you can give an image containing a field of stars without a header and without giving any more information and the code will find its astrometry with a very high success rate. Of course, you can optionally give some clues which can speed up the computation, like the size of your field, its approximate position, etc. The complete version of the code, which can run without any internet connection, can be installed on your personal computer for between 3 and 30 Gb of disk space (according to how you are gonna use it). For fulfilling its duty, the code relies on a combination of the USNO-B and 2MASS catalogs, containing in total more than one and a half billion sources, which makes (to me) nearly shamanesque its relatively low cost on disk space. Another great advantage with this software is that it will never give wrong solutions : if it can't find the astrometry, it will just tell you so. It never happened to me while using this code on several thousands of images and the paper cited at the bottom of this text says so.

However, you must not forget that making the astrometry of an image can be a really hard task (the ones who already tried to build such code and lost 10% of their capillarity at the same time can understand me). Thus, on some star fields which are very dense or nearly empty, it happens that the code can't find the solution. The time spent on an image can vary between half a second for easy fields on which you gave some clues, to up to 30 minutes for the hardest fields without any clue. Normally, if you use the code the right way on fields that are not too pathologic, 5 minutes should be sufficient.

You'll have guessed it, a Python and C library, this must be like hell to install on a non-linux machine. In fact it is not, you can even install it on Windows with a little more effort with the help of Cygwin. I didn't try to install it on Mac, but it's probably easier than on Windows. You can go and see the links at the end of this document to get help installing it.

The team who developed this series of codes share a “freeware” philosophy concerning astronomic images, that's why they will only give you some important files for the installation if you specify then personally, by mail, that you will not use their software for lucrative reasons. You'll be ok if you want to use this for OPIOMM or some scientific paper.

It is possible to call these routines directly through IDL, even if they were written in Python and C. It's what I have done in a code named “astrometrynet.pro” you can obtain through the CRAQ student workshop organizers. The package is a Zip file containing some astrolib routines you will need, along with subroutines from my personal library. You will have to modify “path_library.pro”, which is more like a configuration file specifying paths on your computer (all of this is explained and more in the code's header).

If you do not have to deal with a high number of images, a simpler solution (which is more “manual”) would be using the http://nova.astrometry.net/ website. You can upload your images there and some servers will run astrometry.net for you, giving in return a header containing all you should need. You can even link your Gmail account to the website through OpenID and then the website will keep your past jobs in memory. Despite the intention of the team to render this kind of information public, you have the option to set the images visible or not to anyone on the internet (on the home page of the website).

However, possibilities do not end here : you can even give .jpg images through Flickr and the software will annotate anything you like on the images, like a coordinate grid, constellations, objet names, “Long Cats”, etc.

This is all pretty nice, but how did they manage to build this nearly magic software library, that can guess so well and so fast your field's astrometric solution ? Well, their secret will be revealed here. When we try to guess a field's position in the sky, the greatest difficulty is linked to the fact we do not know a priori the orientation, the scale and the exact position of the scale. Not only this is a 6 parameters minimisation, but some stars are hussy and move in the sky. Thus, it will never be possible to find the exact solution, as certain stars will have moved (or even changed luminosity / disappeared). There can even be some spots on your CCD which look like “false stars”. More than this, there is always an inherent uncertainty in the stars position on your image, due to anything from the optics of your camera and telescope to the turbulent atmosphere. The question then asks itself : how do *you* do, when you try to recognize star patterns with your eyes ? What we naturally tend to do is trying to recognize “shapes”, or “micro-constellations” made of stars, which are alike both in your image and on the finding chart. This is the way astrometry.net works. More precisely, it begins by taking four stars which are sufficiently bright in your image. Then, the two most distant stars become the reference points, that is positions (0,0) and (1,1) in an orthogonal coordinate system. (This is a normal grid which underwend any linear transformation). The coordinates of the two remaining stars in this system are then the “name” of the constellation. The reason why this is so ingenious is that a constellation's name will remain the same, even if you rotate, stretch or make any linear transformation on your image. Thus, without knowing any of these values, you can begin recognizing where is your image on the sky. By doing this with a lot of micro-constellations (they call it “quads”), the code ends up converging towards a solution or a failure. Now, you can appreciate the project's logo.

Now it's time to explain why some ways of using the code asks for less disk space. For speeding up the execution time, the astrometry.net team already computed a great lot of quad's names (they call them “indexes”) from the 2MASS and USNO-B catalogs, which they gathered in different files, following the size of each constellation. For example, index file #200 contains all constellations which size lies between 2 and 2.8 arcminutes, etc. Thus, when you want to get the astrometry for let's say a 30 x 30 arcminutes field (I really chosed this randomly), you will only need index files for quads between 0.3 and 30 arcminutes.

Useful links :

- D.Lang, D.Hogg et al. paper on the project : http://adsabs.harvard.edu/abs/2010AJ....139.1782L
- Web site of the projet : http://astrometry.net/
- Web site where you can upload images : http://nova.astrometry.net/
- Informations for the installation : http://trac.astrometry.net/browser/trunk/src/astrometry/README
- User's forum for the project (if you have any questions !) : http://forum.astrometry.net/
- More informations on standard WCS linear transformations and SIP polynomials : http://tdc-www.harvard.edu/wcstools/SIP_distortion_v1_0.pdf