A.3.3. Basic Data Handling methods

This section provides methods for data manipulation such as getting intensity or position data. For example, the user can build the hyperobject using manually methods such as set_data() and set_position() and set_label(). The upcoming examples will explain how to use most of the following methods:

`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)`
`copied = mapa.copy()`	`mapa.set_resolution(res)`
`label = mapa.get_label()`	`mapa.set_label(label)`
`name = mapa.get_name()`	`mapa.set_name(name)`
`number = mapa.get_number_spectra()`	`mapa.append(hyperobject)`
`spectrum = mapa.get_pixel(x, y)`	`mapa.reset_index()`
`label = mapa.select_label(list)`	`mapa.concat([hyperobject1, ...])`
`wavenumber = mapa.get_wavenumber()`	`mapa.set_wavenumber(wavenumber)`
	`mapa.clean_data()`
	`mapa.remove_spectrum(index)`
	`mapa.rename_label(list_before, list_after)`
	`mapa.remove_label(list)`
	`mapa.add_peaks(0.5, 'red')`
	`mapa.remove_label([label])`
	`mapa.remove_spectrum([index])`

A.4. Pre-Processing

A.4.1. Selecting wavelength ranges

The selection of work range defines the spectral region of interest. Fig. A.3 shows the full spectral range and examples of range selections.

Fingerprint region (Fig. A.3(b))	`mapa.keep(lower = 400, upper = 1800)`
High wavenumber region (Fig. A.3(c))	`mapa.keep(lower = 2800, upper = 3200)`
Without silent region (Fig. A.3(d))	`mapa.keep(lower = 400, upper = 3200)` `mapa.remove(lower = 1800, upper = 2700)`

A. SpectraMap

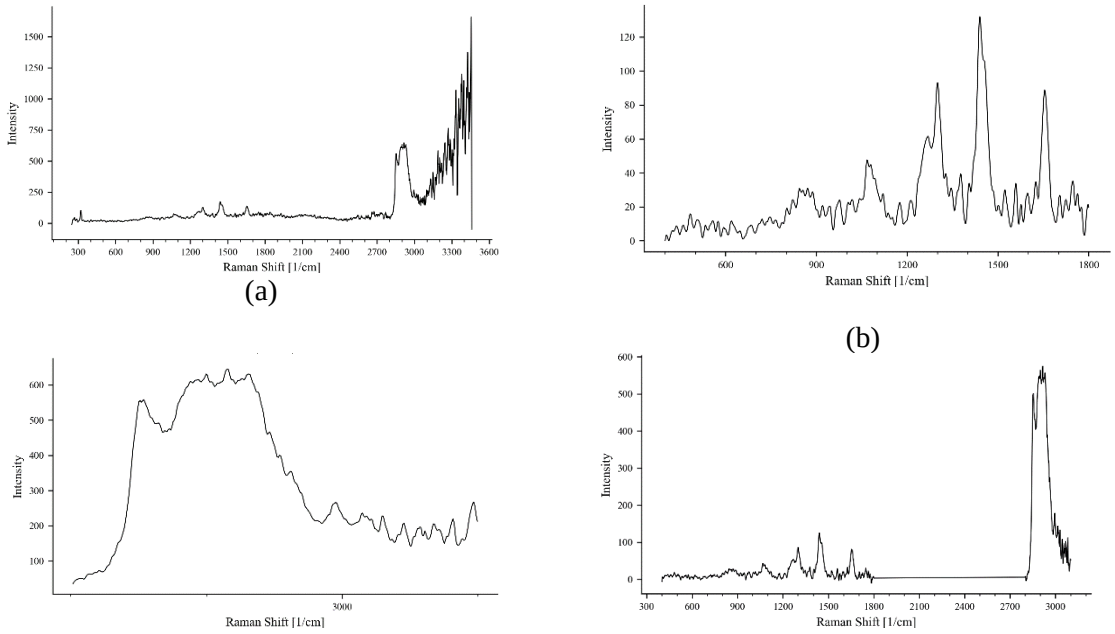


Figure A.3. Selection of work range. (a) full range, (b) low wavenumber (fingerprint region), (c) high wavenumber and (d) selection of low and high wavenumber regions without the silent region.

A.4.2. Pre-processing

Methods for removing noise, outliers, baseline correction, normalization are presented in the next lines.

Removing Outliers

Intensity threshold	`mapa.threshold(peak, lower = 50, upper = 500)`
Cosmic Ray removal tool	`mapa.spikes(threshold = 7, window = 3)`

Smoothing

Savitzky golay filter	`mapa.gol(window = 7, interpolation = 3, order = 0)`
Gaussian filter	`mapa.gaussian(variance = 2)`

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()`

A.3. Processing

The next lines provide an overview of the functions in SpMap as processing tools.

Principal component analysis	`components, loadings = mapa.pca(num_components = 3 ,norm = 'False' )`

Partial least square and linear discriminant analsys	`components, loadings = mapa.pls_lda(num_components = 3 ,norm = 'False' )`
Vertex component analysis	`endmembers = mapa.vca(num_components = 3)`
*Multi curve resolution	`components = mapa.mcr_als(spectra, constrain = 'NNLS', num_iterations = 100)`
Kmeans	`mapa.kmeans(3); mapa.kmeans('auto')`
Hierchical clustering	`mapa.hca('euclidean', 'ward', 2, 3)`
abundance	`concentrations = mapa.abundance(endmembers, constrain = 'NNLS')`
dbscan	`mapa.dbscan(min_samples, eps)`
hdbscan	`mapa.hdbscan(min_samples, min_cluster)`

A.4. Visualization

The final product of processing is the correct visualization of the data by different methods such as plotting spectra, maps, or scatter plots.

Overlapped plot of spectra	`mapa.show_spectra(0,0,'auto')`
Stack of spectra	`mapa.show_stack(0,0,'auto')`
Profile	`mapa.profile('auto')`
Map	`mapa.show_map('auto', None, 1)`
Show scatter points	`scores.show_scatter('auto', label, size)`
Show scatter points 3d	`scores.show_scatter_3d('auto', label, size)`
Show 3d map	`mapa.scatter3D()`
Show intensity map	`intensity.show_intensity()`

A.5. Hyperspectral imaging examples

Set the work file path to the file of examples path is located.

A.5.1 Plastics

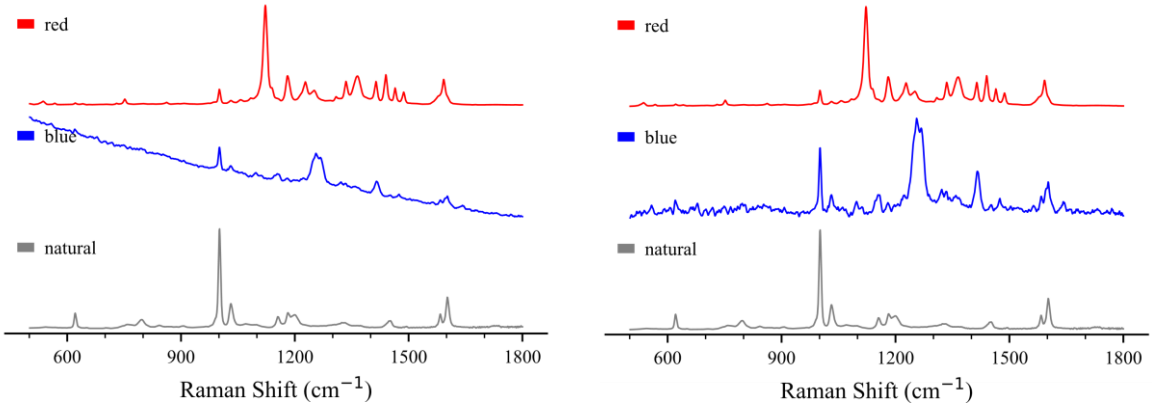
The following script shows how to use *SpMap* library for reading some spectra data and apply baseline correction. Fig. A.4 illustrates the results after the script.

```
from spectramap import spmap as sp
### Defining paths ###
blue_path = 'examples/plastics/blue'
red_path = 'examples/plastics/red'
natural_path = 'examples/plastics/natural'
### Creating and reading the files ###
red = sp.hyper_object('red') ## Declaring the first hyper object for red sample
red.read_csv(red_path) ## Reading the comma separated vector file
red.set_label('red') ## Setting red name to the whole data set spectra
red.keep(500, 1800) ## Keep finger region
meanred = red.mean() ## Compute mean of all red spectra

blue = sp.hyper_object('blue') ## Same procedure for the blue sample
blue.read_csv(blue_path)
```

```
blue.set_label('blue')
blue.keep(500, 1800)
blue.rubber() ## Baseline correction rubber band
meanblue = blue.mean()

natural = sp.hyper_object('natural') ## Same procedure for the natural sample
natural.read_csv(natural_path)
red.set_label('natural')
natural.keep(500, 1800)
meannatural = natural.mean()
### Concataneting the three hyperspectral objects
concat = sp.hyper_object('concat') ## create a new empty object
concat.concat([meanred, meanblue, meannatural])
concat.show_stack(0,0,['red', 'blue', 'gray']) ## show the spectra (see Fig. 5.4)
```



(a) (b)

Figure A.4. Modified plastics: (a) raw data and (b) result

A.5.2. Bladder

The next example displays how to process an hyperspectral image by clustering for obtaining the classification of different tissue signatures. Fig. A.5 illustrates the results after the script.

```
from spectramap import spmap as sp
### reading ###
path = 'H:/Thesis IPHT/Python/examples/bladder/data'
mapa = sp.hyper_object('Patient')
mapa.read_csv(path)
mapa.set_resolution(0.3) ## 300 µm step size resolution
mapa.vector() # vector normalization
original = mapa.get_data() ## get data
### K-means clustering
mapa.kmeans(3) #K-means clustering: 3 components
colors = mapa.show_map(['gray', 'black', 'green'], None, 1) # show 2D map (Fig.
5.5(a))
mapa.show_stack(0, 0, colors) #show stack of the clusters (Fig. 5.5(c))
mapa.remove_label([1]) # remove the noisy cluster
mapa.rename_label([2, 3], ['P', 'L']) # rename the remaining clusters
```

```
colors = mapa.show_map(['black', 'green'], None, 1) # show 2D map (Fig. 5.5 (b))
mapa.show_stack(0, 0, colors) #show stack of the clusters

### Hierarchical clustering
mapa.set_data(original)
mapa.hca('euclidean', 'ward', 1.5, 3) # hca clustering with 1.5 distance  (Fig.
5.5(d))
colors = mapa.show_map(['gray', 'black', 'lightgray', 'green'], None, 1)
### Saving data
mapa.save_data(path, 'clustering') # saving data
```

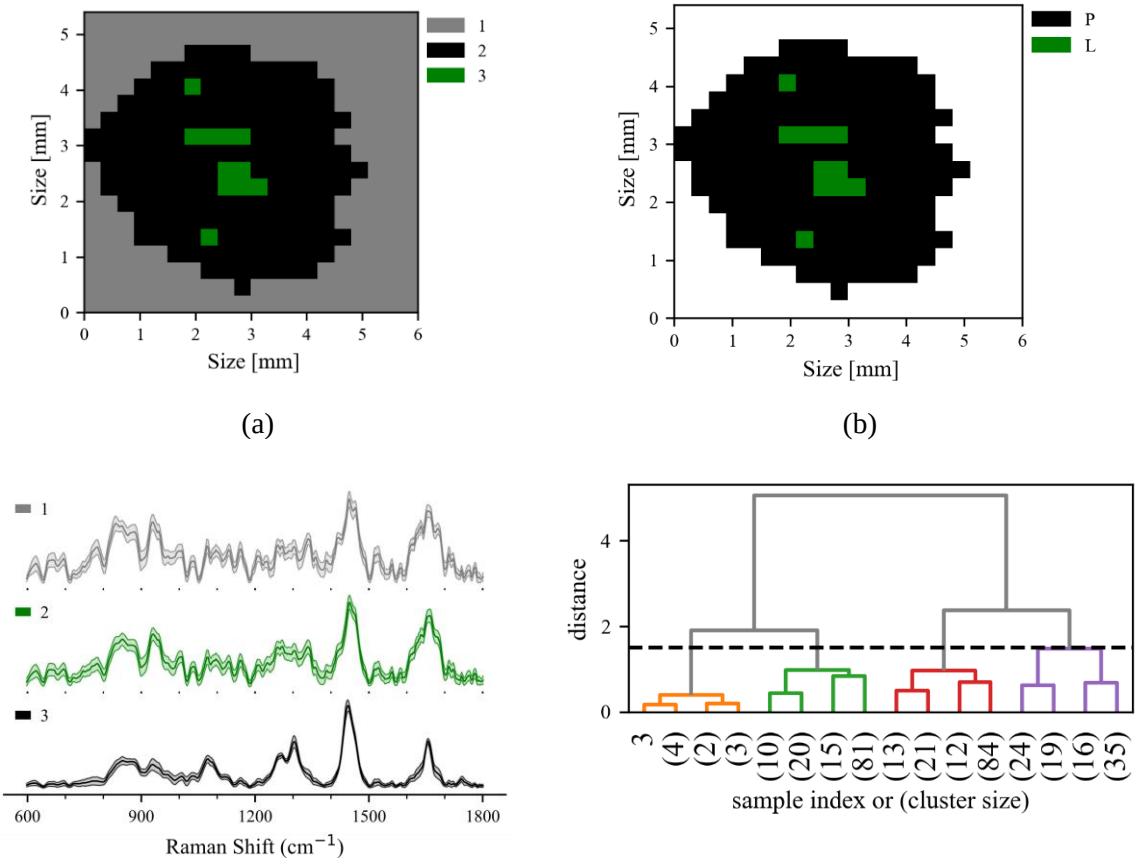


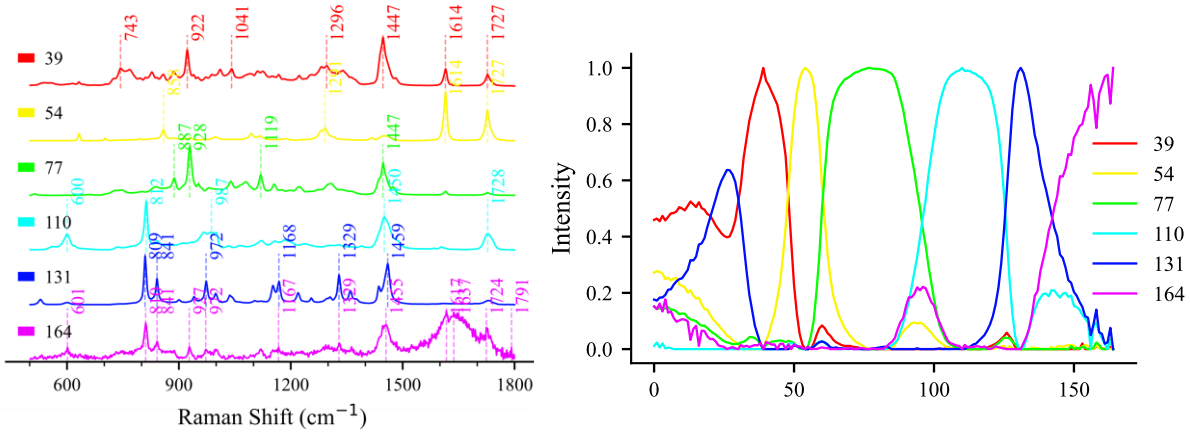
Figure A.5. Clustering results: (a) K-means clustering map, (b) Clean clustering map (without background), (c) cluster means and (d) dendrogram of hierarchical clustering

A.5.3. Multilayer of plastics

The following example: multi-layer of plastics illustrates how to analyze the data of a hyperspectral profile. Fig. A.6 illustrates the results after the script.

```
from spectramap import spmap as sp
### reading ###
path = 'examples/layers/data'
stack = sp.hyper_object('layers') #creating the hyper object
stack.read_csv(path) #reading the csv file
### preprocessing ###
stack.keep(500, 1800) #finger print selection
stack.rubber() #baseline correction
```

```
stack.vector() #vector normalization
### processing ###
endmember = stack.vca(6) # vertex component analysis
endmember.show_stack(1, 0, 'auto') # visualization of spectra and strong
peaks(Fig.5.6(a))
abundance = stack.abundance(endmember, 'NNLS') # concentration estimation by NNLS
abundance.set_resolution(0.02) # set resolution of 20 µm for the profile
abundance.profile('auto') # plot profile (Fig.5.6(b))
```

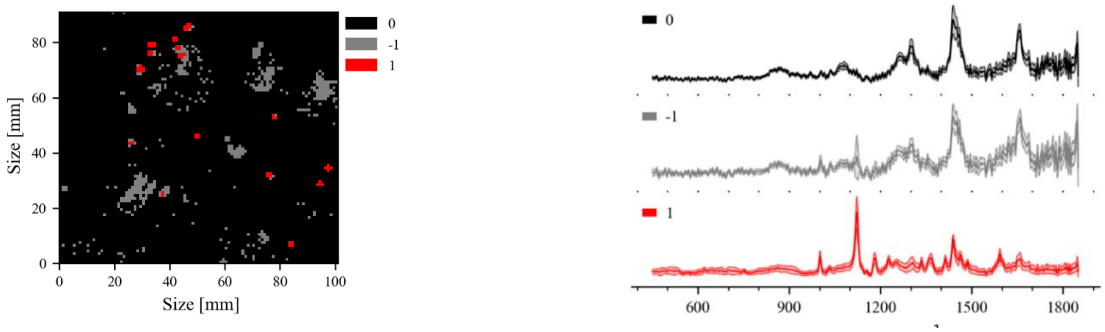


(a) (b)

Figure A.6. Hyperspectral profile analysis: (a) stack of plastics components found by VCA and (b) concentration profile.

A.5.4. Microplastic in tissue

The next examples illustrate the detection of pixel-sized microplastics on a tissue matrix using advanced clustering tools. Fig. A.7 illustrates the results after the script.



(a) (b)

Figure A.7. Density based clustering or detection of small pixels details: (a) hyperspectral image and (b) spectral cluster means.

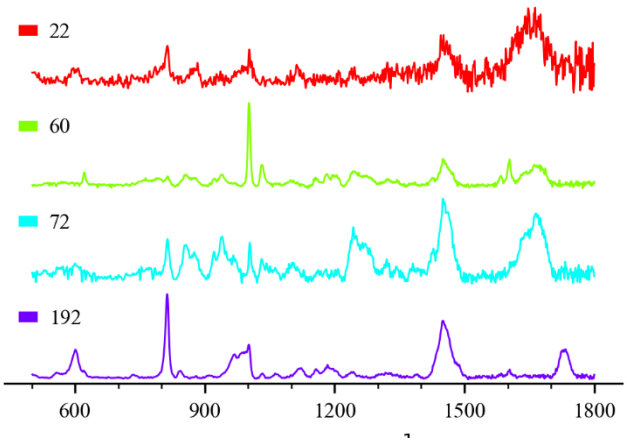
```
from spectramap import spmap as sp
### reading ###
path = 'examples/microplastic+tissue/data'
micro = sp.hyper_object('MP') #creating the hyper object
micro.read_csv(path) #reading the csv file
### processing ###
micro.hdbscan(2, 15) # hierarchical density-based clustering
```

```
colors = micro.show_map(['gray', 'k', 'r'], None, 1) # 2D map of the clusters
(Fig.5.6(a))
micro.show_stack(0,0, colors) # stack of the spectral clusters (Fig. 5.6(b))
```

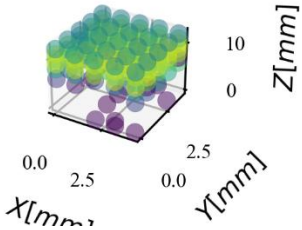
A.5.5. 3D hyperspectral imaging

The final example shows how to handle basically 3D hyperspectral data and scatter 3D visualization as voxel. Fig. A.8 illustrates the results after the script.

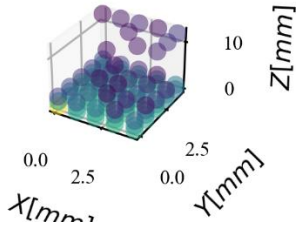
```
from spectramap import spmap as sp
### reading ###
path = 'examples/3D/data'
cube = sp.hyper_object('cube') #creating the hyper object
cube.read_csv_3D(path, 0.06, 0.07) #reading the 3d csv file and placing the
resolutions xy = 60 µm and z = 70 µm
### preprocessing ###
cube.keep(500, 1800) #finger print selection
cube.airpls(100) #advanced baseline correction
cube.vector() #vector normalization
### processing ###
vca = cube.vca(4) # number of expected components
vca.show_stack(0, 0, 'auto')
abundance = cube.abundance(vca, 'NNLS') # concentration estimation by NNLS
aux = abundance.show_intensity_3d(0.3) # 3d plot of all clusters (Fig. 5.8(b-c))
```



(a)



(b)



(c)

Figure A.8 3D imaging: (a) stack of the endmembers, (b) 3D concentration map of first layer 72 label and (c) 3D concentration map of 192.