reproject: A Fresh Look at Astronomical Images

[1] Title: reproject: A Fresh Look at Astronomical Images
Author: Ryan Baig
May 18, 2025

[2] The reproject package is a coordinated package for the larger and more well-known astropy package for Python coding. It implements image reprojections for astronomical images and n-dimensional data. In this case, image reprojection is the re-gridding of information from one system to another, whether that be coordinates, pixel resolution, or orientation; reproject does all of the above within a unified package.

[3] I picked this package because I am currently an undergraduate Astronomy student; I found the package to be a great way to introduce myself to the applications of astronomy with Python (with PHYS 265 already providing great help to my introductory astronomy coding). Image reprojection also seems to me to be an interesting and useful tool for astronomers, so I thought that there would be no better way to learn about it than by learning this package.

[4,5] reproject had its first version (0.1) in 2015, being created by authors Christoph Deil and Adam Ginsburg, and Thomas Robitaille.

It is currently on version 0.14.1 (which is the version I am using for this project), released on November 18, 2024. It is currently being maintained by Thomas Robitaille and the Python community.

Documentation on how to contribute to the code is provided in the GitHub repository (see “RELEASE_INSTRUCTIONS.md”).

Montage is another package that performs the same actions; there is a Python version of this package, but it is mostly built on C. Reproject is also implemented in Julia, and sunpy also provides capabilities for image reprojection (however, it’s oriented towards sun-based data).

[6, 7] Installing reproject was a relatively simple process. It can be done via the “standard” pip or conda commands; I used the pip install command:

```
!pip install reproject
```

However, for the code to work properly, other required packages must be installed. These include numpy, astropy, scipy, astropy-healpix, dask, zarr, and fsspec. I had to install the last four, along with reproject. The package also only properly runs in an Anaconda 3 environment.

[8] Although there is no link to the source code on the astropy subpage for reproject, the source code for the package can be found on GitHub: https://github.com/astropy/reproject/tree/main

Ryan Baig
reproject:

[9] There are no other packages that use reproject within them; reproject itself does rely on many other packages, however (see [6,7] and [19]).

[10] The code is used via a Python script or Jupyter notebook. Both provide the ability to execute multiple lines of code required for the code to work properly. For the project, I used a JupyterLab Desktop (JDL) notebook.

[11, 12] I've included examples of using the code within the JDL, including the example reprojection that will be shown in [13]. reproject does not produce figures on its own, but is supported by matplotlib for graphing reprojections.

[13] Below is a figure showing the main capability of reproject: astronomical image reprojection:

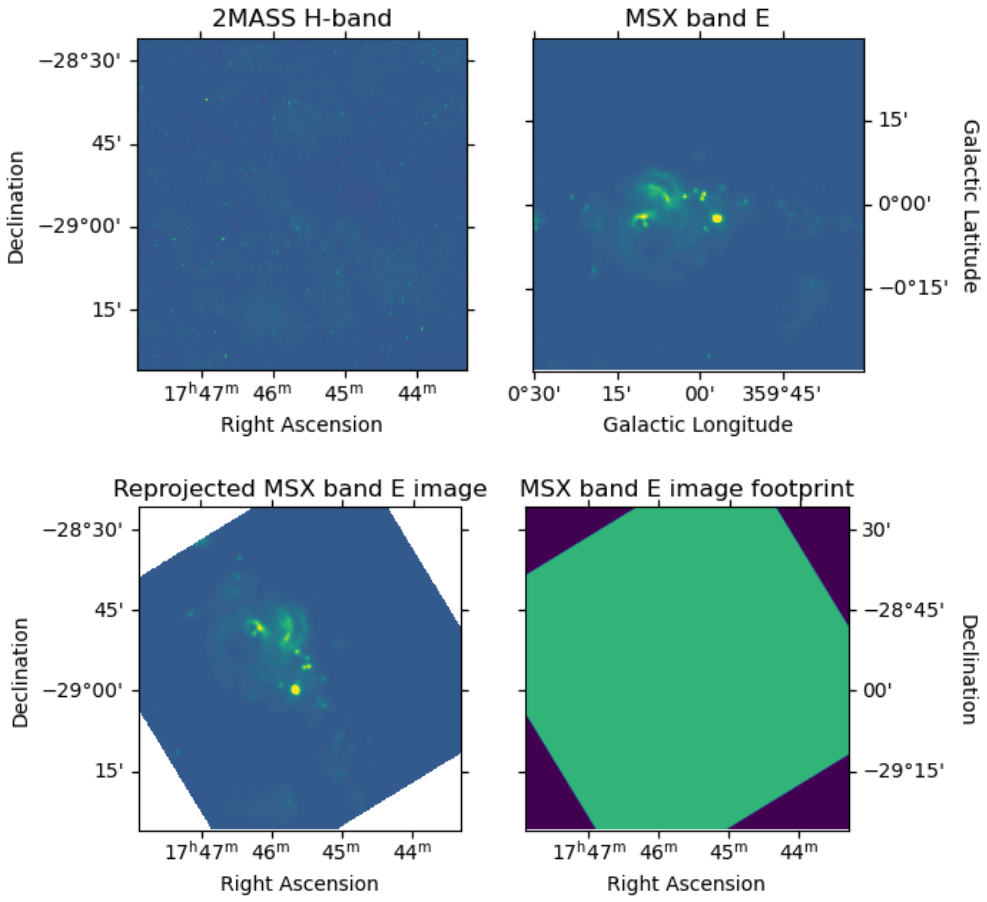

Figure 1 *- Above: Two Galactic Center images with different coordinates. Below: On the left, the MSX band E image reprojected onto the coordinate system of the 2 MASS H-band image; on the right, the "footprint" of the MSX image (what pixels from the reprojected image were used from the original image)*

[14] reproject is Python-based code. Some accompanying code is in C and Cython, but most of it (80%) is in Python.

Ryan Baig
reproject:

[15, 16] reproject takes input from array data that is in a WCS (World Coordinate System) in either a standard or HEALPIX image (an oval-shaped image to simulate a 2D projection of a spherical image) and outputs two arrays: the reprojected data array and the footprint array.

[17, 18] Although there is no native code to check a reprojection's correctness, the package dask has support on reproject to run algorithms to ensure the correctness of the code.

Plotting via matplotlib can also show if there are visual issues (given that you have a general idea of how the reprojection should end up).

[19] As mentioned above, reproject relies on numpy, astropy, scipy, astropy-healpix, dask, zarr, and fsspec. These are all specified in the documentation that I had to install these, along with reproject to use them.

[20] Documentation is mainly found on the readthedocs page, in the form of a Jupyter notebook. It is extensive and provides the code to be able to reproduce. This provided a great introduction to the package and allowed me to learn the code while providing a reference to see if I was implementing it correctly, while displaying its capabilities before writing any line of code.

[21, 22] References:

How to cite: https://zenodo.org/records/10931886 (includes a tool for citations)

Astropy page (readthedocs): https://reproject.readthedocs.io/en/stable/
GitHub repository: https://github.com/astropy/reproject
PyPI page: https://pypi.org/project/reproject/

[23]

Original reproject paper: https://ui.adsabs.harvard.edu/abs/2020ascl.soft11023R/citations
Latest version (only has 1 citation):
https://ui.adsabs.harvard.edu/abs/2024zndo..10931886R/abstract

Only one citation was identified using ADS in the latest ASCL reference. The original paper has been cited 54 times.

[24] I had to learn some new commands in matplotlib to use the package properly (including `ax.imshow` to plot the FITS image itself and `ax.coords` to specify the coordinate values instead of a command like `ax.set_xlabel`)

[25] *DISCLAIMER: Although I had no prior knowledge of this package before this project, I did work with fellow student David Morejon for the coding and retrieving information on this project. The report and my findings are my work.*