





Rastereasy: A Python package for an easy manipulation of remote sensing images

Thomas Corpetti¹, Pierrick Matelot², Augustin de la Brosse¹, and Candide Lissak²

¹ CNRS, UMR 6554 LETG, Univ. Rennes 2, Place du Recteur Henri Le Moal, 35043 Rennes Cedex, France ² Université de Rennes, Inserm, Irset, UMR_S 1085  Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

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

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

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

Summary

Working with remote sensing data often involves managing large, multi-band georeferenced rasters with varying spatial resolutions, extents, and coordinate reference systems ([Mamatov et al., 2024](#)).

rastereasy is a Python library designed to provide a high-level, human-readable interface for common geospatial raster and vector operations (e.g., `.tif`, `.jp2`, `*.shp`) ([Mamatov et al., 2024](#); [Ritter & Ruth, 1997](#)). Built on well-established libraries including `rasterio`, `numpy`, `shapely`, `geopandas`, and `scikit-learn` ([Gillies et al., 2013](#); [Gillies & others, 2013](#); [Harris et al., 2020](#); [Jordahl et al., 2021](#); [Kramer, 2016](#)), it enables users to perform typical GIS tasks—such as resampling, cropping, reprojection, stacking, clipping rasters with shapefiles, or rasterizing vector layers—in just a few lines of code. Some basic Machine Learning functionalities (clustering, fusion) are also implemented.

By abstracting away much of the underlying technical complexity, **rastereasy** makes geospatial processing directly accessible within Python scripts. It is particularly suited for analysts and machine learning practitioners who need to integrate geospatial data handling into their workflows without deep GIS expertise, while also helping experienced geographers prototype more quickly. Beyond core raster operations, it includes utilities for harmonizing multi-source imagery, performing clustering and domain adaptation, and preparing datasets for downstream analysis.

With its current implementation, **rastereasy** provides a solid foundation for further development and integration into the Python geospatial ecosystem.

Statement of need

Many established remote sensing libraries such as `rasterio`, `Raster Forge`, `PODPAC`, `EarthPy` or `GDAL` ([Garrard, 2016](#); [Gillies et al., 2013](#); [Oliveira et al., 2024](#); [Ueckermann et al., 2020](#); [Wasser et al., 2019](#)) provide extensive and powerful functionality for reading, writing, and processing geospatial raster data. However, they can be verbose and often require a solid understanding of geospatial concepts such as projections, data structures, coordinate reference systems, geotransforms, and metadata management. While efficient, many of these libraries are specialized in specific sub-tasks (e.g., visualization, array manipulation, or graphical interfaces) and may not fully meet the needs of users whose primary expertise lies outside GIS—such as data scientists, ecologists, agronomists, or climate researchers. As a result, the learning curve can be steep and may slow down the development of operational workflows.

rastereasy addresses this gap by offering a high-level, human-readable interface that abstracts away much of the underlying complexity while retaining the flexibility of the core libraries.

41 Rather than replacing efficient lower-level libraries, **rastereasy** builds upon them, most notably
42 rasterio, shapely, geopandas and abstracts away repetitive or technical boilerplate code.
43 This design makes it possible to perform in a few lines of Python what would otherwise
44 require many more lines in a raw rasterio or GDAL workflow. It provides streamlined access
45 to common geospatial operations, including:

- 46 ▪ **Band manipulation:** select, reorder, or remove spectral bands by index or by name.
- 47 ▪ **Tiling and stitching:** split large rasters into smaller tiles for processing or machine learning
48 workflows, and reconstruct them when needed.
- 49 ▪ **Harmonization:** align rasters with different resolutions, projections, and extents, optionally
50 adapting spectral values via domain adaptation (Courty et al., 2016).
- 51 ▪ **Visualization tools:** quickly generate color composites, histograms, and spectral plots for
52 georeferenced images.
- 53 ▪ **Filtering:** apply common filters (Gaussian, Laplacian, Sobel, median) or custom convolu-
54 tion kernels..
- 55 ▪ **Basics of machine learning:** clustering (Ikotun et al., 2023) and classification fusion
56 using the Dempster–Shafer framework (Shafer, 1992).

57 **rastereasy** is intended for researchers and practitioners who need to integrate geospatial raster
58 processing into broader data analysis or machine learning pipelines, without having to become
59 GIS specialists. At the same time, it can also benefit geographers and remote sensing experts
60 by offering a concise syntax for prototyping and testing ideas quickly.

61 Example of use

62 The core class of **rastereasy** is **GeoImage**, which wraps a raster as a numpy array while preserving
63 all georeferencing metadata. The **GeoImage** class provides numerous functions to manipulate
64 images (crop, reproject, resample, stack, extract bands, etc.), process them (e.g., filtering),
65 visualize them, harmonize bands, manage band names, or even perform fusion and basic
66 machine learning algorithms.

67 All these operations are carried out while preserving spatial consistency. Users can therefore
68 manipulate spectral bands with high-level functions, compute features such as vegetation or
69 water indices (Xue & Su, 2017) (NDVI, NDWI), extract specific areas of interest, or filter
70 images efficiently.

71 Apart from **GeoImage** class, **rastereasy** also provides functions to handle bounding boxes (e.g.,
72 extracting common areas between two images, or extending the spatial area of an image to
73 match the extent of another) and to create stacks from individual band files.

74 Visualization

75 Here are some outputs for visualize histograms, spectra and color composites.

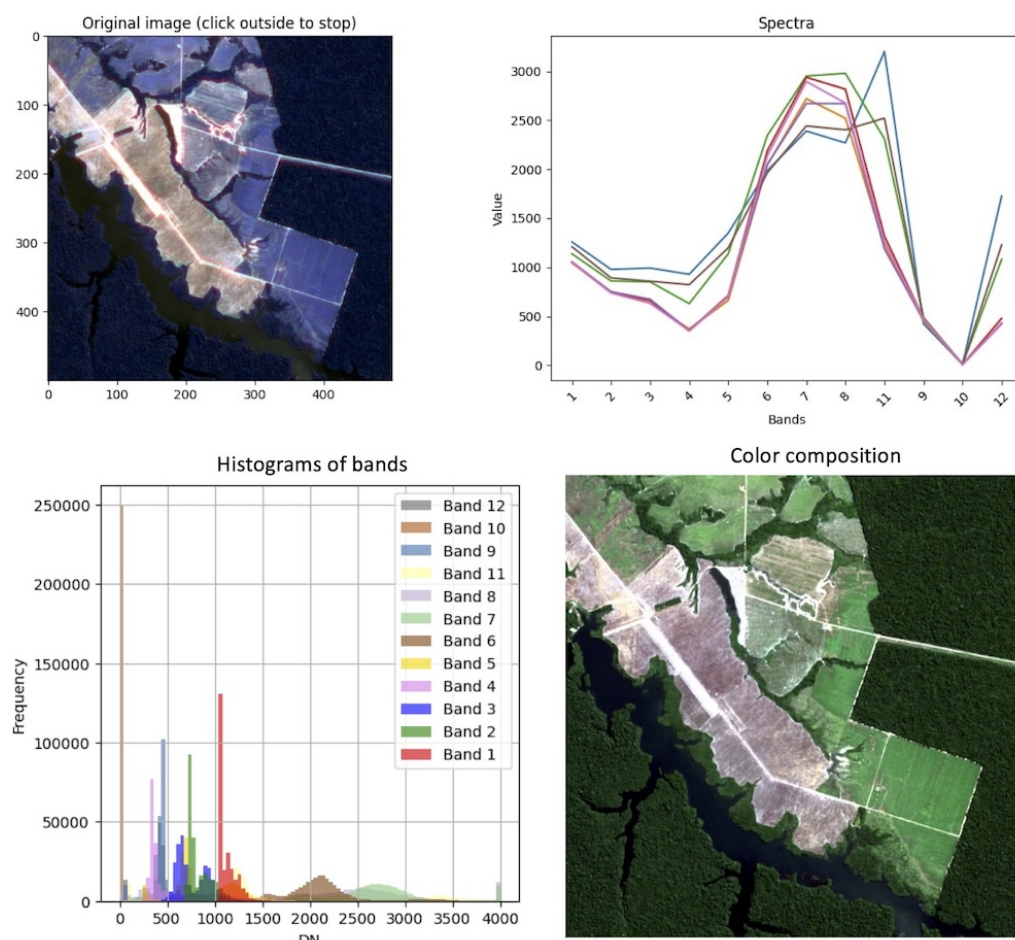


Figure 1: Examples of visualizations provided by rastereasy. Complete examples can be seen on the rastereasy package documentation : <https://rastereasy.github.io/>

Harmonization of bands

Here is an example of adapting the histogram of a source image to a target image (domain adaptation), which is useful, for instance, when applying a machine learning algorithm trained on the target domain to the source domain.

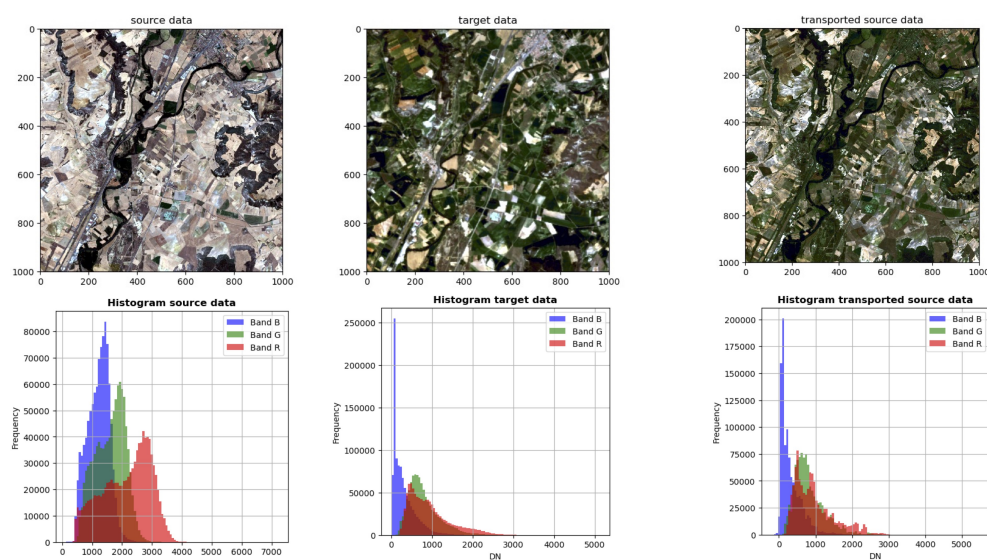


Figure 2: Examples of band harmonization with rastereasy

Filters

Most classical filters (gaussian, laplacian, sobel, median) as well as user-defined generic filters can be performed, as illustrated below.

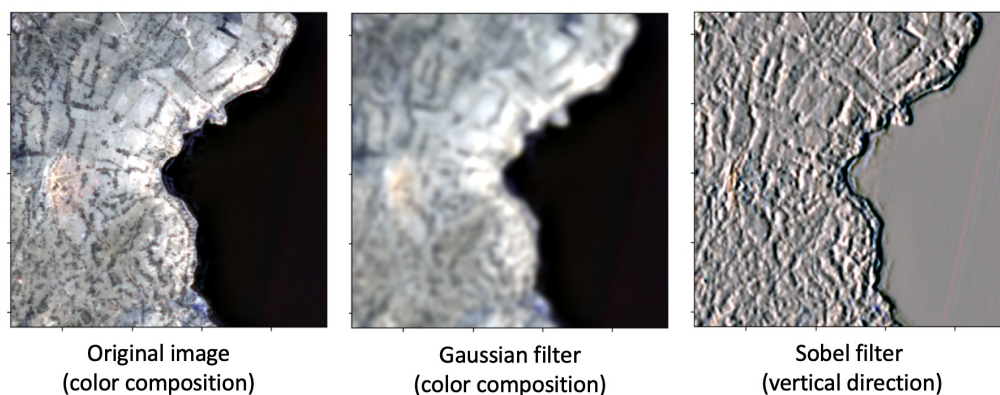


Figure 3: Examples of image filters with gaussian, laplace and sobel

For additional functionalities such as spectral plots, rasterization, harmonization, clustering, or classification fusion, see the [rastereasy documentation](#).

Performance and Scalability

rastereasy is designed as a high-level wrapper around efficient geospatial libraries such as rasterio, numpy, and geopandas. In its current implementation, the default behavior is either to load full rasters into memory and it also supports windowed reading via the underlying rasterio API, allowing users to read and process only subsets of rasters without loading entire files into memory.

91 While this is convenient for small to medium-sized datasets, it can become a limiting factor
 92 when working with very large georeferenced images (e.g., > 10 GB).

93 Currently, most operations are single-threaded and executed in memory; planned enhancements
 94 include lazy loading (processing data on demand) and parallel processing (e.g., for tiling,
 95 reprojection, or large mosaics) to improve scalability.

96 Acknowledgments

97 This library is partly supported by the [ANR MONI-TREE](#) project (ANR-23-CE04-0017)

98 References

- 99 Courty, N., Flamary, R., Tuia, D., & Corpetti, T. (2016). Optimal transport for data fusion
 100 in remote sensing. *2016 IEEE International Geoscience and Remote Sensing Symposium*
 101 *(IGARSS)*, 3571–3574. <https://doi.org/10.1109/IGARSS.2016.7729925>
- 102 Garrard, C. (2016). *Geoprocessing with python*. Simon; Schuster.
- 103 Gillies, S., & others. (2013). The shapely user manual. In <https://pypi.org/project/Shapely>.
- 104 Gillies, S., Ward, B., Petersen, A., & others. (2013). Rasterio: Geospatial raster i/o for python
 105 programmers. In URL <https://github.com/mapbox/rasterio>.
- 106 Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau,
 107 D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming
 108 with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 109 Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Heming, J. (2023). K-means
 110 clustering algorithms: A comprehensive review, variants analysis, and advances in the era of
 111 big data. *Information Sciences*, 622, 178–210. <https://doi.org/10.1016/j.ins.2022.11.139>
- 112 Jordahl, K., Van den Bossche, J., Wasserman, J., McBride, J., Fleischmann, M., Gerard, J.,
 113 Tratner, J., Perry, M., Farmer, C., Hjelle, G. A., & others. (2021). Geopandas/geopandas:
 114 v0. 7.0. In *Zenodo*. <https://doi.org/10.5281/zenodo.3669853>
- 115 Kramer, O. (2016). Scikit-learn. In *Machine learning for evolution strategies* (pp. 45–53).
 116 Springer.
- 117 Mamatov, I., Galety, M. G., Alimov, R., Sriharsha, A., Rofoo, F. F. H., & Sunitha, G.
 118 (2024). Geospatial data storage and management. In *Ethics, machine learning, and*
 119 *python in geospatial analysis* (pp. 150–167). IGI Global Scientific Publishing. <https://doi.org/10.4018/979-8-3693-6381-2.ch007>
- 120
- 121 Oliveira, A., Fachada, N., & Matos-Carvalho, J. P. (2024). Raster forge: Interactive raster
 122 manipulation library and GUI for python. *Software Impacts*, 20, 100657. <https://doi.org/10.1016/j.simpa.2024.100657>
- 123
- 124 Ritter, N., & Ruth, M. (1997). The GeoTiff data interchange standard for raster geographic
 125 images. *International Journal of Remote Sensing*, 18(7), 1637–1647. <https://doi.org/10.1080/014311697218340>
- 126
- 127 Shafer, G. (1992). Dempster-shafer theory. *Encyclopedia of Artificial Intelligence*, 1, 330–331.
- 128 Ueckermann, M. P., Bieszczad, J., Entekhabi, D., Shapiro, M. L., Callendar, D. R., Sullivan,
 129 D., & Milloy, J. (2020). PODPAC: Open-source python software for enabling harmonized,
 130 plug-and-play processing of disparate earth observation data sets and seamless transition
 131 onto the serverless cloud by earth scientists. *Earth Science Informatics*, 13(4), 1507–1521.
 132 <https://doi.org/10.1007/s12145-020-00506-0>

- 133 Wasser, L., Joseph, M., McGlinchy, J., Palomino, J., Korinek, N., Holdgraf, C., & Head, T.
134 (2019). EarthPy: A python package that makes it easier to explore and plot raster and
135 vector data using open source python tools. *Journal of Open Source Software*, 4(43), 1886.
136 <https://doi.org/10.21105/joss.01886>
- 137 Xue, J., & Su, B. (2017). Significant remote sensing vegetation indices: A review of develop-
138 ments and applications. *Journal of Sensors*, 2017(1), 1353691. [https://doi.org/10.1155/](https://doi.org/10.1155/2017/1353691)
139 [2017/1353691](https://doi.org/10.1155/2017/1353691)

DRAFT