1. Introduction

hsi is a python package for high level hyperspectral data processing scripting. The type of datasets can be any optical spectroscopic with intensity and wavenumber axes. The upcoming content shows how to handle the data from an imaging spectroscopic point of view rather pure programming and developing approach.

A high-level programming holds a short scripting structure for fast and efficient hyperspectral data analysis.

hsi comes with 4 dataset-examples for understanding.

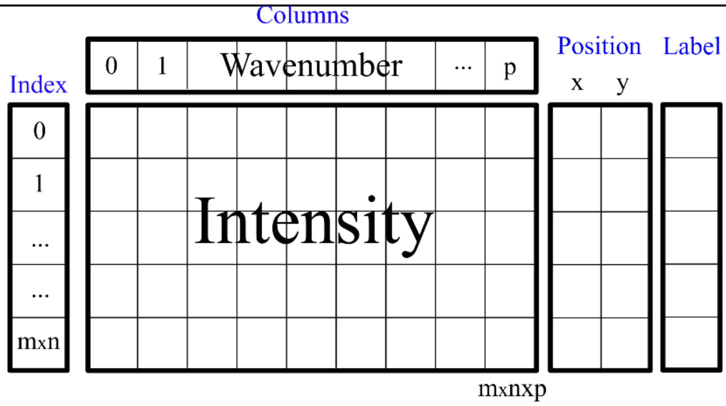
Plastics	Datasets of different plastics
Plastics+Tissue	Raman Map of tissue model and microplastics
Plastics+Layer of plastics	Z scanning of transparent plastic layers
Cancer+noncancer bladder	Cancer and noncancer tissue models
3D imaging	3D Raman Imaging of translucent tissue and plastics

The previous datasets will illustrate how to process raw datasets and obtain a processed result by several *hsi* tools. A complete information regarding HSI functions is shown in the appendix C.

1.1. Structure of type of data in HIS.

Fig. 1 shows the hyperspectral data object that is stablished in the *hsi.py* package. The object holds an index for the spectra captured, columns hold wavenumber or *x* axis. The intensity table contains the data of the Raman intensity. Datasets from Imaging provide spatial information that are represented as *x* and *y* in position. The *label* indicates the final result after processing the data.

Hyperspectral Imaging is represented by a 2D representation considering *x* and *y* coordinates and the information of label. Confocal Raman microscopy supplies *z* or depth information and thereby is possible to plot 3D images through voxels.



(a)

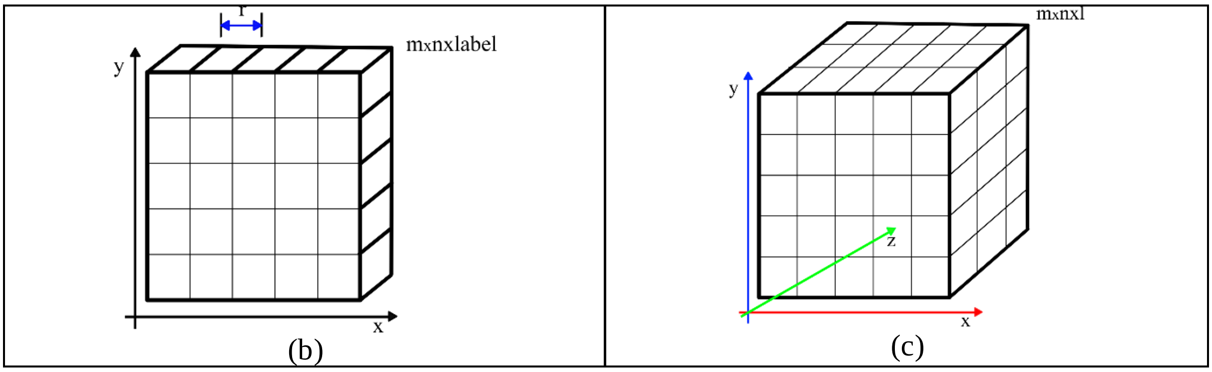


Figure 1. Hyperspectral object. (a) representation of hyperpectral object, (b) 2D plotting of hyperspectral data and (c) voxel illustration for 3D imaging.

1.2. Python and Spyder interface

For further information how to install the required requirements, the link: provides information about installation, issues and further examples.

1.3. HSI objects, functions and assignments.

HSI is based on dataframes (Fig.1) and contains properties and methods for reading, manipulation, preprocessing, processing and visualization.

Loading the hsi.py. Firstly, select the correct path where the library hsi.py is saved.

```
from hsi import*
```

1.3.1. Creating/Deleting an hsi object

Creating an instance/object hyperspectral is starting step for further steps. The object can contain any kind of data, spc, csv, txt or custom as long as it keeps the standard dataframe shape (Fig.1)

Creating	`mapa = hyper_object('name')`
deleting	`del mapa`

1.3.2. Reading/saving csv, spc, mat and txt files

Some standard data files are spc, csv and txt.

Reading single spc	`mapa.read_single_spc(path)`
Reading several spc	`mapa.read_multi_spc(path)`
Reading hologram mat file	`mapa.read_spc_holomap(path)`
Csv	`mapa.read_csv(path, resolution = 1)`
txt	`mapa.read_txt(path, spacer, skipcols, skiprows)`
Show data	`mapa.show(True)/mapa.show(False)`
Save data	`mapa.save_data(path, 'name')`

For specific definition of the parameters check the code help.

1.3.3. Basic Data Handling methods

His provides methods for data manipulation such as getting the whole intensity or position dataframe. The user can build the hyperobject dataframe using set_data() and adding set_position() and the remain properties.

`data = mapa.get_data()`	`mapa.set_data(data)`
`position = mapa.get_position()`	`mapa.set_position(position)`
`wavenumber = mapa.get_wavenumber()`	`mapa.set_wavenumber(calibration_row)`

`number = mapa.get_number()`	`mapa.set_resolution(res)`
`label = mapa.get_label()`	`mapa.set_label(names)`

1.4. Pre-Processing and tools

Tools for preprocessing hyperdata

1.4.1 Manipulation Tools

Basic tools for hyperspectral handling.

Hyper appending	`mapa.append(hyper)`
Hyper Concatanateing	`mapa.concat([hyper1, hyper2, ...])`
Mean	`mean = mapa.mean()`
	`spectrum = mapa.get_pixel(x, y)`
Difference	`diff = mapa.diff(hyper)`
Obtaining intensity at certain peak	`intensity = mapa.get_peak_intensity(peak)`
	`area = mapa.get_area(waveumber_range)`

1.4.2. Selecting wavelength ranges

The selection of work range defines the spectral region of interest. Fig. 2 shows the full spectral range.

Finger print region (Fig. 2(b))	`mapa.keep(lower = 400, upper = 1800)`
High wavenumber region (Fig. 2(c))	`mapa.keep(lower = 2800, upper = 3200)`
Without silent region (Fig. 2(d))	`mapa.keep(lowers = [400, 2800], uppers = [1800, 3200])`

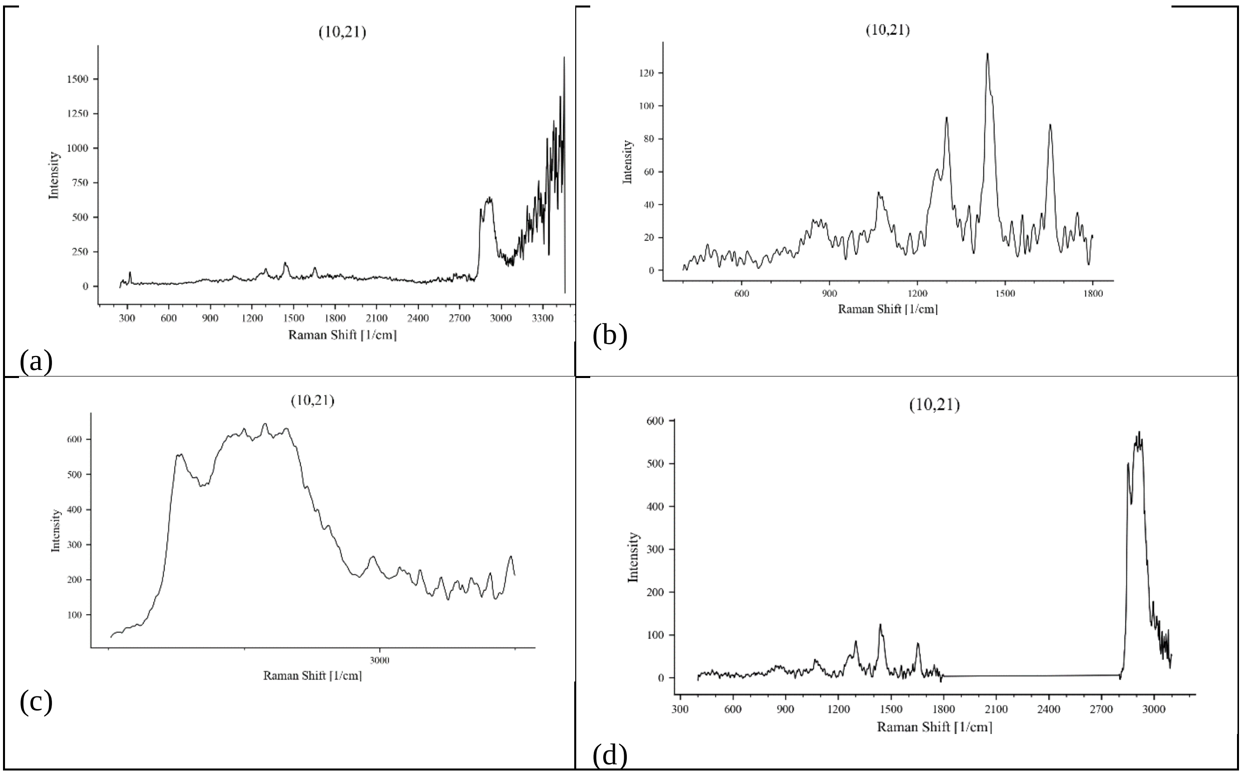


Figure 2. Selection of work range. (a) full range, (b) low wavenumber (finger print region), (c) high wavenumber and (d) selection of low and high wavenumber regions and ignoring the silent region.

1.4.3. Pre-processing

Removing Outliers

Intensity threshold	`mapa.threshold(lower = 50, upper = 500)`
	`mapa.covarience(contamination = 10)`
Cosmic Ray removal tool	`mapa.spikes(threshold = 7, window = 3)`
Advanced Cosmic Ray removal tool	`mapa.adv_spikes(components) + further steps`

Smoothing

Savitzky golay filter	`mapa.gol(window = 7, interpolation = 3, order = 0)`
Gaussian filter	`mapa.gaussian(variance = 1800)`
PCA denoising	`mapa.pca_denoising(components = 3)`

Baseline correction

Rubber Band correction	`mapa.rubber()`
Snip baseline	`mapa.snip(iterations = 100)`
Alternative iterative reweighted partial least squares	`mapa.airpls(landa = 100)`

Normalization

Value / (Max - Min)	`mapa.norm()`
Considering a peak as reference	`mapa.norm_peak(peak = 1001)`
Vector normalization	`mapa.vector()`

1.5. Processing

Unmixing

Principal component analysis	`components, loadings = mapa.pca(path, num_components = 3, colors = 'auto')`
Vertex component analysis	`components = mapa.vca(num_components = 3)`
Multi curve resolution	`components = mapa.mcr(spectra)`

Multi curve resolution (MCR)

Unsupervised Methods

Kmeans++

```
mapa.kmeans(num_components = 3)
```

Hierchical clustering

```
mapa.HCA(type_distance = ’euclidean’, linkage = ‘ward’ , distance = 3 , num_branches = 3)
```

Clara

Cure

Semi-Supervised Methods

PLS

PLS-LDA

Plotting

Predefined methods

spectrum

stack

map

profile

Remarks on Python – Appendix

Loading and package configuration

Installation

External libraries: psysptools, pyclustering, skitlearn, matplolib, numpy,

Library (functions – flow of programming)

Reading

Pre-Processing

Processing

Visualization