The Journal of Open Source Software

PlanetMapper: A Python package for visualising, navigating and mapping Solar System observations

Oliver R. T. King [1¶] **and Leigh N. Fletcher** [1]

1 School of Physics and Astronomy, University of Leicester, University Road, Leicester, LE1 7RH, United Kingdom ¶ Corresponding author

DOI: 10.21105/joss.05728

Software

- Review
- Repository
- Archive

Editor: Monica Bobra

Reviewers:

- @steo85it
- @tedjohnson12

Submitted: 14 June 2023
Published: 10 October 2023

Summary

`PlanetMapper` is an open source Python package to visualise, process and understand astronomical observations of Solar System objects, such as planets, moons and rings. Astronomers can use `PlanetMapper` to 'navigate' observations by calculating coordinate values (such as latitude and longitude) for each pixel in an observed image, and can map observations by projecting the observed data onto a map of the target body. Calculated values are exportable and directly accessible through a well documented API, allowing `PlanetMapper` to be used for custom analysis and processing. `PlanetMapper` can also be used to help generate publication quality figures, and has a Graphical User Interface to significantly simplify the processing of astronomical data. `PlanetMapper` can be applied to a wide range of datasets, including both amateur and professional ground-based observations, and data from space telescopes like Hubble and JWST.

Statement of need

In order to accurately interpret astronomical observations of objects in the Solar System, it is crucial to understand the exact geometry and illumination conditions of the observation. Some of the first questions in analysing a new dataset are working out exactly what you are looking at, for example:

- How is the planet oriented in the image?
- What are the latitude and longitude coordinates for each pixel?
- How is the planet's surface illuminated?
- Are points of light in the background sky moons of the target planet, or background stars?
- What is the line-of-sight velocity of the target's surface, and what is the associated doppler correction?

Without answering these kinds of questions, it is often challenging to accurately interpret the data. However, calculating the appearance of a target body is a complex problem, as it requires accurate knowledge of both the target and observers position and orientation in space at specific times. To add to the complexity, non-trivial effects, such as the light travel time from the target to the observer and stellar aberration must also be accounted for.

The NAIF SPICE Toolkit (Acton et al., 2018) was developed by NASA to provide a standardised set of 'SPICE kernels', datasets containing the positions of Solar System objects, and a set of tools to interface with these kernels. This toolkit provides low level functions which can be combined to solve the problem of calculating the appearance of a target body. `PlanetMapper` is designed to significantly simplify the use of SPICE for planetary astronomers, effectively providing a high level interface to the toolkit. For example, the conversion between right

ascension/declination coordinates (in the sky of the observer) to latitude/longitude coordinates (on the target body) requires calling ~10 SPICE functions, but can be done in a single function call with `PlanetMapper`. `PlanetMapper` makes use of the `SpiceyPy` package (Annex et al., 2020) which provides a Python interface to the low level SPICE toolkit functions.

Planetary astronomers have developed toolkits for image navigation, such as the `IDL` language based `DRM` (Fletcher et al., 2009) and the `WinJUPOS` windows application for analysing Jupiter observations (jupos.privat.t-online.de). The USGS `ISIS` software (Laura et al., 2023) also provides a comprehensive set of tools for processing and navigating data from specific instruments on some spacecraft missions, but does not support generalised datasets or ground-based observations. `PlanetMapper` is the first general purpose Python package for navigating and mapping astronomical observations, and it is designed to be used with any form of imaging data observing any Solar System object which has a SPICE kernel available.

Functionality

Publication quality figures can be created with `PlanetMapper` and the `matplotlib` package (Hunter, 2007) - for example, Figure 1 shows a visualisation of the appearance of Saturn at a specific time. These plots can be used to help visualise and plan observation campaigns, and to help interpret observations by providing geometric context for data. Information about the observer-target geometry can also be generated, including data such as calculating the apparent size of the target and testing if a moon is in eclipse or occultation.

Astronomers can use `PlanetMapper` to calculate the geometry of an astronomical observation, and generate a series of 'backplanes' which contain the coordinates (latitude/longitude, illumination angles, ring plane coordinates, velocities etc.) for each pixel in the observation. These backplanes can be saved to FITS data files for future use, or used directly in Python code. `PlanetMapper` contains functions to project observed data to a map (Figure 2), and to export FITS files containing this mapped data.

The `PlanetMapper` Graphical User Interface (GUI), shown in Figure 3, allows users to interactively fit, navigate and save observations with no coding required. This GUI can also be invoked from within Python code, allowing users to easily fit observations within their own data reduction and processing workflow.

`PlanetMapper` is actively used in the processing and analysis of observations in JWST Giant Planets programmes (King, 2023), and has been used to help create data visualisations and figures. It can work with a wide range of datasets, including those from ground (e.g. VLT) and space based (e.g. JWST) telescopes, spacecraft missions and amateur observations. The ability to generate generalised data and plots, such as Figure 1, also makes `PlanetMapper` well suited for more general research purposes, even if the user is not specifically working with astronomical images.

`PlanetMapper` is tested with both unit and integration tests which run automatically using GitHub's continuous integration service. The package is well documented, with all public methods and functions containing detailed docstrings, and documentation automatically built at planetmapper.readthedocs.io. `PlanetMapper` is distributed on PyPI and the code is licensed under the MIT license.

Figures

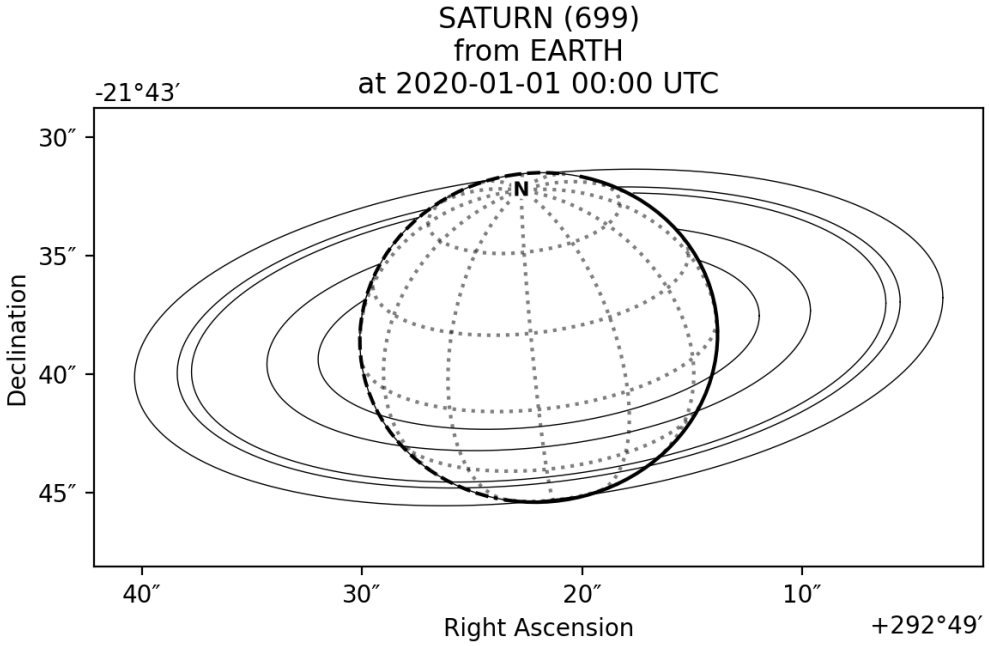

Figure 1: Saturn 'wireframe' plot generated with `PlanetMapper`, visualising the appearance of Saturn from the Earth on 1 January 2020. This plot was created with a single function call, and all elements are fully customisable.

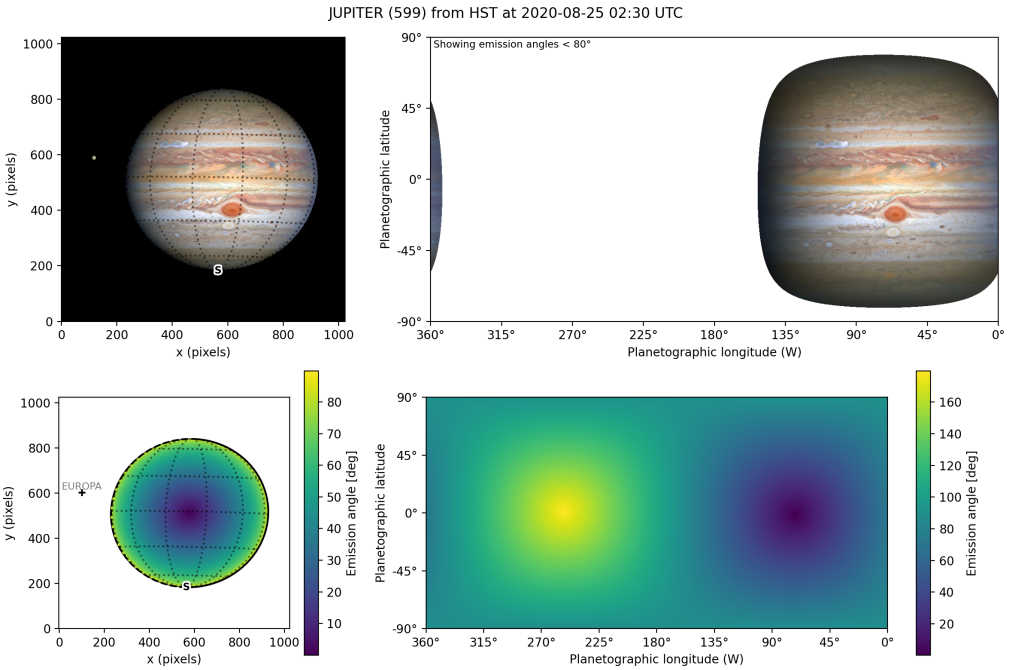

Figure 2: More complex example of `PlanetMapper`'s functionality. The navigated Jupiter observation (top left) was mapped (top right) using `PlanetMapper`. The emission angle backplanes generated with `PlanetMapper` are shown in the bottom panels. Jupiter image credit: NASA, ESA, STScI, A. Simon (Goddard Space Flight Center), and M.H. Wong (University of California, Berkeley) and the OPAL team.

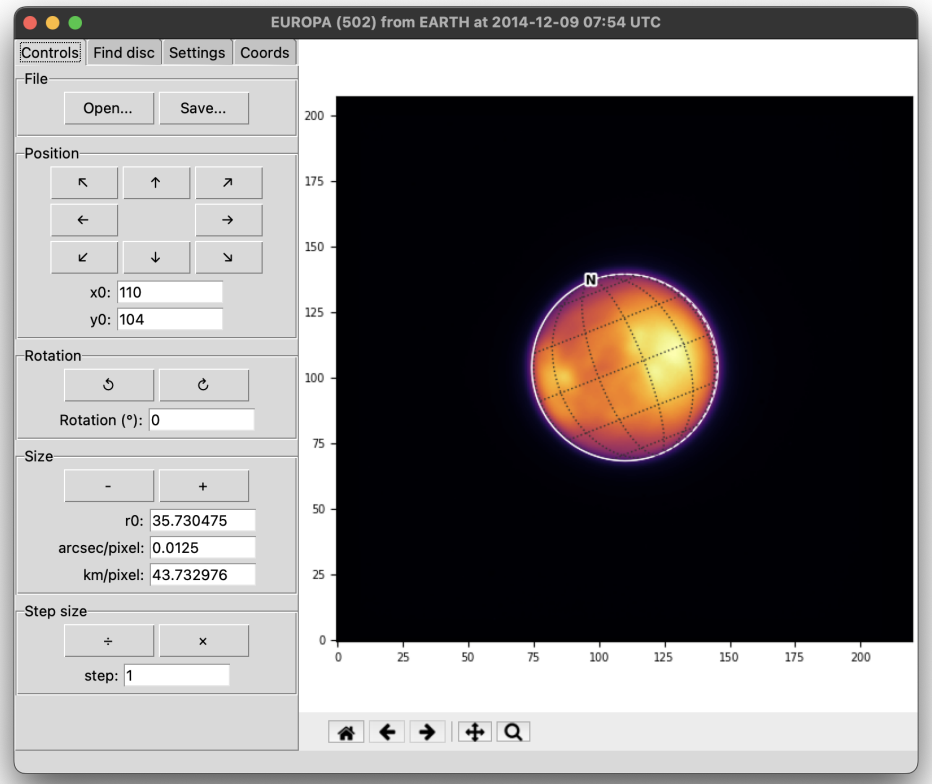

Figure 3: Screenshot of the `PlanetMapper` graphical user interface being used to fit a ground-based VLT observation of Europa (King et al., 2022). The user can adjust the location of Europa's fitted disc (the white circle) until it matches Europa's observed disc. If the observation has embedded WCS information (about the approximate telescope pointing), the disc position, rotation and size is initialised with the position derived from the WCS, so often only small manual adjustments to the disc position are needed.

Acknowledgements

`PlanetMapper` was developed with support from a European Research Council Consolidator Grant (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 723890). Thanks to Mike Roman and Naomi Rowe-Gurney for their suggestions, beta testing and feedback.

References

Acton, C., Bachman, N., Semenov, B., & Wright, E. (2018). A look towards the future in the handling of space science mission geometry. *Planetary and Space Science*, *150*, 9–12. https://doi.org/10.1016/j.pss.2017.02.013

Annex, A. M., Pearson, B., Seignovert, B., Carcich, B. T., Eichhorn, H., Mapel, J. A., Forstner, J. L. F. von, McAuliffe, J., Rio, J. D. del, Berry, K. L., Aye, K.-M., Stefko, M., Val-Borro, M. de, Kulumani, S., & Murakami, S. (2020). SpiceyPy: A pythonic wrapper for the SPICE toolkit. *Journal of Open Source Software*, *5*(46), 2050. https://doi.org/10.21105/joss.02050

Fletcher, L., Orton, G., Yanamandra-Fisher, P., Fisher, B., Parrish, P., & Irwin, P. (2009). Retrievals of atmospheric variables on the gas giants from ground-based mid-infrared imaging. *Icarus*, *200*(1), 154–175. https://doi.org/10.1016/j.icarus.2008.11.019

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

King, O. (2023). *JWSTGiantPlanets/pipelines: v1.0.1* (Version v1.0.1). Zenodo. https://doi.org/10.5281/zenodo.7896689

King, O., Fletcher, L. N., & Ligier, N. (2022). Compositional mapping of europa using MCMC modeling of near-IR VLT/SPHERE and galileo/NIMS observations. *The Planetary Science Journal*, *3*(3), 72. https://doi.org/10.3847/PSJ/ac596d

Laura, J., Acosta, A., Addair, T., Adoram-Kershner, L., Alexander, J., Alexandrov, O., Alley, S., Anderson, D., Anderson, J., Anderson, J., Annex, A., Archinal, B., Austin, C., Backer, J., Barrett, J., Bauck, K., Bauers, J., Becker, K., Becker, T., … Young, A. (2023). *Integrated software for imagers and spectrometers* (Version 7.2.0_RC1). Zenodo. https://doi.org/10.5281/zenodo.7644616