PYRAF TUTORIAL

W. E. KERZENDORF

1. Installing IRAF and PyRAF

Installing IRAF is normally a menial task. Fortunatley there are good instructions and tutorials for the main platforms.

1.1. **Mac OSX.** There is a very good package for both Intel and Power PC Macs available at: `http://www.macsingularity.com` On the site you can find the package and there are instructions on how to install it. Before Installing PyRAF one needs to install two extra packages from STSCI.

Download from the binary and source packages from: `http://www.stsci.edu/resources/software_hardware/stsdas/download` Create directories: `/iraf/extern/stsdas` and `/iraf/extern/tables` Change into `/iraf/extern/stsdas` and untar the source into the directory. bin directories will have be created. Change into `bin.macintel` and untar the STSDAS binary into the directory.
Repeat the same procedure for the Tables package. The last thing we need to do is enable both packages for IRAF. Edit the fle `/iraf/iraf/unix/hlib/extern.pkg`:

```
reset tables = /iraf/extern/tables/
task tables.pkg = tables$tables.cl

reset stsdas = /iraf/extern/stsdas/
task stsdas.pkg = stsdas$stsdas.cl

reset helpdb = "lib$helpdb.mip\
,noao$lib/helpdb.mip\
,tables$lib/helpdb.mip\
,stsdas$lib/helpdb.mip\
"
```

To be able to use the iraf graphics, add this `export PYRAF_WUTIL_USING_AQUA=1` to your `.bash_profile` in the home directory.

1.2. **Linux.** Instructions for installing IRAF on Ubuntu can be found at `http://mjhutchinson.com/journal/2006/11/05/install_iraf_on_ubuntu_edgy_amd64`. The instructions need to be adjusted a little bit to install the newest version of all the Software Tools. After installing IRAF you need to install PyRAF. Again the instructions need to be adjusted a little bit:
`http://mjhutchinson.com/journal/2006/11/14/adding_pyraf_to_iraf_on_ubuntu_edgy`

2. Interactive Data Reduction

Change to your IRAF directory and start PyRAF there with `pyraf` or the way I recommend it with `pyraf --ipython`.

The first thing we want to do is create a textfile with all the images we want to process. To do that simple pipe the ls command into a a file, like this: `files image?.fits > images.lis`. We need the `files`-task to get them in a single column listing.

Lists are really cool to work with you just put "@" in front of the file name in any task and it will iteratively process the files given in the list. We can even give the files a suffix or a prefix.

The next thing we want to do, is crop the files, subtract the bias and flatfield them. I will use the very versatile task `ccdproc` for that. First we need to load the package `ccdred` that contains the task by typing `imred` and then `ccdred`. We will have to change the task setup with the parameter editor `epar`:

`epar ccdproc`
input=@images.lis
output=fbt@images.lis
Set these process options:
overscan=yes
trim=yes
flatcor=yes
Set the parameters for the process options:
biassec=[1:50,1:2148]
trimsec=[500:1700,400:1700]
flat=flat (which will set the flatfield image to flat.fits)

This will create files: fbtimage[1-5].fits. Looking at the header of the processed images we can see that ccdproc added a log of what it did. Now our images are prereduced and we can start with the scripted photometry.

3. Scripting

The images are of a standard star field. Standard stars are stars with known location and magnitude. We use these to calibrate the photometric measurement of the science objects we took on the night.

Multiple images are taken for various reasons. They are dithered to avoid stars falling on the same pixels, in case they are bad. Cosmic rays are another issue.

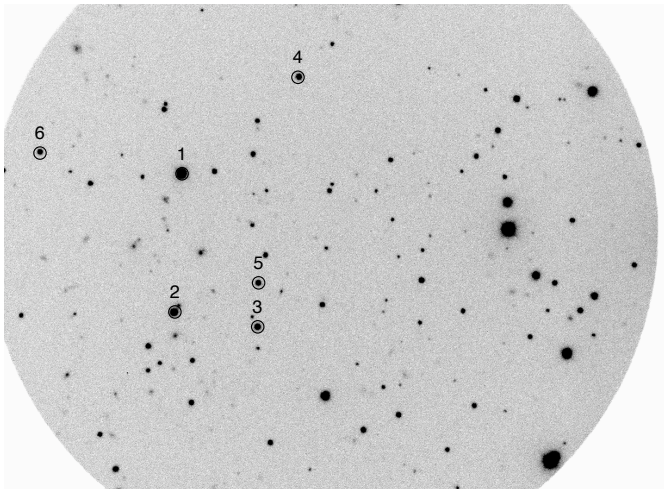
We could now go into each image and find the standard stars and do photometry on them. This is the manual and hard way of doing things. What we will do is find out the shift using one or two prominent stars and calculate the shifts between the first and all other images. Then we will identify the standard stars in the first image and shift their coordinates to matching coordinates for each image. Then we will just do photometry on each image and use the mean measurement for each star. Sounds simple enough well it is:

I have created some functions for this exercise in the `imutil.py`. It is easier to use that with PyRAF in ipython mode than normal PyRAF. Run `pyraf --ipython` and

use the run macro with the -i flag to load the functions to be used in interactive mode (like this `run -i imutil.py`).

The first thing we will do is run the `createShift` function. The function needs the prefix of the files and how many there are, in this case `createShift('fbtimage',5)`. This will create a file called `shift.dat` which has the necessary information to create individual coordinate files.

Now we need to find our standard stars in the image. This star chart should help:



Create regions in DS9 and save them (with the x y format) to a file. So we have a shifts file and a coordinate file for the first image. The function will create coordinates for all images. To run it: `shiftCoord('fbtimage',5,'shift.dat','ds9.reg')`.

We now have fbtimage[1-5].coo files. Now we can do the photometry automatically on all of them. The last function is a bit more complex because we will use more iraf functions and also extract the photometry out of a special IRAF format in a space seperated format:
`phot=doPhot('fbtimage',5)` This will put an array of magnitudes into the variable `phot`.

This is an easy example to automate a normally menial task. You can expand these simple tasks and write a fully automated pipeline for your data.

The last thing we want to do is to find the zeropoint and colour term for our measurement:

We will plot the zeropoint ($m_{apar} - m_{instr}$) against the colour of the stars. The slope will give us the colour term. The following table will are the standard stars and the colours:

Table 1.

ID	RA	Dec	R	V-R	R-I
1	06:42:54.67	-45:08:34.70	13.74	0.36	0.34
2	06:42:55.03	-45:09:56.80	15.2	0.49	0.42
3	06:42:50.37	-45:10:05.50	15.97	0.3	0.3
4	06:42:48.14	-45:07:37.00	16.44	0.63	0.52
5	06:42:50.34	-45:09:39.30	16.56	0.53	0.49
6	06:43:02.53	-45:08:22.50	16.63	0.4	0.41

4. Fitting a WCS

Once you have installed PyWCS you can just type `addwcs <image> <catalogue>`. To get more options type `addwcs --help`.