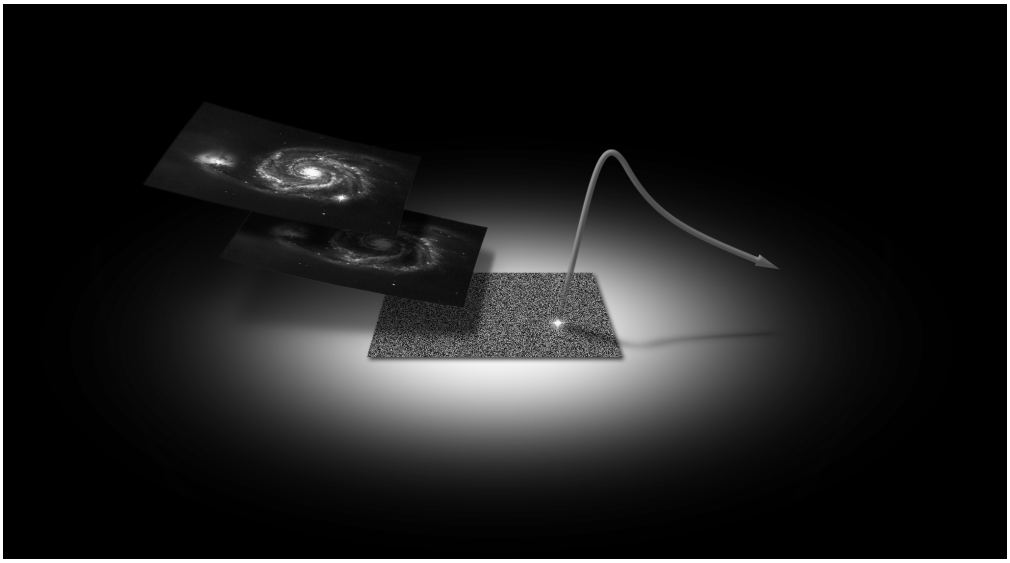
CLASP

Create Lightcurves with Alignment, Subtraction and Photometry

Joe Lyman*
Astrophysics Research Institute
Liverpool John Moores University

December 3, 2012



*jdl@astro.livjm.ac.uk

Abstract

A user manual of sorts for *CLASP*

1 Quick Use

The program consists of two pipelines, called from the scripts `run-subpipe.py` (image alignment/subtraction) and `run-photpipe.py` (photometry), or via the GUI - `CLASP.py`. Below are examples how to run them.

NB: Even if you're in a hurry, prior to performing these, look in `PIPEcfg.py` and `ISIScfg.py` and change the parameters in there to suit your data (see 4.1), otherwise you may run into problems.

Running from command line (including required arguments):

```
$ run-subpipe.py imagedir template workdir

$ run-photpipe.py workdir
```

Help on command line arguments:

```
$ run-subpipe.py -h

$ run-photpipe.py -h
```

Example use of parameters (including all possible optional arguments):

```
$ run-subpipe.py /path/to/imagedir /path/to/template.fits
/path/to/workdir -s "*.fits" -u -f /path/to/fringeframe.fits
-b /path/to/bpm.fits -c -ti 1 -ii 1 -t 5  -r 1
-stamps /path/to/mystamps.txt -d

$ run-photpipe.py /path/to/workdir -c -n -sa 3 -sl 15 -o
"123,456" -l myphotlogname.txt -d
```

(Note the string quotes around the selection and object coordinates arguments, `-s` and `-o`.)

For more in-depth look at usage as well as more details about the configuration files, see section 4.

2 Introduction

This section just details the form of the pipelines. Skip forward to section 4 to see how to actually run the pipelines.

CLASP (*C*reate *L*ightcurves with *A*lignment *S*ubtraction and *P*hotometry) is a program consisting of two pipelines (Subtraction and Photometry) to automate transient data reduction and lightcurve creation. Data specific requirements have been kept to a minimum, barring some FITS headers, in order for the program to accept a wide range of imaging data from any CCD imager.

The program is written in the Python[1] environment of IRAF[2] afforded by Pyraf[3], also using the NumPy[4] and SciPy[5] Python packages. (See section 6 for further details.)

For an overview of what each pipeline does, see section 3. For usage information and how to call the pipelines, see section 4. For requirements, see section 6.

[1] `http://www.python.org/`
[2] `http://iraf.noao.edu/`
[3] `http://www.stsci.edu/institute/software_hardware/pyraf`
[4] `http://numpy.scipy.org/`
[5] `http://www.scipy.org/`

3 Pipelines

This section describes in plain English the main stages of each pipeline; what they expect of the data, what they do to the data, and what is output (see section 5 for a brief overview of what files are output and their contents).

3.1 The Subtraction Pipeline

In a nutshell:

- Create a working directory
- Copy Template and Science image to work directory plus some config files
- Create report and log files in work directory
- Clean the Template if required
- Clean the Science image
- Align images using methods requested by the config file
- Determine seeing in both images and convolve better seeing image PSF to match worse seeing image PSF
- Subtract Template from Science image
- Make PNG images of the Aligned and Subtracted images
- Fill in report line with diagnostics of how the subtraction went
- Rinse and repeat for all Science images

Out of the nutshell to do...

3.2 The Photometry Pipeline

In a nutshell:

- Create report, log and lightcurve files in work directory
- Parse the provided image coordinates of the object to measure, or display the image for the user to select the object if not provided
- Read the star centres of stars to use for photometry if provided, else display the Template and ask user to select them
- Perform small aperture photometry of the object in each Science image
- Perform aperture correction photometry of the previously selected stars in the Template and each Science image

- Calculate an aperture correction for each image, and an offset to the Template for each Science image
- Apply the aperture correction and offset to the small aperture object magnitude per Science image
- Write the final, corrected magnitudes to the lightcurve file
- Fill in report with various values deduced (i.e. offset to template etc.)

Out of the nutshell to do...

4 Usage

The *CLASP* pipelines can be run from either the command line or GUI and these methods for these are detailed below. **However**, there are two configuration files that need to be set properly for your data before you begin running the pipes. Most unsuccessful attempts at running the pipes can probably be traced back to incorrect configuration files.

4.1 The configuration files

The two configuration files are named `PIPEcfg.py` and `ISIScfg.py`, located in the main directory. These files hold parameters that may need to be tweaked depending on the data you are feeding the pipelines. Comments in these files should explain most of the parameters.

`ISIScfg.py` holds all the parameters for the ISIS code and in most cases directly translate with the same parameter names (see the documentation for ISIS[6] for more technical details on these). Particular attention should be paid to the saturation levels and the minimum value to fit/miniumum value for a stamp (a source to use in psf matching). In images with only a few point sources, it may be prudent to reduce `kernelorder` to 1 as the kernel form can vary wildly if allowed to without enough constraints (i.e. enough stamps sampled across the entire image).

`PIPEcfg.py` holds data dependant parameters. Some of these relate to the names of FITS headers which should be self-explanatory. Pay particular attention, however, to the parameters defining the min/max number of objects to find in the image(s)/template. The pipelines run `iraf.daofind` to detect objects at the given thresholds (initially `IMAGETHRESH` and `TEMPTHRESH`). This is adjusted to meet the min/max object requirements (up to a limit). A judge by eye of the number of sources in your image should let you set these sensibly to broadly enclose this estimate and prevent the pipeline spend too long trying to meet your number of object requirements. The `WREGISTER` and `XYXYMATCH` are two `iraf` image registration methods and can be switched on and off as required. Typically using `WREGISTER` can be good to quickly get a rough alignment if the image(s) and template are severely rotated/scaled/translated from each other (such that `XYXYMATCH` may struggle to find a solution) and then using `XYXYMATCH` on this roughly aligned image. If your image(s) and template are only slightly misaligned, and a large fraction of the sources are common to both images, using just `XYXYMATCH` is preferred.

4.2 From the Command Line

The two pipelines can be called from the terminal, general command line argument help is obtained via: Help on command line arguments:

```
$ run-subpipe.py -h
```

[6] `http://www2.iap.fr/users/alard/package.html`

```
$ run-photpipe.py -h
```

4.2.1 run-subpipe.py

Required arguments:

`imagedir`: the path to a directory containing the Science image(s) to be fed to the pipes. Alternatively set to the file path of a single file to run the pipeline on one image. Data selected will not be altered as all image(s) to be processed are copied to the work directory.

`template`: the file path to the Template file. Again this file will not be affected and will be copied to the work directory in a subfolder, 'template'.

`workdir`: the path to a directory to store all the pipeline's output. If the path doesn't exist then it will be created. If the directory exists, the pipeline will ask whether you want to remove it, in which case *everything contained within* `workdir` *will be deleted!*

Optional arguments:

`-s SELECTION`: the pattern for sub-selecting files within the image directory as the Science image(s). Note that `SELECTION` must be enclosed in string quotes since it will probably contain some kind of wildcard. `Default = '*.fits'`.

`-u`: including this flag will run the subtraction pipeline in 'update' mode. In this case an existing `workdir` must be specified containing previous subtraction pipeline output. The log and report files will be appended to rather than overwritten. This is useful should a small subset of images have poor subtraction for example. By tweaking the appropriate parameters in `PIPEcfg.py` and `ISIScfg.py` and rerunning `run-subpipe.py` with `SELECTION` to just pick the troublesome images (and including `-u`), these will be overwritten with the new pipeline output. Similarly should you wish to add observations to an existing directory of output, these will be added alongside without destroying previous work.

`-f FRINGEFRAME`: the file path to the appropriate fringe frame for the Science image(s). Should be a single extension FITS file of the same dimensions as the Science image(s).

`-b BADPIXELMASK`: the file path to the appropriate bad pixel mask for the Science image(s).Should be a single extension FITS file of the same dimensions as the Science image(s) (if not, it will be ignored). Should be zero at all good pixels and non-zero to indicate bad pixels.

`-c`: including this flag will mark the Template for cleaning, i.e. defringing, bad pixel removal (as supplied by `-f` and `-b`).

`-ti TEMP_ITER`: the number of cosmic ray cleaning iterations to perform on the Template. Additional passes may find more cosmic rays not found by the first pass but generally one or two will remove the bulk of them. Requires some work as not performing well on some data (meaning it fails to find cosmic rays rather than erroneously thinking true source pixels are cosmic rays as far as I can see, so is most likely to be safe). Set to 0 to skip cosmic ray cleaning.

`-ii IMAGE_ITER`: as above but for the Science image(s).

`-t TRIM`: an integer value denoting the number of pixels to fix around the edge of the image. These pixels will be fixed to have a value zero. This can mask any instrumental effects at the edges of the image that may look like cosmic rays hits to the removal algorithm and will cause significant slowing of the pipeline.

`-r REVERSEFLAG`: an integer specifying the desired direction of convolving prior to subtraction. `0` - the program attempts to determine the best direction, `1` reverses the programs decision, `2` always convolves the Science image, `3` always convolves the Template. See section 3.1 for further information.

`-stamps STAMPS`: the file path to a file containing stamp centres to be used by ISIS. The file should have `x` and `y` pixel coordinates in the first two columns, separated by white space.

`-d`: include this flag to run verbosely to stdout. The log file will always have full verbosity but this flag signifies to output to the terminal verbosely also.

4.2.2 run-photpipe.py

Required arguments:

`workdir`: the path to the directory containing the output from the subtraction pipeline, upon which photometry is to be performed. In between calling `run-subpipe.py` and `run-photpipe.py`, it is best to leave this work directory alone (i.e. don't go renaming it or moving files in/out of it) else no doubt something will faulter!

Optional arguments:

`-c`: including this flag will clobber the existing lightcurve file in `workdir`. Use this to redo photometry if something went wrong. If a lightcurve file exists and this flag isn't present, then the pipeline will complain.

`-sa SMALLAP`: an integer value denoting the radius of the aperture (in pixels) to use for small aperture photometry. This is the aperture used around the object before applying an aperture correction to `LARGEAP`.

`-la LARGEAP`: an integer value denoting the radius of the aperture (in pixels) to correct the object photometry to. Photometry will be performed on aperture correction stars in steps between `SMALLAP` and `LARGEAP` to compute the curve of growth.

`-o OBJCOORDS`: a string of the form `"x,y"` denoting the x and y pixel positions of the object. Centering will be applied to these so high accuracy at this stage is not required. If omitted a subtracted image will be displayed with instructions on how to provide the pipeline with the objects coordinates.

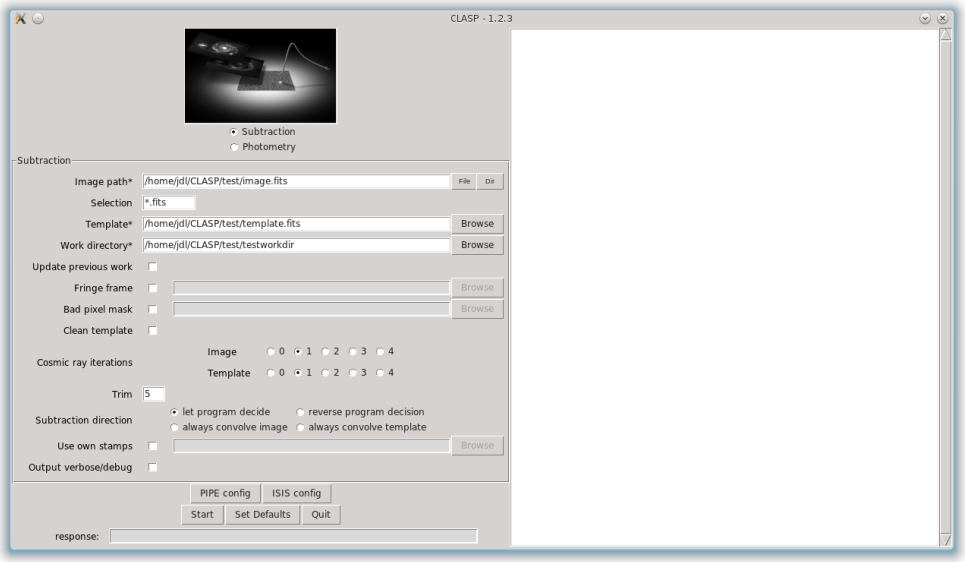
`-n`: specifying this flag means to ignore a previous list of stars that were used for aperture/offset corrections. E.g. if clobbering over a previous lightcurve and you wish to select a new list of stars, use this flag. See section 3.2 for information about how these stars are selected in the first place.

`-l LOGNAME`: the name of the file desired to hold the logged output from the photometry pipeline. This will be created in `workdir`

`-d`: include this flag to run verbosely to stdout. The log file will always have full verbosity but this flag signifies to output to the terminal verbosely also.

4.3 GUI

Optionally, either pipeline can be called from the gui named `CLASP.py`. Running this file will open a window as below with input boxes for the various parameters and a box on the right to display the pipeline output.



Choose subtraction or photometry using the radio buttons near the logo. The resultant boxes relate directly to the arguments explained above and should be self-explanatory.

If browsing for an Image Path, choose 'File' to select a single image or 'Dir' to specify you want to use a directory of images (Python won't let me combine these two options into one browser easily).

The 'PIPE config' and 'ISIS config' buttons will open the configuration files for editing prior to running the pipeline.

Once the boxes have been filled as desired, pressing the 'Start' button will call the selected pipe. Output will be show on the right, so this may include errors which cause the pipeline to cancel (e.g. your Template doesn't exist). A new call can be made after altering the parameters by clicking 'Start' again. Once the pipeline is running, 'Start' will be replaced by a 'STOP (term)' button, which, if pressed, will send the `SIGTERM` command to the pipeline, if this doesn't stop it a `SIGKILL` command can be sent by subsequent clicks, effectively ending the pipeline if desired.

Should the pipeline require input (e.g. your specified `workdir` exists when trying to run the subtraction pipeline and the program wants to know whether to overwrite it), then it will ask for it in the right hand output box. You can interact with the pipeline via the 'response' entry bar at the bottom left. Type what you wish to send to the pipeline (e.g. 'y' for 'yes, remove the existing work directory') into this followed by the Return or Enter key to send this to the pipeline.

5 Output

Here is an explanation of the many (sorry about that) output files you will see appear in your work directory and what they contain. *(template)* is the Template filename minus the extension, *(image)* is the Science image filename minus the extension, and there will be one of these files in the working directory for every input Science image.

5.1 subpipe

After a run-subpipe.py call:

template A directory containing:

***(template)*.coo** Output from the `iraf.daofind` run on the Template

***(template)*.fits** Cleaned Template (fringing, bad pixels and cosmic rays as supplied)

***(template)*.fits.stars** Output from the Sextractor run on the Template

***(image)*.alardout** Output from the ISIS code

***(image)*.coo** Output from the `iraf.daofind` run on the image

***(image)*.fits** Cleaned image (fringing, bad pixels and cosmic rays as supplied) aligned to the Template

***(image)*.fits.stars** Output from the Sextractor run on the image

***(image)*.png** A quicklook png of the image (note scaling needs to be worked out as they levels are a bit naff currently)

***(image)*.sub.fits** The Subtracted image that has the aligned image header copied to it

***(image)*.sub.png** As for the png above but for the Subtracted image

***(image)*.sum_kernel** The value of the 'sum kernel' found by ISIS, basically a scaling factor between the two images

ISIScfg.py A copy of the ISIScfg.py file as it existed when the pipeline was called, this makes it easier to see what parameters were used when returning to/repeating workdir

pipe.shelve A shelve file used by the photometry pipeline to read in all the parameters of the subtraction pipeline as it was run as well as some diagnostics of the images used

PIPEcfg.py As above for ISIScfg.py but for PIPEcfg.py

subpipe_log.txt The output from the subtraction pipeline with verbosity turned on

subpipe_report.txt A breakdown of interesting values found for the Template and Image(s) as well as the outcome of the subtraction for each image. Good to quickly see if anything went wrong and with what image

5.2 photpipe

Additionally, after a run-photpipe.py call:

TODO TODO

5.3 temporary files

The following files can also be created by the pipelines, and will be cleared from the current working directory if they exist when the pipeline ends[7], so ensure there's nothing you want to retain named as any of the following:

`objmask.fits, conv.fits, conv0.fits, default_config,`
`kernel_table, toto.bmp, imxymatch.1, match.coo, geomap.db,`
`wregister.db, kernel_coeff0, STAMPS, tmp*.fits, den*.fits`

Ensuring these are created in the `/tmp` directory to prevent any filename issues, is a todo.

[7] Usually these are cleaned by the pipeline during its normal operation, but additionally if the pipeline is terminated abnormally, it will attempt to remove these files to prevent clutter

6 Requirements

CLASP was built, and thus has been tested on:

- Python 2.7, 2.7.3
- Pyraf 1.10, 1.11
- IRAF 2.14, 2.15.1a
- Numpy 1.6.1
- Scipy 0.9.0

I have little clue about what versions will/will not work. Python definitely needs to be 2.x (most likely 2.7+) and not 3.x and IRAF will probably need to be 2.x.x.

The program has a version of the ISIS code and Sextractor packaged with it and will call on these whether they are installed or not elsewhere on your machine.