Hyperspectral Processing

Felix M. Riese

Mar 14, 2021

CONTENT:

We present the hyperspectral processing module for the HydReSGeo dataset. This module includes classes, functions, and scripts for the processing.

Author Felix M. Riese

Citation see Citation and in the bibtex file

CHAPTER

ONE

INSTALLATION

1.1 Dependencies

Install Python 3, e.g. via Anaconda.

Install the required packages with conda:

```
conda install --file requirements.txt
```

Or install the required packages with pip:

```
pip install -r requirements.txt
```

1.2 Install latest development version

The module does not need to be installed. It can be run directly from the downloaded folder. If you want to easily import it into your code, try:

```
git clone https://github.com/felixriese/hyperspectral-processing.git
cd hyperspectral-processing/
python setup.py install
```

And to import:

```
import hprocessing
```

CHAPTER

TWO

FIRST STEPS

2.1 Process the HydReSGeo Dataset

The first steps can be found in this Process_HydReSGeo_Dataset.ipynb.

First, import the modules. Afterwards, run the automatic processing function `processHydReSGeoDataset()` with the paths which need to be adapted to your local paths. In the config file, the necessary paths and options need to be set.

```
from hprocessing.ProcessFullDataset import processHydReSGeoDataset

output_df = processHydReSGeoDataset(
    config_path="config/HydReSGeo.ini",
    data_directory="data/HydReSGeo/")
```

The pandas DataFrame `output_df` includes all processed data.

2.2 Example Plots

Exemplary plots can be found in this Example_Plots.ipynb.

Todo: Add exemplary plots.

CHAPTER

THREE

CONFIGURATION

The configuration of the processing for datasets such as the HydReSGeo dataset should be provided in a configuration file in the folder `config/`. One example for a configuration file is the `config/HydReSGeo.ini`:

```
[Paths]
positions_hyp = rs/masks/positions_hyp_lowres.csv
positions_lwir = rs/masks/positions_IR.csv
data_hyp = rs/hyp/
data_lwir = rs/lwir/
data_sm = hyd/TDR.csv
data_output = ../data/output/HydReSGeo_Output.csv
ignore_hyp_measurements = rs/masks/ignore_hyp_measurements.csv
ignore_hyp_fields = rs/masks/ignore_hyp_fields.csv
ignore_hyp_datapoints = rs/masks/ignore_hyp_datapoints.csv
masks_hyp = rs/masks/hyp_masks.csv

[Process]
overwrite_csv_file = True
grid_rows = 1
grid_columns = 1
hyp_image_rows = 50
hyp_image_columns = 50
time_window_width = 6
hyp_stat_mode = median
hyp_spectralon_factor = 0.95
```

CHAPTER

FOUR

HYDRESGEO DATASET

The HydReSGeo dataset is published in [HydReSGeo]. In the following, the file structure is described and the files in `rs/masks/` are summarized.

4.1 File structure

```
├── gpr
│   ├── field_A
│   │   ├── plot_A_2017-08-15T11:03:21+02.00.sgy
│   │   └── ...
│   └── ...
├── hyd
│   ├── TDR.csv
│   ├── bromide.csv
│   ├── coresamples.csv
│   ├── irrigation_protocol.txt
│   ├── read_in_hydro_data.ipynb
│   ├── sensor_pos.txt
│   └── tensio.csv
├── rs
│   ├── fieldspec.csv
│   ├── hyp
│   │   ├── 20170815_hyp_meas1
│   │   │   ├── Auto000.cub
│   │   │   ├── Auto000.cue
│   │   │   ├── Auto000.hdr
│   │   │   ├── Auto000.jpg
│   │   │   ├── Auto000_PAN.tiff
│   │   │   ├── Auto000_highres.hdr
│   │   │   └── ...
│   │   └── ...
│   ├── lwir
│   │   ├── ir_export_20170815_P0000004_000_12-23-22.csv
│   │   └── ...
│   └── masks
│       ├── hyp_masks.csv
│       ├── ignore_hyp_datapoints.csv
│       ├── ignore_hyp_fields.csv
│       ├── ignore_hyp_measurements.csv
│       ├── meta_IR.txt
│       ├── positions_IR.csv
│       └── positions_hyp_lowres.csv
```

(continues on next page)

(continued from previous page)

```
├── rs_masked
└── site
```

4.2 File descriptions

The file descriptions for the geophysical files (`gpr`), the hydrological files (`hyd`), the remote sensing files (`rs/fieldspec.csv`, `rs/hyp`, and `rs/lwir`), and the site files (`site`) are described in [HydReSGeo].

Overall, we divide the hyperspectral data into folders, which include images (= files = datapoints), which consist of different zones (= measurement fields). The fields/zones in each hyperspectral and LWIR image are named as A1-D2 or zone1-zone8 as follows:

```
zone_dict = {
        "A1": "zone1",
        "A2": "zone2",
        "B1": "zone3",
        "B2": "zone4",
        "C1": "zone5",
        "C2": "zone6",
        "D1": "zone7",
        "D2": "zone8"}
```

Over the three measurement days of the HydReSGeo dataset, the sensor positions and angles of the hyperspectral camera and LWIR camera change. This change is taken into account by including time-dependend masks in `rs/masks/`, which are described in the following.

4.2.1 `hyp_masks.csv`

This file includes information about four wodden bars which are included in the measurement area and should be masked. The columns are:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.
- `start_row`, `end_row`, `start_col`, `end_col`: Start and end rows and columns for the mask.
- `bar[1-4]_p[1-2]_[x,y]`, `bar[1-4]_height`: Information about the geometry of the wodden bar. This is used in ProcessEnviFile.

4.2.2 `ignore_hyp_datapoints.csv`

This file includes information about which hyperspectral images (datapoints) need to be excluded for various reasons. The columns are:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.
- `filenumber`: Number of the hyperspectral image in the respective measurement folder.

4.2.3 ignore_hyp_fields.csv

This file includes information about the zones/fields to be ignored in each hyperspectral image due to several reasons: a GPR measurement within that field at the same time, the irrigation platform, or a person walking through the image. The columns are:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.
- `filenumber`: Number of the hyperspectral image in the respective measurement folder.
- `zone`: Zone/field which needs to be ignored within the respective file. For the HydReSGeo dataset, eight zones are defined. They are numerated either as A1, A2, B1, B2, C1, C2, D1, and D4, or as zone1 to zone8 for technical reasons.

4.2.4 ignore_hyp_measurements.csv

This file includes information about which measurement folders to be ignored. The column is:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.

4.2.5 meta_IR.csv

This file is not important for this repository (for now) and can be ignored.

4.2.6 positions_hyp_lowres.csv

This file includes information about the eight different measurement zones of the HydReSGeo dataset as well as the spectralon (= white reference) with respect to the hyperspectral images. The columns are:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.
- `spec_row_start`, `spec_row_end`, `spec_col_start`, and `spec_col_end`: Start and end rows and columns for the spectralon.
- `zone[1-8]_[row/column]_[start/end]`: Start and end rows and columns for the eight measurement zones.

4.2.7 positions_IR.csv

This file includes information about the eight different measurement zones of the HydReSGeo dataset with respect to the LWIR data. The columns are:

- `measurement`: Measurement folder name in the format `YYYYmmDD_meas[1-9]`.
- `zone[1-8]_[row/column]_[start/end]`: Start and end rows and columns for the eight measurement zones.

4.3 Opening the CSV files

The CSV files can be opened in `python3` with `pandas`:

```
import pandas as pd

df = pd.read_csv("hyp_masks.csv", sep="\s+")
```

4.4 References

CHAPTER

FIVE

CITATION

The bibtex file including both references is available in bibliography.bib.

5.1 Code

F. M. Riese, "Hyperspectral Processing Scripts for HydReSGeo Dataset," Zenodo, 2020. DOI:10.5281/zenodo.3706418

```
@misc{riese2020hyperspectral,
    author = {Riese, Felix~M.},
    title = {{Hyperspectral Processing Scripts for the HydReSGeo Dataset}},
    year = {2020},
    DOI = {10.5281/zenodo.3706418},
    publisher = {Zenodo},
    howpublished = {\href{https://doi.org/10.5281/zenodo.3706418}{doi.org/10.5281/
↪zenodo.3706418}}
}
```

5.2 Dataset

S. Keller, F. M. Riese, N. Allroggen, and C. Jackisch, "HydReSGeo: Field experiment dataset of surface-subsurface infiltration dynamics acquired by hydrological, remote sensing, and geophysical measurement techniques," GFZ Data Services, 2020. DOI:10.5880/fidgeo.2020.015

```
@misc{keller2020hydresgeo,
    author = {Keller, Sina and Riese, Felix~M. and Allroggen, Niklas and
              Jackisch, Conrad},
    title = {{HydReSGeo: Field experiment dataset of surface-subsurface
              infiltration dynamics acquired by hydrological, remote
              sensing, and geophysical measurement techniques}},
    year = {2020},
    publisher = {GFZ Data Services},
    DOI = {10.5880/fidgeo.2020.015},
}
```

CHAPTER

SIX

CHANGE LOG

6.1 [1.0.1] - 2021-03-14

- [ADDED] Python 3.9 support.

6.2 [1.0.0] - 2020-04-21

- Initial release

CHAPTER

SEVEN

PROCESSFULLDATASET

CHAPTER

EIGHT

PROCESSENVIFILE

Class and functions to process envi file.

The package *spectral* is used in this repository with permission by the authors (see spectralpython/spectral/issues/103).

class hprocessing.ProcessEnviFile.**ProcessEnviFile**(*image*, *wavelengths: list*, *bbl: list*, *zone_list: list*, *positions: dict*, *index_of_meas: int*, *mask=None*, *grid: tuple = (1, 1)*, *stat_mode: str = 'median'*, *spectralon_factor: float = 0.95*)

Class to process ENVI files.

Parameters

- **image** (*spectral image*) – Image file of the hyperspectral image
- **wavelengths** (*list of int*) – List of measured wavelength bands
- **bbl** (*list of str/int/bool*) – List of bbl values that say which wavelengths are measured in good quality (1) and which are not (0)
- **zone_list** (*list*) – List of measurement zones in the image. That does not include the spectralon (white reference). If a zone needs to be ignored, it needs to be removed from this list.
- **positions** (*dict*) – Dictionary with information of the positions config file
- **index_of_meas** (*int*) – Index of dataset in positions CSV file
- **mask** (*numpy array, optional (default=None)*) – Zero = pixel to be masked, One = good pixel
- **grid** (*tuple (int, int), optional (default=(1, 1))*) – Size of the grid (rows, columns). If row/column zero, every pixel is one row/column.
- **stat_mode** (*str*) – Mode for calculating the "mean spectrum". Possible values: median, mean, max, max10 (= maximum of the top 10 pixels), std.
- **spectralon_factor** (*float, optional (default=0.95)*) – Factor of how much solar radiation the spectralon reflects.

getCalibratedSpectra(*spectra: pandas.core.frame.DataFrame*, *spectralon: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

Get calibrated spectra.

Parameters

- **spectra** (*pd.DataFrame*) – Not-calibrated spectra

- **spectralon** (*pd.DataFrame*) – Spectrum of the spectralon (white reference)

Returns Calibrated spectra

Return type pd.DataFrame

getEdgesFromPrefix(*prefix: str*)

Get start and end values of edges in rows and columns.

These four values describe a rectangle on the hyperspectral image.

Parameters **prefix** (*str*) – Name of the rectangle which corresponds to the resulting edges

Returns **edges** – Edges of the resulting rectangle

Return type list of [int, int, int, int]

getMeanSpectraFromSquareGrid(*edges*, *mode: str = 'median'*)

Get mean spectra from squared grid area (region of interest, ROI).

Parameters

- **edges** (*list of 4 int*) – Edges of the square (row_start, row_end, col_start, col_end)
- **mode** (*str*) – Mode for calculating the "mean spectrum". Possible values: median, mean, max, max10 (= maximum of the top 10 pixels), std.

Returns **df** – DataFrame with all spectra as rows.

Return type pd.DataFrame

getMeanSpectrumFromRectangle(*edges: list*, *mode: str = 'median'*) → pandas.core.frame.DataFrame

Get mean spectrum from rectangle area (region of interest, ROI).

Parameters

- **edges** (*list of 4 int*) – Edges of the square (row_start, row_end, col_start, col_end)
- **mode** (*str*) – Mode for calculating the "mean spectrum". Possible values: median, mean, max, max10 (= maximum of the top 10 pixels), std.

Returns **df_spectrum** – Dataframe with the spectrum as row, wavelengths as columns

Return type pd.DataFrame

Todo:

- implement statsmodels.robust.scale.Huber as robust mean

---

getMultipleSpectra()

Get soil spectrum for measurements with multiple soil spectra.

In these measurements, there are multiple spectra measured: the one of the spectralon and the multiple soil spectra. The soil spectra are combined with the spectralon spectrum to get the reflectance spectra of the soil measurements.

Returns **zones_fields_df** – DataFrame containing the reflectance spectra of the soil measurements

Return type DataFrame

Todo:

- Replace pandas by numpy

getRealGridSize(*edges: list*)

Calculate grid size.

Parameters **edges** (*list of 4 int*) – Edges of the square (row_start, row_end, col_start, col_end)

Returns Number of grid rows and columns

Return type (int, int)

hprocessing.ProcessEnviFile.**convertWavelength**(*wavelength*) → str

Convert wavelength into string in nano meter.

Parameters **wavelength** (*str, int, float*) – Wavelength in nano meters or micro meters

Returns Wavelength as string

Return type str

Raises **ValueError** – If wavelegnth between 5 and 200

hprocessing.ProcessEnviFile.**formatTime**(*time*, *ampm*)

Format time from 6:02PM to 18:02 (or 10:02PM to 22:02).

Parameters

- **time** (*str*) – Time formated as "6:02:24" (or 10:02:24)
- **ampm** (*str*) – Formatted as "A" or "P" for AM or PM

Returns **newtime** – Time formatted as "18:02:24" (or 22:02:24)

Return type str

hprocessing.ProcessEnviFile.**getCalibratedSpectrum**(*soil*, *spectralon*, *spectralon_factor: float = 0.95*)

Calibrate hyperspectral spectrum from soil via spectralon.

Calibrate each bin of the soil spectrum on the respective bin in the spectralon (= white reference) spectrum.

Parameters

- **soil** (*list of float*) – Spectrum of the soil.
- **spectralon** (*list of float*) – Spectrum of the spectralon.
- **spectralon_factor** (*float*) – Factor of how much solar radiation the spectralon reflects.

Returns List of reflectance values for each band of the soil image.

Return type np.array of floats

hprocessing.ProcessEnviFile.**getEdgesForGrid**(*edges: list*, *grid_real*)

Calculate the grid geometry (edges).

Parameters

- **edges** (*list of 4 int*) – Edges of the square (row_start, row_end, col_start, col_end)
- **grid_real** (*(int, int)*) – Number of grid rows and columns

Returns **new_edges** – New edges of the grid inside the rectangle.

Return type list of int

`hprocessing.ProcessEnviFile.`**`getEnviFile`**(*filepath*)

Read from envi file.

The envi files consist of one header file (.hdr) and one image file (.cue). The documentation for the ENVI functions can be found here: https://github.com/spectralpython/spectral/blob/master/spectral/io/envi.py The documentation for the ENVI header files can be found here: https://www.harrisgeospatial.com/docs/ENVIHeaderFiles.html

Parameters **filepath** (*str*) – Path to header file

Returns

- **header** (*spectral header*) – Contains description, samples, lines, bands, header offset, file type, data type, interleave, sensor type, z plot average, z plot range, default stretch, plot titles, reflectance, byte order, bbl, wavelength, wavelength units.
- **image** (*spectral image*) – Image file of the hyperspectral image. Order of the indices: image[row, column], image[row, column, band] See here: http://www.spectralpython.net/fileio.html

`hprocessing.ProcessEnviFile.`**`getEnviHeader`**(*filepath*)

Read envi header file.

Parameters **filepath** (*str*) – Path to header file

Returns **header** – Contains description, samples, lines, bands, header offset, file type, data type, interleave, sensor type, z plot average, z plot range, default stretch, plot titles, reflectance, byte order, bbl, wavelength, wavelength units.

Return type spectral header

`hprocessing.ProcessEnviFile.`**`getGridElements`**(*grid: tuple*) → list

Get elements of a 2-dimensional grid.

Parameters **grid** (*tuple (int, int), optional (default=(1, 1))*) – Size of the grid (rows, columns). If row/column zero, every pixel is one row/column.

Returns List of grid elements

Return type list of tuples (int, int)

`hprocessing.ProcessEnviFile.`**`readEnviHeader`**(*header*)

Read out the header of the ENVI file.

The documentation of the ENVI Header Files can be found here: https://www.harrisgeospatial.com/docs/ENVIHeaderFiles.html

Parameters **header** (*envi header format*) – Header of the ENVI file

Returns

- **date_formatted** (*str*) – Date formatted as yyyymmdd
- **time_formatted** (*str*) – Time formatted as hh:mm:ss

`hprocessing.ProcessEnviFile.`**`removeBadBands`**(*spectrum*, *wavelengths*, *bbl*)

Remove bands that are marked as bad in bbl list.

Parameters

- **spectrum** (*list of int*) – Spectrum as a list.
- **wavelengths** (*list of int*) – List of measured wavelength bands
- **bbl** (*list of str/int/bool*) – List of bbl values that say which wavelengths are measured in good quality (1) and which are not (0)

Returns

- **newwavelengths** (*list of int*) – List of "good" wavelength bands
- **newspectrum** (*list of int*) – Spectrum of all "good" bands as a list

`hprocessing.ProcessEnviFile.`**`validateWavelengths`**(*wavelengths: list*, *bbl: list*)

Validate wavelengths and bbl.

Parameters

- **wavelengths** (*list of int*) – List of measured wavelength bands
- **bbl** (*list of str/int/bool*) – List of bbl values that say which wavelengths are measured in good quality (1) and which are not (0)

Returns

- *list of int* – Validated *wavelengths* list
- *list of int* – Validated *bbl* list

Raises **`ValueError:`** – Raised if *wavelengths* and *bbl* are of a different length.

CHAPTER

NINE

PLOTUTILS

BIBLIOGRAPHY

[HydReSGeo] S. Keller, F. M. Riese, N. Allroggen, and C. Jackisch, "HydReSGeo: Field experiment dataset of surface-subsurface infiltration dynamics acquired by hydrological, remote sensing, and geophysical measurement techniques," GFZ Data Services, 2020. DOI:10.5880/fidgeo.2020.015

PYTHON MODULE INDEX

h

INDEX