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 imaging spectroscopic point of view rather pure programming and developing approach (Appendix code).

A high-level programming means a friendly interface user to computer with easygoing and shorting way of programming for fast and efficient hyperspectral analysis.

HSI comes with 4 datasets for easygoing understanding.

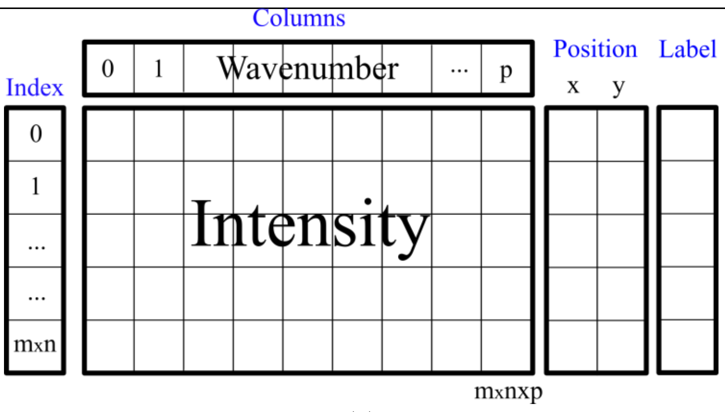
Plastics	Datasets of different plastics
Plastics+Tissue	Raman Map of tissue model and microplastics
Plastics+Layer of plastics	Z scanning of transparent plastic layers
Cancer+noncancer bladder	Cancer and noncancer tissue models
3D imaging	3D Raman Imaging of translucent tissue and plastics

The previous datasets will illustrate how to process the raw datasets and obtain a processed results by different HIS 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)

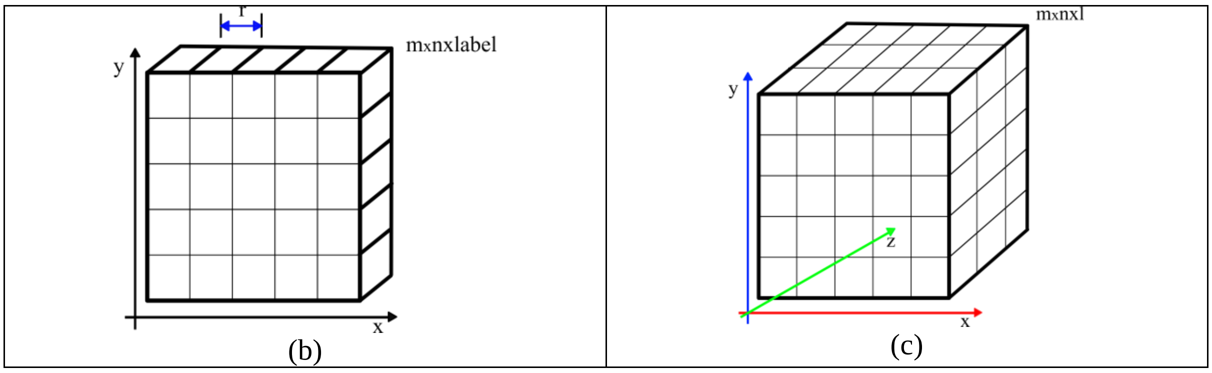


Figure 1. Hyperspectral object. (a) representation of hyperpectral object, (b) 2D plotting of hyperspectral data and (c) voxel illustration for 3D imaging.

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.

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

```
mapa = hyper_data('name')
mapa.delete()
```

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

```
mapa.read_csv('plastics')
mapa.read_multi_spc(path)
Mapa.read_map_SPC(path)
mapa.show(True)
mapa.save_data(path, 'name')
```

1.2.3. Creating Hyperspectral Objects from a custom file

```
mapa.set_data(data_file)
mapa.create_position(number x points, number y points)
mapa.set_calibration(calibration_row)
```

1.2.4. Access to the data

```
data = mapa.get_data()
position = mapa.get_position()
wavenumber = mapa.get_wavenumber()
number = mapa.get_number_points()
```

1.2.5. Selecting pixel from an hyperspectral image

```
spectrum = mapa.get_spectrum(x, y)
```

1.3.6. 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])`

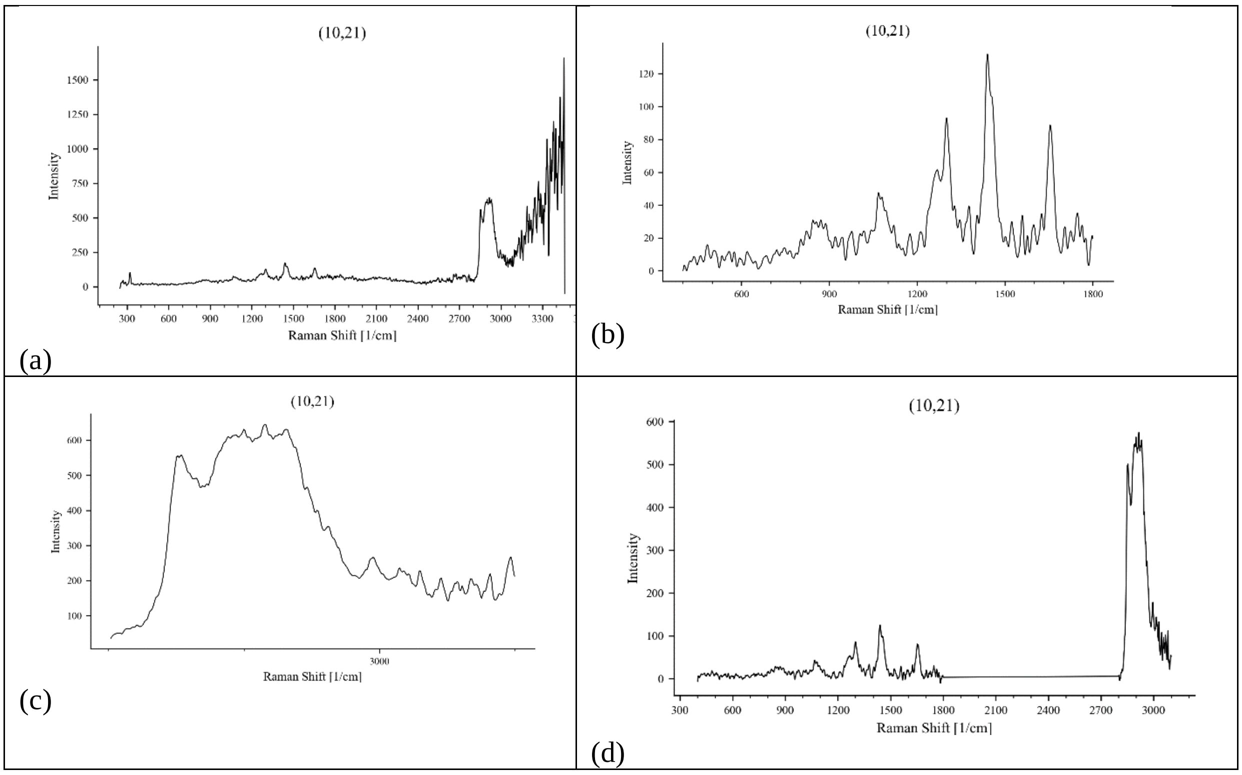


Figure 2. Selection of work range. (a) full range, (b) low wavenumber (finger print region), (c) high wavenumber and (d) selection of low and high wavenumber regions and ignoring the silent region.

Combining Spectra

Difference, Sum and Multiplication

Shifting Spectra

Removing Outliers

```
mapa.threshold(lower = 50, upper = 500)

mapa.covarience(contamination = 10)

mapa.spikes(threshold = 7, window = 3)
```

Smoothing

```
mapa.gol(window = 400, interpolation = 1800, order = 1800)
```

```
mapa.gaussian(variance = 1800)
```

Background correction

```
mapa.background(window = 400, interpolation = 1800, order = 1800)
```

Baseline correction

```
mapa.airpls(value = 3)
```

```
mapa.rubber()
```

Normalization

```
mapa.vector()
```

```
mapa.norm_peak(peak = 1001)
```

2.3. Processing

Unmixing

Principal component analysis (PCA)

```
components, loadings = mapa.PCA(num_components = 3, colors = ['blue'], path)
```

Vertex component analysis (VCA)

```
components = mapa.VCA(num_components = 3, colors = ['blue'], path)
```

Multi curve resolution (MCR)

Unsupervised Methods

Kmeans++

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

Hierchical clustering

```
clusters = 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

Read_SPC_Holo(path)

Read_multi_SPC(path)

Pre-Processing

Processing

Visualization