

# <sup>1</sup> Rastereasy: A Python package for an easy manipulation of remote sensing images

<sup>3</sup> **Thomas Corpelli**  <sup>1</sup>, **Pierrick Matelot**<sup>2</sup>, **Augustin de la Brosse**<sup>1</sup>, and  
<sup>4</sup> **Candide Lissak** 

<sup>5</sup> 1 CNRS, UMR 6554 LETG, Univ. Rennes 2, Place du Recteur Henri Le Moal, 35043 Rennes Cedex,  
<sup>6</sup> 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](#) 

## <sup>7</sup> Summary

<sup>8</sup> Working with remote sensing data often involves managing large, multi-band georeferenced  
<sup>9</sup> rasters with varying spatial resolutions, extents, and coordinate reference systems ([Mamatov](#)  
<sup>10</sup> et al., 2024).

**Editor:** [Open Journals](#) 

**Reviewers:**

- [@openjournals](#)

**Submitted:** 01 January 1970

**Published:** unpublished

## License

Authors of papers retain copyright<sup>18</sup> and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).<sup>19</sup>

<sup>11</sup> **rastereeasy** 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.

<sup>12</sup> By abstracting away much of the underlying technical complexity, **rastereeasy** 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.<sup>21</sup>

<sup>22</sup> With its current implementation, **rastereeasy** provides a solid foundation for further development and integration into the Python geospatial ecosystem.<sup>25</sup>

## <sup>28</sup> Statement of need

<sup>29</sup> 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.<sup>38</sup>

<sup>39</sup> **rastereeasy** 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.<sup>40</sup>

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

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

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

## Example of use

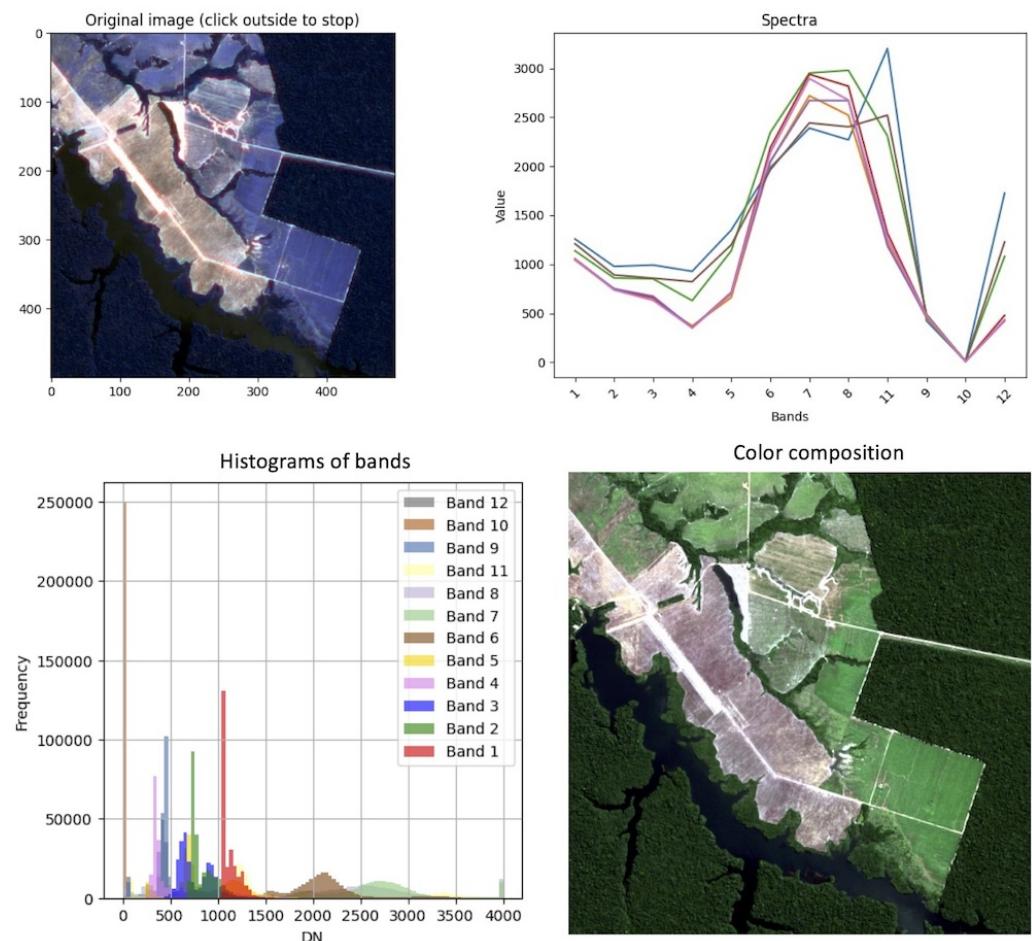
The core class of `rastereeasy` is **GeolImage**, which wraps a raster as a numpy array while preserving all georeferencing metadata. The **GeolImage** class provides numerous functions to manipulate images (crop, reproject, resample, stack, extract bands, etc.), process them (e.g., filtering), visualize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

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

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

## Visualization

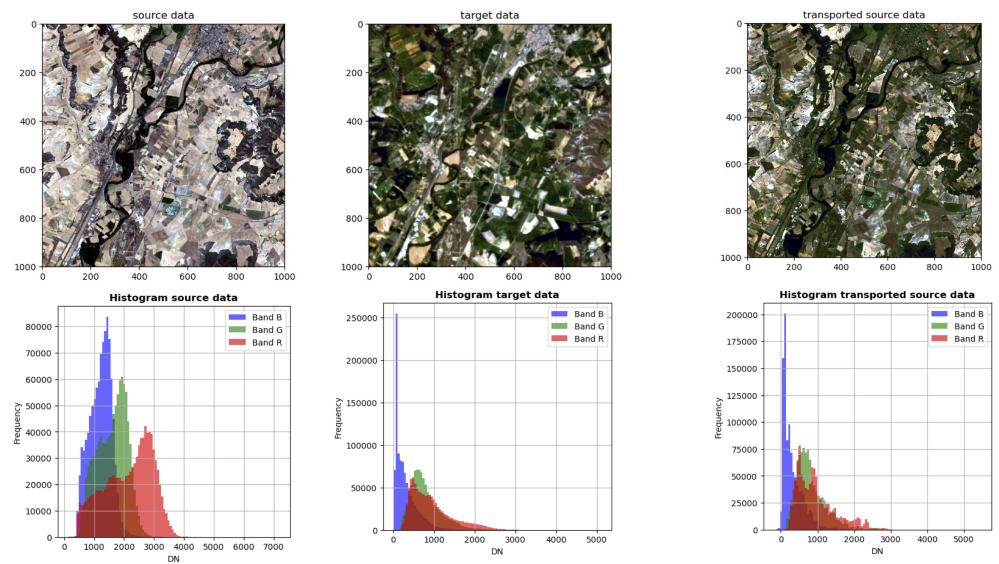
Here are some outputs for visualize histograms, spectra and color composites.



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

## 76      Harmonization of bands

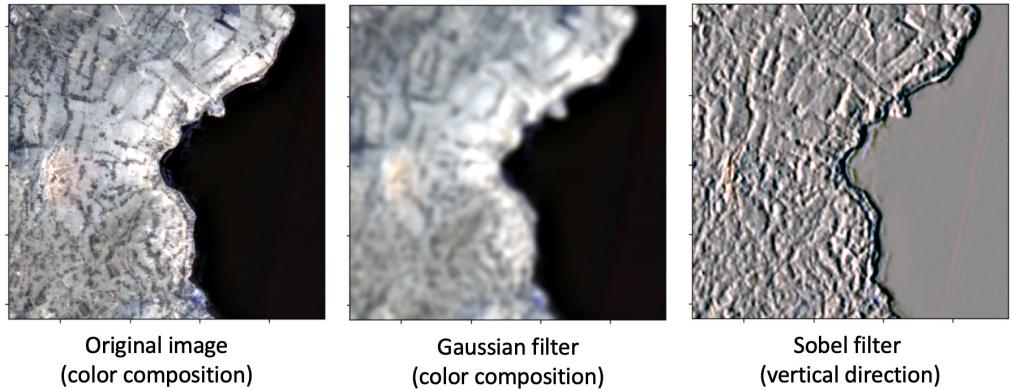
77      Here is an example of adapting the histogram of a source image to a target image (domain  
 78      adaptation), which is useful, for instance, when applying a machine learning algorithm trained  
 79      on the target domain to the source domain.



**Figure 2:** Examples of band harmonization with rastereasy

## 80    Filters

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



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

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

## 85    Performance and Scalability

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

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

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

## <sup>96</sup> Acknowledgments

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

## <sup>98</sup> References

- <sup>99</sup> Courty, N., Flamary, R., Tuia, D., & Corpetti, T. (2016). Optimal transport for data fusion  
<sup>100</sup> in remote sensing. *2016 IEEE International Geoscience and Remote Sensing Symposium  
(IGARSS)*, 3571–3574. <https://doi.org/10.1109/IGARSS.2016.7729925>
- <sup>102</sup> Garrard, C. (2016). *Geoprocessing with python*. Simon; Schuster.
- <sup>103</sup> Gillies, S., & others. (2013). The shapely user manual. In <https://pypi.org/project/Shapely>.
- <sup>104</sup> Gillies, S., Ward, B., Petersen, A., & others. (2013). Rasterio: Geospatial raster i/o for python  
programmers. In URL <https://github.com/mapbox/rasterio>.
- <sup>106</sup> Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming  
with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- <sup>109</sup> Iktoun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaiba, B., & Heming, J. (2023). K-means  
clustering algorithms: A comprehensive review, variants analysis, and advances in the era of  
big data. *Information Sciences*, 622, 178–210. <https://doi.org/10.1016/j.ins.2022.11.139>
- <sup>112</sup> Jordahl, K., Van den Bossche, J., Wasserman, J., McBride, J., Fleischmann, M., Gerard, J.,  
Tratner, J., Perry, M., Farmer, C., Hjelle, G. A., & others. (2021). Geopandas/geopandas:  
v0. 7.0. In Zenodo. <https://doi.org/10.5281/zenodo.3669853>
- <sup>115</sup> Kramer, O. (2016). Scikit-learn. In *Machine learning for evolution strategies* (pp. 45–53).  
Springer.
- <sup>117</sup> Mamatov, I., Galety, M. G., Alimov, R., Sriharsha, A., Rofoo, F. F. H., & Sunitha, G.  
(2024). Geospatial data storage and management. In *Ethics, machine learning, and  
python in geospatial analysis* (pp. 150–167). IGI Global Scientific Publishing. <https://doi.org/10.4018/979-8-3693-6381-2.ch007>
- <sup>121</sup> Oliveira, A., Fachada, N., & Matos-Carvalho, J. P. (2024). Raster forge: Interactive raster  
manipulation library and GUI for python. *Software Impacts*, 20, 100657. <https://doi.org/10.1016/j.simpa.2024.100657>
- <sup>124</sup> Ritter, N., & Ruth, M. (1997). The GeoTiff data interchange standard for raster geographic  
images. *International Journal of Remote Sensing*, 18(7), 1637–1647. <https://doi.org/10.1080/014311697218340>
- <sup>127</sup> Shafer, G. (1992). Dempster-shafer theory. *Encyclopedia of Artificial Intelligence*, 1, 330–331.
- <sup>128</sup> Ueckermann, M. P., Biesczad, J., Entekhabi, D., Shapiro, M. L., Callendar, D. R., Sullivan,  
D., & Milloy, J. (2020). PODPAC: Open-source python software for enabling harmonized,  
plug-and-play processing of disparate earth observation data sets and seamless transition  
onto the serverless cloud by earth scientists. *Earth Science Informatics*, 13(4), 1507–1521.  
<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/10.1155/2017/1353691>

DRAFT