i.hyper: processing hyperspectral imagery in GRASS

Alen Mangafić, Tomaž Žagar

Geodetic Institute of Slovenia, alen.mangafic@gis.si, tomaz.zagar@gis.si

Keywords: hyperspectral imagery, imaging spectroscopy, preprocessing, remote sensing, open source, GRASS GIS

Abstract

Hyperspectral satellite missions such as EnMAP, PRISMA and Tanager have made imaging spectroscopy widely accessible, yet their heterogeneous formats, high dimensionality and demanding preprocessing requirements still hinder efficient scientific use. To address these challenges, we developed i.hyper, a multimodule addon for GRASS designed to provide a harmonized, reproducible and scalable workflow for hyperspectral data. The methodology integrates a unified 3D raster data model with per-band metadata, ensuring consistent handling of wavelength, FWHM, validity and radiometric units across sensors. A modular preprocessing engine implements established imaging spectroscopy and machine-learning techniques, including Savitzky–Golay smoothing, baseline correction, continuum removal and several linear and nonlinear dimensionality-reduction methods such as PCA and Nyström approximation—executed spectrally while preserving spatial alignment.

The resulting toolset comprises the modules i.hyper.import, i.hyper.preproc, i.hyper.composite, i.hyper.explore and i.hyper.export, which together implement a complete workflow from sensor-specific import through preprocessing, visualization and export. The modules resolve practical gaps such as mixed multi-file product structures and metadata conventions, scalable nonlinear reduction and the need for integrated spectral exploration. The addon is available as an official extension in the GRASS repository, enabling direct integration of imaging spectroscopy into established geospatial workflows. This work demonstrates that GRASS can serve as an efficient environment for hyperspectral processing, reducing the data-engineering overhead that typically precedes scientific modelling, and providing a foundation for further preprocessing steps performed by the powerful, native functionalities that GRASS offers.

1. INTRODUCTION

Hyperspectral remote sensing provides densely sampled spectra for every image pixel, typically from the visible through the near infrared and into the shortwave infrared. It captures hundreds of contiguous bands with spectral resolutions on the order of a few nanometres, which enables discrimination of minerals, vegetation properties, soil chemistry and many other surface characteristics that cannot be resolved with traditional multispectral sensors (Pu, 2017). Recent spaceborne imaging spectroscopy missions such as EnMAp (Chabrillat et al., 2024; Storch et al., 2023), PRISMA (Cogliati et al., 2021), and Tanager (Keremedjiev et al., 2024), as well as upcoming missions including CHIME (Celesti et al., 2022) and FLEX (Kraft et al., 2012), have made hyperspectral imagery increasingly available for research and operational applications. Parallel progress in machine learning and numerical libraries has enabled the use of hyperspectral data to the extent that running demanding workflows on a standard workstation is now feasible.

However, exploiting this potential in practice still requires substantial data engineering effort. Hyperspectral products are large, often multi-file, and heterogeneous. Sensors differ in spectral sampling, radiometric units and metadata conventions. There are existing open-source tools specialized for such data, such as EnMAP-Box (Jakimow et al., 2023) and HyperCoast (Liu and Wu, 2024) These tools are powerful but typically implement their own data structures and workflows, which limits tight integration with geospatial raster databeses or processing chains mixed with external Python libraries, or voxel-like 3D raster models.

Several practical gaps motivated our work. First, there is a lack of harmonized import for spaceborne hyperspectral products into a single in-memory representation that can be used consistently across algorithms. Mixed product structures are common: PRISMA and Tanager are distributed as HDF5 containers, while EnMAP products may appear as directory trees with multiple GeoTIFFs and metadata layers. Second, preprocessing for imaging spectroscopy is usually implemented as a sequence of loosely connected tools. Noise suppression, baseline correction, continuum removal and dimensionality reduction are often handled in separate environments, which complicates reproducibility and makes it hard to track the exact pipeline that produced a given data cube. Finally, practical aspects such as missing data handling (for dimensionality reduction algorithms that don't allow null's), clamping of negative reflectance (to deal with numerical errors that can be produced by some atmospheric corrections) and scalable non-linear dimensionality reduction (for example using the Nyström approximation) are rarely integrated into a single workflow engine.

We chose GRASS (Team et al., n.d.) as our execution environment because it combines several properties that are particularly well suited for hyperspectral work. GRASS provides a true 3D raster data type, (3D raster maps), that stores volumetric grids and offers a mature set of accompanying tools for their processing (th r3 family). Its raster engine behaves like a database, with clear handling of spatial regions, tiling and null values, which is important when dealing with large cubes that must be processed in memory-efficient chunks. GRASS integrates well with Python, so we can move between 3D raster maps and NumPy arrays back and forth and integrate state-of-the-art algorithms from scientific libraries such such as SciPy

(Virtanen et al., 2020), RAPIDS cuML(Raschka et al., 2020), scikit-learn (Pedregosa et al., 2011) and TorchGeo (Stewart et al., 2021) without sacrificing spatial context. At the same time, it offers a stable addon ecosystem and a strong community that already uses GRASS for demanding remote sensing tasks.

On this basis we developed i.hyper, a multimodule addon for hyperspectral imagery in GRASS. The addon implements a consistent data model and a complete processing chain for imaging spectroscopy. It includes product-specific import, a flexible spectral preprocessing engine, composite generation, interactive spectral exploration and export. In this paper we first describe the methodological choices behind i.hyper, focusing on the data model and preprocessing and dimensionality reduction strategies. We then present the resulting modules i.hyper.import, i.hyper.preproc, i.hyper.composite, i.hyper.explore and i.hyper.export, and demonstrate how they form an integrated workflow for working with spaceborne hyperspectral imagery.

2. Methodology

2.1 Data model and design goals

The design goal of i.hyper was to reduce the amount of ad-hoc data wrangling required before hyperspectral imagery can be used for analysis or machine learning. The central decision was to represent all imported cubes as GRASS 3D raster maps. In this representation the horizontal dimensions correspond to easting and northing and the vertical dimension encodes the spectral axis, so each voxel contains either reflectance or radiance for one band at one spatial position. This is conceptually identical to the conventional three-dimensional hyperspectral cube and matches how imaging spectroscopy is typically introduced in remote sensing literature.

To make different sensors interoperable we defined a shared metadata scheme. For every band the system stores wavelength, FWHM (full width at half maximum), a validity flag and a unit indicator as metadata comments. This per-band metadata is parsed during import and then reused by all subsequent i.hyper modules. In this way, the same cube can be visualized in wavelength space, processed by filters that depend on the spectral sampling, or reduced to components using methods that ignore bands flagged as invalid. Products that contain all-zero or all-NULL bands, or known artefact ranges, can be marked with validity equal to zero at import stage so that later preprocessing steps can skip them.

Because spaceborne hyperspectral products are not standardized across missions, the importer uses product-specific backends. PRISMA and Tanager products are read through h5py, while EnMAP L2A products are treated as multi-file directories. The user can either define a representative file or select a folder when invoking i.hyper.import from the graphical interface. In the latter case the module scans the directory, identifies the relevant bands and metadata and assembles them into a single cube. This behaviour is essential for mixed products such as EnMAP, which distribute visible and NIR, SWIR and metadata layers in separate files.

The module temporarily sets the computational region to match the native resolution and extent of the input data, assembles the 3D raster maps and then restores the user region. No spatial or spectral resampling is performed at this stage. We deliberately postponed resampling decisions to later parts of the workflow to preserve the original measurements. After that, the users can simply change region, resample and change the imported hyperspectral data with GRASS native tools and also transfer the metadata from it's parent 3D raster maps with r3.support to make it compliant with the other i.hyper modules.

2.2 Spectral preprocessing

The second design focus was a general, pipeline-based preprocessing engine for hyperspectral cubes. Our aim was to align common spectroscopy and machine learning practice with GRASS, rather than to invent new algorithms. i.hyper.preproc implements established methods from spectral analysis and dimensionality reduction, applied voxelwise along the spectral dimension and kept fully aligned in space. The implementation relies on NumPy and SciPy for array manipulation and signal processing and uses scikit-learn for all dimensionality reduction methods. The pipeline is defined through module options and flags and the module prints the effective sequence, for example "Savitzky–Golay → Baseline correction → Continuum removal → Nystroem", to make the applied transformations transparent.

2.2.1 Savitzky–Golay smoothing and derivatives

The first stage in many workflows is spectral smoothing and derivative computation. We chose the Savitzky–Golay filter because it is widely used in analytical chemistry and spectroscopy for denoising spectra while preserving the shape and depth of absorption bands (Savitzky and Golay, 1964). The filter fits a local polynomial of specified order to a moving window centred on each band and uses the resulting coefficients either to output a smoothed value or to approximate derivatives of the spectrum.

In i.hyper.preproc the user controls this step through three parameters: window_length, polyorder and derivative_order. A polynomial of order polyorder is fitted over a window of window_length bands for each voxel, and derivative_order determines whether only smoothing or also first or higher derivatives are returned. Setting derivative_order to zero yields a smoothed cube suitable for subsequent baseline correction or continuum removal. A positive derivative_order produces spectral derivatives, which can enhance subtle absorption features and reduce the impact of illumination differences.

Hyperspectral reflectance data can contain small negative values, especially after erroneus atmospheric correction. These are problematic for derivative-based interpretation and for some downstream algorithms. For this reason, i.hyper.preproc offers an option to clamp values less than zero to zero before applying Savitzky–Golay. The -z flag activates this clamping. In our own work we used clamping to stabilise derivatives in soil spectra while accepting that this imposes a weak prior that reflectance should be non-negative in the analysed range.

2.2.2 Baseline correction

Spectral baselines can vary due to sensor characteristics, residual atmospheric effects, surface roughness or illumination geometry. Such trends deform the overall shape of reflectance curves and can obscure diagnostic absorption features. Baseline correction attempts to remove these broad trends so that absorption features are measured relative to a more neutral reference (Barnes et al., 1989).

In i.hyper.preproc baseline correction is an optional step activated by the -b flag. For each voxel, the module fits a baseline model across the spectral axis and subtracts it from the smoothed spectrum. The current implementation uses a robust, low-order

approximation designed to capture slow variations rather than band-scale structure. The corrected cube is still on an arbitrary vertical scale but exhibits reduced offset differences and more comparable absorption shapes. In our experiments this step made it easier to compare spectra from different surfaces and to interpret subsequent continuum removal.

2.2.3 Continuum removal

Continuum removal is a standard technique in spectroscopy for normalising absorption features. It constructs a convex hull over the spectrum and divides the spectrum by this hull, which yields relative depths between zero and one for each wavelength (Clark and Roush, 1984). This representation facilitates comparison of absorption band positions and shapes between spectra that differ in overall brightness.

The continuum removal stage in i.hyper.preproc operates voxelwise across the spectral axis. After optional smoothing and baseline correction the module computes the spectral continuum for each voxel, stores it internally and then generates a continuum-removed cube by normalisation. The original and continuum-removed cubes share the same spatial grid, which allows direct comparison in exploratory analysis or modelling. In our own work continuum removal was particularly useful for highlighting subtle mineralogical features in SWIR ranges.

2.2.4 Handling missing and negative values

Practical hyperspectral products often contain gaps: masked bands, detector artefacts, or wavelengths outside the valid calibration range. Many machine learning algorithms and dimensionality reduction methods cannot handle null values directly (Pedregosa et al., 2011). To make those methods usable without discarding affected pixels, i.hyper.preproc implements spectral interpolation of missing values.

When the -q flag is set, the module identifies null values in bands flagged as valid and interpolates them along the spectral axis using neighbouring valid bands. This is done per voxel so that spatial patterns are preserved. Bands marked as invalid in metadata are not interpolated and remain excluded from analysis. This design respects sensor-level information while resolving algorithmic limitations. For methods that are highly sensitive to missing values, such as PCA or Nystroem, this interpolation is practically necessary.

In addition to interpolation, the module can clamp negative values to zero via the -z flag, as discussed above. The combination of interpolation and clamping provides a controlled way to satisfy the assumptions of methods that expect complete, non-negative input without modifying the original cube that is stored in GRASS.

2.2.5 Dimensionality reduction and Nyström approximation

High dimensionality and strong band-to-band correlation are characteristic of hyperspectral data. Dimensionality reduction compresses these bands into a smaller set of components that retain most of the variance or information content. i.hyper.preproc relies entirely on scikit-learn for this step and exposes several linear and non-linear methods: PCA, KPCA, Nystroem approximation, FastICA, TruncatedSVD, NMF and SparsePCA (Pedregosa et al., 2011).

For PCA and TruncatedSVD we used these methods to obtain orthogonal components that capture most of the variance in the spectra and to reduce collinearity prior to regression or classification models. KPCA allows non-linear structure in the spectra to be captured through kernel functions, but full kernel methods are difficult to scale to large cubes. To address this, i.hyper.preproc implements the Nystroem approximation available in scikit-learn. In this approach a subset of spectra is used to approximate the kernel map, and the full dataset is then projected into this low-rank feature space. The module allows the user to specify dr_components, kernel type, gamma and other kernel parameters, and to export the fitted feature map and subsequent PCA compressor to a pickle file via dr_export. This exported model can later be applied to external spectra, such as laboratory measurements, to ensure that image and field data share the same reduced feature space.

Large hyperspectral cubes may not fit into memory when processed in a single batch. i.hyper.preproc therefore supports chunked dimensionality reduction controlled by dr_chunk_size. When a positive chunk size is specified, the cube is processed in sequential blocks of spectra. For kernel-based methods this results in an approximate solution, but it allows us to scale non-linear dimensionality reduction to entire satellite scenes. The trade-off between exactness and scalability is left to the user through these parameters.

3. RESULTS

3.1 i.hyper.import

The i.hyper.import module implements the data model described above and serves as the entry point to the i.hyper workflow. It reads PRISMA, EnMAP and Tanager products and converts them to a single 3D raster map that encapsulates all spectral bands. During import the module selects appropriate product drivers from an internal library, parses metadata, validates bands and populates the per-band comments for wavelength, FWHM, validity and unit.

From the user perspective the module can be invoked either from the command line or through the graphical interface. In the graphical interface the user selects a product type and then either clicks a representative file or chooses an entire folder. The latter option is essential for multi-file products such as EnMAP L2A, where spectral bands and metadata are distributed over several GeoTIFFs and XML files. i.hyper.import scans the selection, discovers the relevant files and constructs the cube accordingly. The -n flag controls whether all-NULL bands are imported. By default, such bands are skipped, but when this flag is set, they are included and marked as invalid in metadata.

The module can optionally generate false-colour composites during import through the composites and composites_custom parameters, where the users can define the wavelengths of the bands that it wants to visualize. These composites are produced as 2D rasters and are useful for quickly inspecting the imported scene. Region handling is transparent: a temporaty computational region is set to match the input data so that all bands align exactly, then the original region is restored at the end of processing. The imported cube can immediately be used with other GRASS modules or with the rest of the i.hyper family.

An example of a created hyperspectral composite can be seen below. On Figure 1 we can see a SWIR-geology composite generated from a PIRSMA scene.

Figure 1: PRISMA SWIR-geology composite generated with i.hyper.import. Data source: PRISMA Product © Italian Space Agency (ASI), used under ASI License to Use.

3.2 i.hyper.preproc

The i.hyper.preproc module embodies the preprocessing methodology described in Section 2.2. It takes a 3D raster map as input and returns another, whose spectral axis has been transformed by a user-defined pipeline. The module is controlled by options specifying Savitzky–Golay parameters, selection of baseline correction and continuum removal, dimensionality reduction method, number of components and kernel parameters. Flags -b, -c, -q and -z activate baseline correction, continuum removal, interpolation of missing values and clamping of negative values respectively.

In this module, we focused on two classes of workflows. The first class consisted of purely spectral preprocessing applied to reflectance cubes. A typical command smoothed the spectra, interpolated missing values and removed the continuum, yielding normalised absorption features ready for further exploration. The second class combined preprocessing with dimensionality reduction. Both workflows can be used at the same time in a pipeline, which respects the correct order of the processes.

The module loops over spatial tiles, reads spectral vectors into NumPy arrays, applies the pipeline and writes back to the output 3D raster map. The spatial region and resolution are preserved, so all components remain perfectly co-registered with the original cube. This allows immediate use of the preprocessed or reduced cubes in GRASS-based classification or regression, for example with r.learn.ml2 or with external Python scripts that ingest NumPy arrays directly from GRASS via garray.array3d().

3.3 i.hyper.composite

The i.hyper.composite module translates hyperspectral 3D raster maps into 2D colour composites suitable for visual interpretation. It takes an input raster and an output prefix, reads the wavelength metadata from r3.info and selects bands nearest to specified target wavelengths. Predefined composites exist for true-colour RGB, colour-infrared, SWIR for agriculture and SWIR for geology. Users can also pass custom triplets of wavelengths via composites_custom, rather than band nubers, which differ from product to product. The module temporarily converts the relevant bands to 2D rasters using r3.to.rast, applies contrast enhancement with i.colors.enhance and composes the final image using r.composite.

3.4 i.hyper.explore

The i.hyper.explore module provides interactive spectral exploration. It accepts one or more rasters and either a list of coordinates or a vector point map and then uses r3.what (GRASS addon) to sample values across all bands at each query location. The resulting spectra are plotted with wavelengths on the x-axis when metadata are present or with component indices when the cube represents dimensionality-reduced data.
The module can open an interactive window or export static figures as PNG, PDF or SVG and can alternatively export the spectra in JSON format when the -p flag is used. This makes it easy to use the spectra with external plotting or analysis environments.

3.5 i.hyper.export

The i.hyper.export module closes the loop by exporting a 3D raster hyperspectral map to a compressed multi-band GeoTIFF. Internally it converts the cube to individual 2D rasters using r3.to.rast, groups them into an imagery group and writes them as a single file with r.out.gdal, using DEFLATE compression and an appropriate predictor for floating point data. Null values are mapped to a fixed nodata value. The export preserves band order and spatial alignment, so external tools can treat the result as a standard multi-band raster stack. Wavelength and other metadata remain in GRASS and can be reattached in external workflows if needed, but within the i.hyper context these metadata are already available to all modules.

3.6 Installation and availability

The i.hyper addon is published as an official GRASS addon in the GRASS repository. It can be installed on any GRASS version newer than 8.4 using the standard g.extension mechanism. The fact that the modules are part of the official addon catalogue ensures discoverability and better long-term m development with the GRASS community.

4. DISCUSSION

The i.hyper addon was built to reduce the distance between raw hyperspectral products and analysis-ready data. The methodology section highlighted two central aspects of this effort: the choice of 3D raster maps as a unified cube representation with rich metadata and the implementation of a flexible spectral preprocessing pipeline that embeds established spectroscopy and machine learning methods into a spatially aware environment.

The results demonstrate that these design choices translate into a coherent toolset. i.hyper.import hides the heterogeneity of mission-specific formats and presents a uniform cube with consistent metadata. i.hyper.preproc exposes a range of commonly used preprocessing and dimensionality reduction steps through a single module that operates directly on cubes and reports its pipeline explicitly. i.hyper.composite, i.hyper.explore

and i.hyper.export provide the visualisation and data exchange capabilities that are necessary to integrate hyperspectral workflows into broader geospatial analyses.

Below, we outline directions for future work. Additional importers are needed to support both new and legacy hyperspectral missions. Metadata handling can be revised, enriched and expanded to store sensor descriptions more comprehensively, moving beyond the current use of comments in the raster map metadata and enabling more flexible storage solutions. This will facilitate the future export of hyperspectral data and/or metadata in formats such as Zarr, HDF5, or other industry-standard formats like BSQ or ENVI, which would support multidimensional arrays and metadata export. A key priority is implementing atmospheric correction and radiance-to-reflectance conversion within GRASS, enabling users to process raw radiance products and generate analysis-ready reflectance cubes without leaving the environment. Additionally, incorporating machine learning tasks such as classification or regression will support the creation of quantitative or semi-quantitative maps, expanding the tool's applicability to predictive modeling. The integration of field spectrometer measurements and support for aerial hyperspectral imagery would further enhance the tool's capabilities. Incorporating preprocessing steps like wavelet transforms would enrich the processing pipeline. Finally, performance improvements could be achieved through optimization with native GRASS libraries, enhanced parallelization, and advanced visualization features, including interactive 3D hyperspectral cube rendering.

5. CONCLUSION

The i.hyper add-on extends GRASS with a preprocessing workflow for spaceborne hyperspectral imagery. It provides a harmonised import path for satellite hyperspectral products, a general spectral preprocessing and dimensionality reduction engine, tools for composite generation and spectral exploration and a straightforward export facility. By relying on the 3D raster map data model, leveraging per-band metadata and integrating established numerical libraries, i.hyper makes it possible to perform sophisticated hyperspectral analysis entirely within GRASS and the wider Python ecosystem while preserving spatial context.

As hyperspectral missions continue to proliferate and as machine learning models become ever more capable, such integrated tooling is essential. i.hyper already performs the bulk of the data engineering that previously had to be scripted case by case. With the planned extensions and refinements outlined above, it can become a central component of open-source imaging spectroscopy workflows that bridge satellite, airborne and field data.

6. REFERENCES

Barnes, R.J., Dhanoa, M.S., Lister, S.J., 1989. Standard Normal Variate Transformation and De-Trending of Near-Infrared Diffuse Reflectance Spectra. Appl Spectrosc 43, 772–777. https://doi.org/10.1366/0003702894202201

Celesti, M., Rast, M., Adams, J., Boccia, V., Gascon, F., Isola, C., Nieke, J., 2022. The Copernicus Hyperspectral Imaging Mission for the Environment (Chime): Status and Planning. International Geoscience and Remote Sensing Symposium (IGARSS) 2022-July, 5011–5014. https://doi.org/10.1109/IGARSS46834.2022.9883592

Chabrillat, S., Foerster, S., Segl, K., Beamish, A., Brell, M., Asadzadeh, S., Milewski, R., Ward, K.J., Brosinsky, A., Koch, K., Scheffler, D., Guillaso, S., Kokhanovsky, A., Roessner, S., Guanter, L., Kaufmann, H., Pinnel, N., Carmona, E., Storch, T., Hank, T., Berger, K., Wocher, M., Hostert, P., van der Linden, S., Okujeni, A., Janz, A., Jakimow, B., Bracher, A., Soppa, M.A., Alvarado, L.M.A., Buddenbaum, H., Heim, B., Heiden, U., Moreno, J., Ong, C., Bohn, N., Green, R.O., Bachmann, M., Kokaly, R., Schodlok, M., Painter, T.H., Gascon, F., Buongiorno, F., Mottus, M., Brando, V.E., Feilhauer, H., Betz, M., Baur, S., Feckl, R., Schickling, A., Krieger, V., Bock, M., La Porta, L., Fischer, S., 2024. The EnMAP spaceborne imaging spectroscopy mission: Initial scientific results two years after launch. Remote Sens Environ 315, 114379. https://doi.org/10.1016/J.RSE.2024.114379

Clark, R.N., Roush, T.L., 1984. Reflectance spectroscopy: Quantitative analysis techniques for remote sensing applications. J Geophys Res Solid Earth 89, 6329–6340. https://doi.org/10.1029/JB089IB07P06329

Cogliati, S., Sarti, F., Chiarantini, L., Cosi, M., Lorusso, R., Lopinto, E., Miglietta, F., Genesio, L., Guanter, L., Damm, A., Pérez-López, S., Scheffler, D., Tagliabue, G., Panigada, C., Rascher, U., Dowling, T.P.F., Giardino, C., Colombo, R., 2021. The PRISMA imaging spectroscopy mission: overview and first performance analysis. Remote Sens Environ 262, 112499. https://doi.org/10.1016/J.RSE.2021.112499

Jakimow, B., Janz, A., Thiel, F., Okujeni, A., Hostert, P., van der Linden, S., 2023. EnMAP-Box: Imaging spectroscopy in QGIS. SoftwareX 23, 101507. https://doi.org/10.1016/J.SOFTX.2023.101507

Keremedjiev, M., Roth, K.L., Barentsen, G., Haag, J., Bourne, H., Wurster, K., Radel, M., Giuliano, P., Mcdonald, T., Duren, R., Thompson, D.R., Guido, J., Green, R.O., Seaman, K., Keremedjiev, M., Roth, K.L., Barentsen, G., Haag, J., Bourne, H., Wurster, K., Radel, M., Giuliano, P., Mcdonald, T., Duren, R., Thompson, D.R., Guido, J., Green, R.O., Seaman, K., 2024. First Light Results from the Tanager Hyperspectral Mission. AGUFM 2024, GC41H-0019.

Kraft, S., Del Bello, U., Bouvet, M., Drusch, M., Moreno, J., 2012. FLEX: ESA's Earth Explorer 8 candidate mission. International Geoscience and Remote Sensing Symposium (IGARSS) 7125–7128. https://doi.org/10.1109/IGARSS.2012.6352020

Liu, B., Wu, Q., 2024. HyperCoast: A Python Package for Visualizing and Analyzing Hyperspectral Data in Coastal Environments. J Open Source Softw 9, 7025. https://doi.org/10.21105/JOSS.07025

Pedregosa, F., Michel, V., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot andÉdouardand, M., Duchesnay, andÉdouard, Duchesnay Édouard, 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830.

Pu, R., 2017. Overview of Hyperspectral Remote Sensing. Hyperspectral Remote Sensing 1–30. https://doi.org/10.1201/9781315120607-1/OVERVIEW-HYPERSPECTRAL-REMOTE-SENSING

Raschka, S., Patterson, J., Nolet, C., 2020. Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. Information (Switzerland) 11. https://doi.org/10.3390/info11040193

Savitzky, A., Golay, M.J.E., 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Anal Chem 36, 1627–1639. https://doi.org/10.1021/AC60214A047/ASSET/AC60214A047.FP.PNG_V03

Stewart, A.J., Robinson, C., Corley, I.A., Ortiz, A., Ferres, J.M.L., Banerjee, A., 2021. TorchGeo: Deep Learning With Geospatial Data. GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems. https://doi.org/10.1145/3557915.3560953

Storch, T., Honold, H.P., Chabrillat, S., Habermeyer, M., Tucker, P., Brell, M., Ohndorf, A., Wirth, K., Betz, M., Kuchler, M., Mühle, H., Carmona, E., Baur, S., Mücke, M., Löw, S., Schulze, D., Zimmermann, S., Lenzen, C., Wiesner, S., Aida, S., Kahle, R., Willburger, P., Hartung, S., Dietrich, D., Plesia, N., Tegler, M., Schork, K., Alonso, K., Marshall, D., Gerasch, B., Schwind, P., Pato, M., Schneider, M., de los Reyes, R., Langheinrich, M., Wenzel, J., Bachmann, M., Holzwarth, S., Pinnel, N., Guanter, L., Segl, K., Scheffler, D., Foerster, S., Bohn, N., Bracher, A., Soppa, M.A., Gascon, F., Green, R., Kokaly, R., Moreno, J., Ong, C., Sornig, M., Wernitz, R., Bagschik, K., Reintsema, D., La Porta, L., Schickling, A., Fischer, S., 2023. The EnMAP imaging spectroscopy mission towards operations. Remote Sens Environ 294, 113632. https://doi.org/10.1016/J.RSE.2023.113632

Team, G.D., Landa, M., Neteler, M., Metz, M., Petrášová, A., Petráš, V., Clements, G., Zigo, T., Larsson, N., Kladivová, L., Haedrich, C., Blumentrath, S., Andreo, V., Cho, H., Gebbert, S., Nartišs, M., Kudrnovsky, H., Delucchi, L., Zambelli, P., Lennert, M., Mitášová, H., Chemin, Y., Pešek, O., Barton, M., Tawalika, C., Ovsienko, D., Bowman, H., n.d. GRASS GIS. https://doi.org/10.5281/ZENODO.14918754

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Bardelli, A. Pietro, Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C.N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D.A., Hagen, D.R., Pasechnik, D. V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G.A., Ingold, G.L., Allen, G.E., Lee, G.R., Audren, H., Probst, I., Dietrich, J.P., Silterra, J., Webber, J.T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J.L., de Miranda Cardoso, J.V., Reimer, J., Harrington, J., Rodríguez, J.L.C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N.J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P.A., Lee, P., McGibbon, R.T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T.J., Robitaille, T.P., Spura, T., Jones, T.R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y.O., Vázquez-Baeza, Y., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 2020 17:3 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Acknowledgements

This work was done within the scope of the targeted research project V2-24072: Development of a unified spatial physical change detection system with artificial intelligence,
financially supported by the Slovenian Research and Innovation Agency and the Surveying and Mapping Authority of the Republic of Slovenia.

We are grateful for the mentoring provided by Anna Petrášová who was supported by NSF award #2303651.