




PyLongslit: a simple manual Python pipeline for processing of astronomical long-slit spectra recorded with CCD detectors

Kostas Valeckas ^{1,4}, Johan Peter Uldall Fynbo ², Jens-Kristian Krogager ³, and Kasper Elm Heintz ²

1 Niels Bohr Institute, Copenhagen University, Denmark **2** Cosmic Dawn Center, Niels Bohr Institute, Copenhagen University, Denmark **3** Centre de Recherche Astrophysique de Lyon, France **4** Nordic Optical Telescope, Spain

DOI: [10.21105/joss.09264](https://doi.org/10.21105/joss.09264)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Warrick Ball](#)  

Reviewers:

- [@Gabriel-p](#)
- [@kbwestfall](#)

Submitted: 01 April 2025

Published: 16 December 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

We present a new Python package for processing data from astronomical long-slit spectroscopy observations recorded with CCD detectors.

The software is designed to aim for simplicity, manual execution, transparency and robustness. The goal for the software is to provide a manual and simple counterpart to the well-established semi-automated and automated pipelines. The intended use cases are teaching and cases where automated pipelines fail. For further elaboration, please see the [Statement of need](#).

From raw data, the software can produce the following output:

- A calibrated 2D spectrum in counts and wavelength for every detector pixel.
- A 1D spectrum extracted from the 2D spectrum in counts per wavelength (for point-like objects).
- A flux-calibrated 1D spectrum in $\frac{\text{erg}}{\text{s}\cdot\text{cm}^2\cdot\text{\AA}}$ (for point-like objects).

The products are obtained by performing standard procedures for detector calibrations ([Howell, 2006](#); [Richard Berry, 2005](#)), cosmic-ray subtraction ([Dokkum, 2001](#); [McCully et al., 2018](#)), and 1D spectrum extraction ([Bradley et al., 2024](#); [Horne, 1986](#)).

Statement of need

A natural approach when developing data processing pipelines is to seek for precision and automation. The trade off for this is code complexity and “black-box” solutions, where the process of the pipeline is often masked, and the quality assessment output is made under the assumption that the user knows how to interpret it. In research, this is a reasonable trade off, as a certain level of user skill and experience can be assumed. However, in a teaching paradigm, simplicity and transparency are often more favorable, even when this means loss of precision and automation. The PyLongslit pipeline is designed to rely on simple code and manual execution, supported by a large array of quality assessment plots and extensive documentation. The algorithms are designed to produce research-quality results, yet while prioritizing simplicity over high precision. The reason for this is to create a robust and transparent pipeline, where every step of the execution is visualized and explained. We see this as being especially valuable in teaching scenarios and for users who are new to spectroscopic data processing. Furthermore, we hope that the simple coding style will invite users of all skill levels to contribute to the code.

An early beta version of the software was user tested during the [Nordic Optical Telescope IDA summer-course 2024](#), where all student groups were able to follow the documentation and

successfully process data without any significant assistance.

During the development of software it became apparent that the manual nature of the pipeline is also useful for observations where automated pipelines might fail. The PyLongslit software can revert to manual methods instead of using mathematical modelling when estimating the observed object trace on the detector. This is especially useful for objects that have low signal-to-noise ratio, or where several objects are very close to each other on the detector. Furthermore, extraction can be performed with either optimal extraction methods (Horne, 1986), or by summing detector counts for a box-like object shape (Bradley et al., 2024), which can be useful for emission-line-dominated objects.

Pipeline

The figures below describe the pipeline structure. The pipeline architecture is inspired by the very popular (but no longer maintained) IRAF (Tody, 1986). In a broad sense, there are three stages of the data processing, all explained in separate figures. The diamond shapes in the figures represent different pipeline routines that are called directly from the command line. Solid arrows are hard dependencies (must have), dashed arrows are soft dependencies (can use) and the rectangles represent input files and pipeline products.

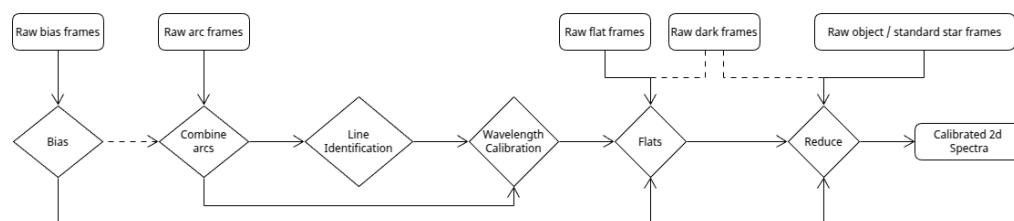


Figure 1: Step 1 - processing raw data. In this step, all the raw observation and calibration frames are used to construct calibrated 2D spectra. After this step, all procedures are performed directly on the calibrated 2D spectra, and the raw frames are no longer used.

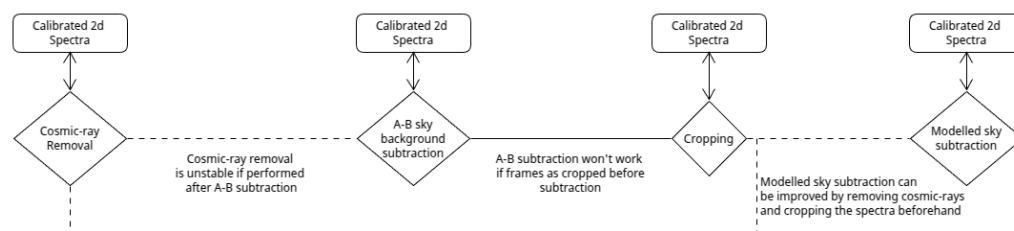


Figure 2: Step 2 - further processing of the calibrated 2D spectra. In this step, the user can deploy cosmic-ray removal, sky background subtraction, and crop the spectra. All procedures alter the 2D spectra in place. All of the steps are optional, but there are some dependencies — these are described in the figure.

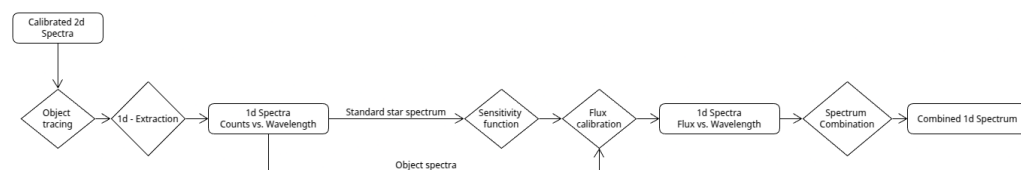


Figure 3: Step 3 - 1d spectrum extraction. In this step, objects are traced, extracted, flux-calibrated and combined (if several spectra of the same object exist).

The software is controlled by a configuration file that has to be passed as an argument to every pipeline procedure. The different parameters of the configuration file are described in the [documentation](#).

Evaluation

To test the software for correctness, we run the pipeline on data from two long-slit instruments, [NOT ALFOSC](#) and [GTC OSIRIS](#), and compare the results with the results from the well-established, semi-automated [Pypelt Python pipeline](#) ([Prochaska, Hennawi, Westfall, et al., 2020](#); [Prochaska, Hennawi, Cooke, et al., 2020](#); version 1.17.1):

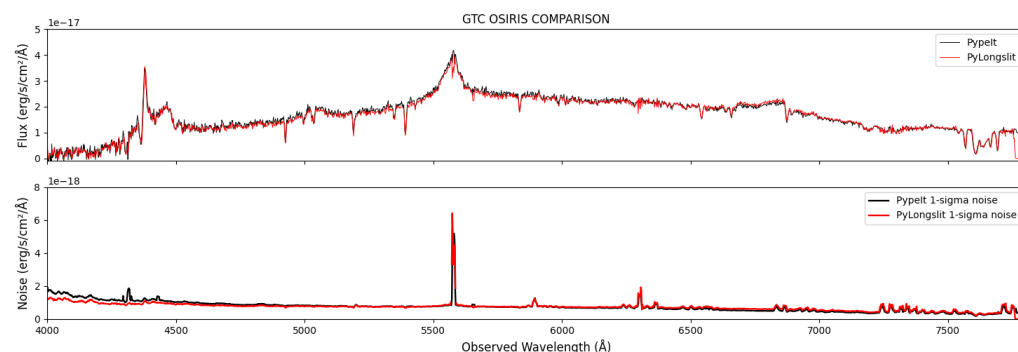


Figure 4: GTC OSIRIS observation of GQ1218+0823.

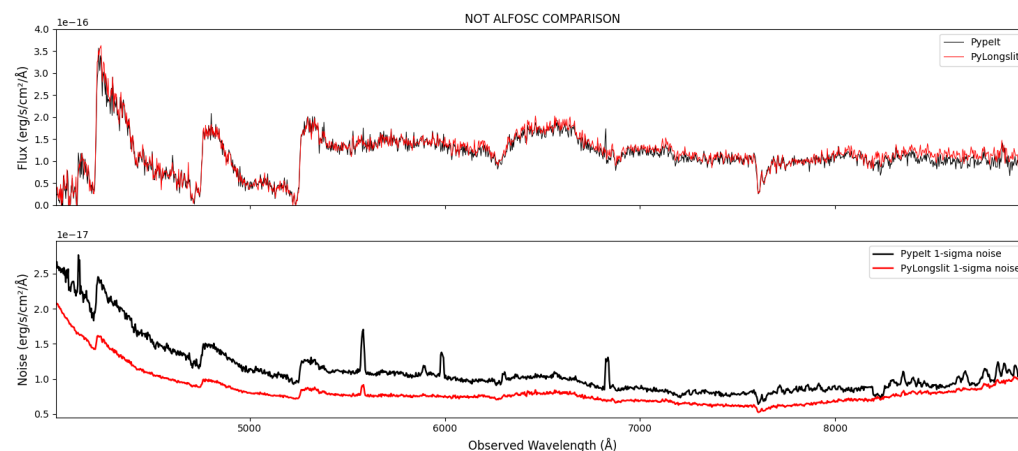


Figure 5: NOT ALFOSC observation of SDSS_J213510+2728.

We see good agreement between the two pipelines on both axes, for both the extracted

spectrum and the noise estimation. For NOT ALFOSC data, we see some deviation in the error magnitude and error related to strong sky lines. This is due to skipping modelled sky subtraction in the PyLongslit run, as the A-B sky background subtraction was sufficient by itself. We calculate the noise numerically for a cut of the spectrum where the flux is somewhat constant to confirm that the PyLongslit noise indeed is smaller in magnitude:

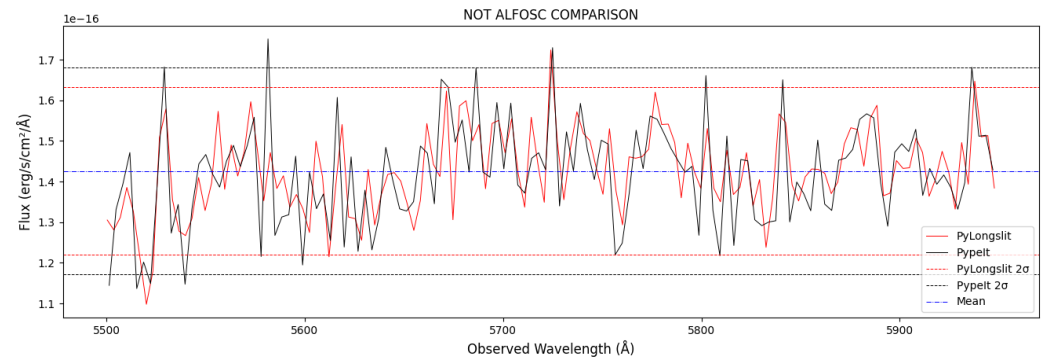


Figure 6: Numerical noise comparison for NOT ALFOSC observation of SDSS_J213510+2728.

The data and instructions needed to reproduce the PyLongslit results can be found [here](#). The raw data and all pipeline output for the Pypelt calculations can be downloaded [here](#). Instructions on how to re-create the results using the raw data can be found in the [Pypelt documentation](#). All the input needed by the pipeline (such as the .pypeit files) are provided together with the raw data.

Limitations

As mentioned in the [Statement of need](#), PyLongslit favors simplicity over high precision. Furthermore, the software is designed to be instrument independent. Due to these design choices, the software does not account for any instrument-specific phenomena, such as detector fringing and the like. The software will likely be less precise than an instrument-specific pipeline (depending on the implementation of the latter). The code is written with focus on loose coupling, and therefore the software can be used as a starting point for an instrument-specific pipeline.

Acknowledgements

We thank the participants of the Nordic Optical Telescope IDA summer course 2024 for very useful feedback on the software.

References

- Bradley, L., Sipőcz, B., Robitaille, T., Tollerud, E., Vinícius, Z., Deil, C., Barbary, K., Wilson, T. J., Busko, I., Donath, A., Günther, H. M., Cara, M., Lim, P. L., Meßlinger, S., Burnett, Z., Conseil, S., Droettboom, M., Bostroem, A., Bray, E. M., ... Perren, G. (2024). *Astropy/photutils: 1.13.0* (Version 1.13.0). Zenodo. <https://doi.org/10.5281/zenodo.12585239>
- Dokkum, P. G. van. (2001). Cosmic-ray rejection by Laplacian edge detection. *Publications of the Astronomical Society of the Pacific*, 113(789), 1420–1427. <https://doi.org/10.1086/323894>

- Horne, K. (1986). An optimal extraction algorithm for CCD spectroscopy. *Publications of the Astronomical Society of the Pacific*, 98(604), 609. <https://doi.org/10.1086/131801>
- Howell, S. B. (2006). *Handbook of CCD astronomy* (2nd ed.). Cambridge University Press.
- McCully, C., Crawford, S., Kovacs, G., Tollerud, E., Betts, E., Bradley, L., Craig, M., Turner, J., Streicher, O., Sipocz, B., Robitaille, T., & Deil, C. (2018). *Astropy/astroscrappy: v1.0.5* (Version v1.0.5). Zenodo. <https://doi.org/10.5281/zenodo.1482019>
- Prochaska, J. X., Hennawi, J. F., Westfall, K. B., Cooke, R. J., Wang, F., Hsyu, T., Davies, F. B., Farina, E. P., & Pelliccia, D. (2020). Pypelt: The Python spectroscopic data reduction pipeline. *Journal of Open Source Software*, 5(56), 2308. <https://doi.org/10.21105/joss.02308>
- Prochaska, J. X., Hennawi, J., Cooke, R., Westfall, K., Wang, F., EmAstro, Tiffanyhsyu, Wasserman, A., Villaume, A., Marijana777, Schindler, J., Young, D., Simha, S., Wilde, M., Tejos, N., Isbell, J., Flörs, A., Sandford, N., Vasović, Z., ... Holden, B. (2020). *pypeit/Pypelt: Release 1.0.0* (Version v1.0.0). Zenodo. <https://doi.org/10.5281/zenodo.3743493>
- Richard Berry, J. B. (2005). *The handbook of astronomical image processing* (2nd ed.). Willmann-Bell. ISBN: 9780943396828
- Tody, D. (1986). The IRAF data reduction and analysis system. In D. L. Crawford (Ed.), *Proc. SPIE instrumentation in astronomy VI* (Vol. 627, p. 733). SPIE. <https://doi.org/10.1117/12.969629>