Automated SpectroPhotometric Image REDuction

Marco C Lam*, **Robert J Smith & Iain A Steele**

Astrophysics Research Institute, Liverpool John Moores University, 146 Brownlow Hill, Liverpool, UK L3 5RF

*** c.y.lam@ljmu.ac.uk**

Liverpool John Moores University

Liverpool Telescope

STFC Newton Fund

OPTICON Horizon 2020

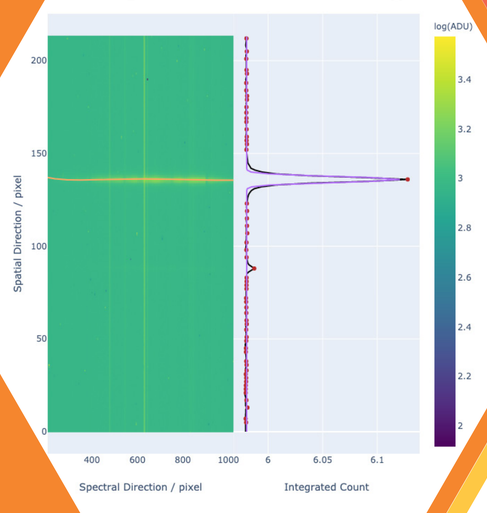
Background

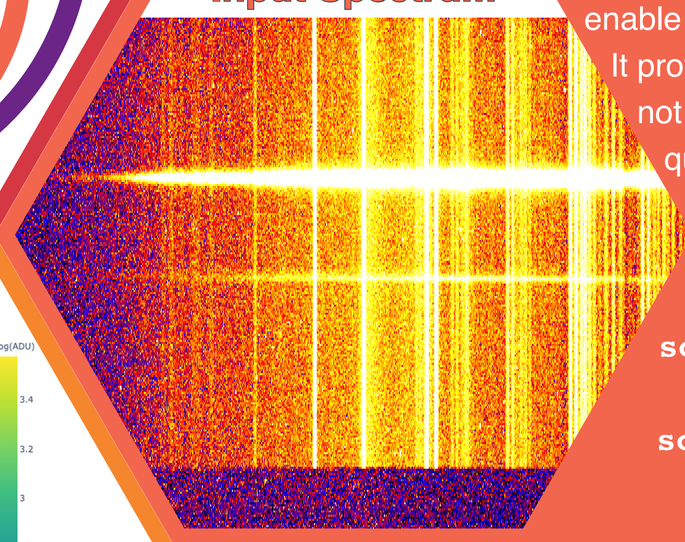
Time-domain Astrophysics is entering its golden age with a number of discovery telescopes coming online, generating high quality data with high cadence in huge volume. Rapid follow-up of various transient objects found from these surveys are essential to provide crucial astrophysical interpretations. As part of the European Commission Horizon 2020, the work package WP13 of OPTICON aims to develop and provide a suite of publicly available long-slit spectral data reduction software to facilite rapid scientific output.

ASPIRED is a low/medium resolution spectral data reduction software written in Python 3, part of the three concurrent developments with the GUI gASPIRED and the wavelength calibrator RASCAL (Poster P10-37). ASPIRED provides a scripting mode of data processing; the gASPIRED sits at a high level, but sharing the same plotting library powered by Plotly. Interactivity is enabled with JS9 and JQuery running in an Electron application. It allows **cross-platform** development with a **single codebase** for **Linux, Mac** and **Windows**.

TwoDSpec is an object for 2D spectral image manipulation, it performs spectral identification, aperture tracing and aperture extraction. Spectral tracing uses spline or polynomial fit. Multiple spectra can be found and traced simultaneously. Example for the simple codes that users are required to use:

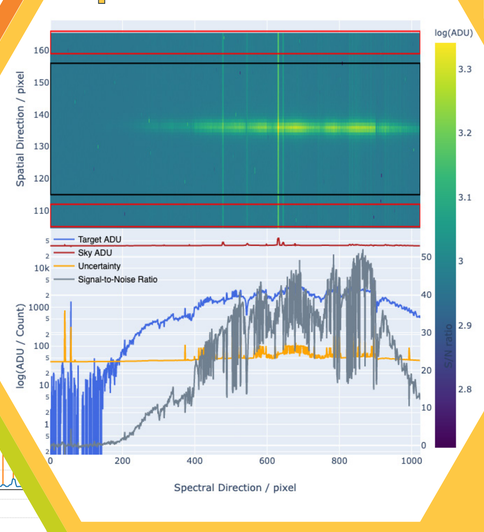
```
science2D = TwoDSpec(science.data)
science2D.ap_trace(display=True)
```





Basic **ImageReduction** routines are available to enable rapid data processing with a single interface. It provides arithmetic level of field-flattening. It is not the intention of this software to provide high quality flatfielding routines, for example, image rotation and fringe removal. These functions are not in the current development plan.

```
science = ImageReduction(
    'file.list')
science.reduce()
```

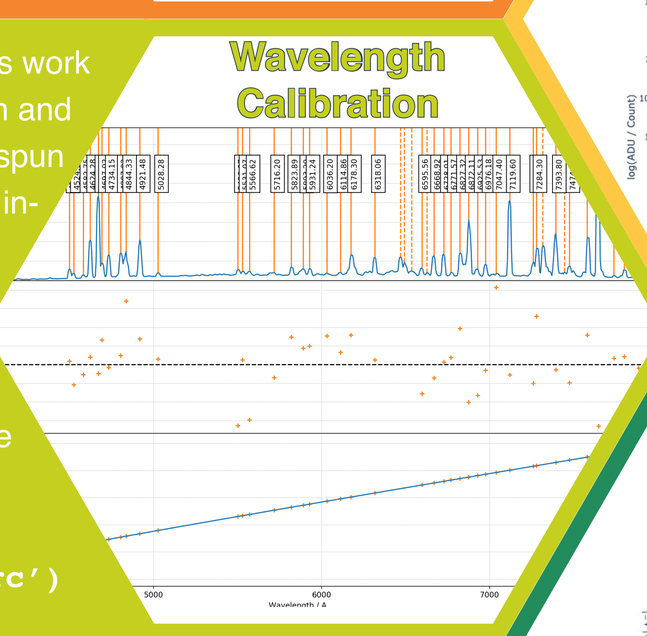


The traced spectrum is stored as a property of the TwoDSpec object. Currently, only the brightest spec trum will be extracted although multiple spectra can be traced simultaneously. An optimal or aperture extraction can be perform along the trace. The black box indicates the region where the signal is extracted; the red boxes are the regions used for fitting the background flux. The extraction contains the spectrum, uncertainty, sky and the signal-to-noise ratio.

```
science2D.ap_extract(display=True)
```

Wavelength calibration is a difficult task. In this work package of automated spectral data extraction and calibration, the entire calibration process has spun off to a separate project to tackle the problem independently. It is designed to be a completely stand-alone module that provides a suite of API in a similar style. The figure on the right shows the preliminary diagnostic plot for the calibration. Please see poster P10-37 for more information.

```
wavecal = RASCAL.calibrator.fit('arc')
```

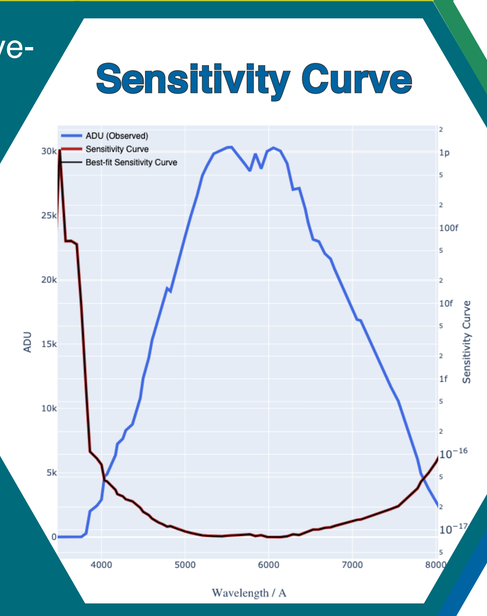


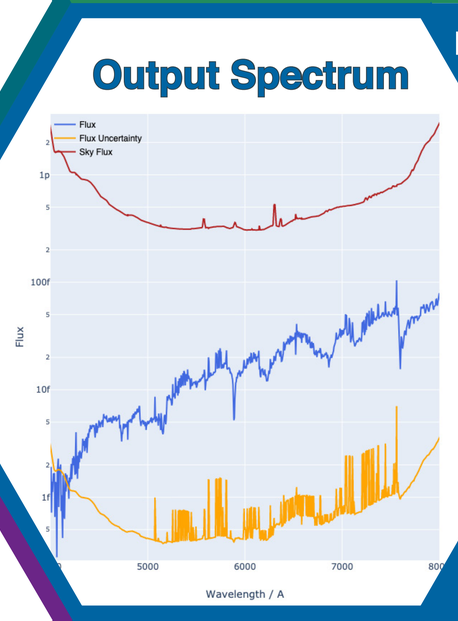
Standard stars available from iraf and ESO webpage are all stored in the software. They can be retrieved with the **StandardFlux** object by passing the name of the standard star. A basic regex algorithm will prompt users with the closest match if there is not an exact match.

```
fluxcal = StandardFlux(
    target='hiltner102',
    group='irafirs')
fluxcal.load_standard()
fluxcal.inspect_standard()
```

A **OneDSpec** object can be created with the wavelength calibration objects, the TwoDSpec objects of the target and the standard stars and StandardFlux object. This creates a sensitivity curve if the standard star is provided.

```
spec1D = OneDSpec(
    science2D, wavecal, standard2D,
    wavecal_standard, flux_cal)
spec1D.apply_wavelength_calibration()
spec1D.compute_sencurve(display=True)
```





By applying the sensitivity curve to the spectrum, the final flux and wavelength calibrated spectra of the target, standard, sky signals and uncertainties are produced. Further 1D spectral data manipulation and analysis will require external packages. The example spectrum shown here is from the low resolution spectrograph SPRAT on the Liverpool Telescope. The star is a dM1.75 main sequence star with a sub-solar metallicity.

```
spec1D.apply_flux_calibration()
spec1D.inspect_reduced_spectrum()
```

gASPIRED, being the graphical plugin of the software, is built with Electron, chosen for its popularity in the industry compared to the equivalent ones for Python. A medium to long term development of this major dependency is guaranteed. The interactive spectrum identification is handled with the JS9 display, while the rest are done with jQuery. The visualisation is done with Python-plotly, and it sends a JSON-string to be rendered by plotly.js and display on Electron.

Double Click the Application Icon

This software has made use of:

GitHub, Travis CI, Bootstrap, The Astropy Project, plotly, SciPy, JS, jQuery, NumPy, python, ELECTRON

RASCAL, SpectRes, JS9, AstroSCRAPPY, ccdprocs, npm, electron-compile, setuptools & their dependencies