NSOM analysis

- We use nbextension in special interact to manipulate line profiles
- NumPy is very useful to manipulate images
- NSOM experiments are exported like vectors, these vectors represent x, y and z

by Oscar Ruiz-Cigarrillo

In [23]:

```
import glob as glob
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.patches as patches
import sys
from matplotlib import cm
from matplotlib import gridspec

from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
from tqdm.auto import tqdm, trange
from IPython.display import display
from IPython.display import HTML
from IPython import display
import pandas as pd
#matplotlib options
plt.rcParams['font.family']         = 'Times New Roman'
#plt.rcParams['font.serif']          = 'Times'
plt.rcParams['xtick.labelsize']     = 17
plt.rcParams['ytick.labelsize']     = 17
plt.rcParams['axes.linewidth']      = 2
plt.rcParams["xtick.minor.visible"] =  False
plt.rcParams["xtick.major.size"]    =  10
plt.rcParams["xtick.minor.size"]    =  5
plt.rcParams["xtick.major.width"]   =  2
plt.rcParams["xtick.minor.width"]   =  2
plt.rcParams["xtick.direction"]     =  'in'
plt.rcParams["ytick.minor.visible"] =  False
plt.rcParams["ytick.major.size"]    =  10
plt.rcParams["ytick.minor.size"]    =  5
plt.rcParams["ytick.major.width"]   =  2
plt.rcParams["ytick.minor.width"]   =  2
plt.rcParams["ytick.direction"]     =  'in'
plt.rcParams['text.usetex']         = True
plt.rcParams['legend.frameon']      = False

# Import files

files = glob.glob("../DATA/*")
nfiles =  [nfiles.split("/")[-1]  for nfiles in files]
datal = {'names': nfiles }
df = pd.DataFrame(data=datal)
df
```

Out[23]:

	names
0	DatosHeight.dat
1	DatosNSOM.dat
2	FG163-5um.stp
3	FG163-5um.txt
4	FG163R_5um.opju
5	RAFM_FG163R.txt
6	RNSOM_FG163R.txt
7	Rutina_FG163R5um.nb
8	TAFM_FG163R.txt
9	TNSOM_FG163R.txt

This function read experimental data and :

- First select experiment mode: T for trace or R for retrace
- Then reshape the array
- Finally return the correct data

In [20]:

```
def nsoma(file,tipo="T"):
    data1 = np.loadtxt(file)
    if tipo == "T":
        Ra = data1[:int(len(data1)/2)].reshape((256,256))
    elif tipo=="R":
        Ra = data1[int(len(data1)/2):int(len(data1))].reshape((256,25
6))

    Rac = Ra.copy()
    Rac[1::2] =  Ra[1::2,::-1]

    return Rac

nsoma(files[0])
```

Out[20]:

```
array([[-482.49 , -482.665, -482.839, ..., -498.272, -498.621, -498.1
85],
       [-481.706, -481.967, -482.578, ..., -498.36 , -498.621, -498.5
34],
       [-481.88 , -481.095, -480.747, ..., -494.523, -494.872, -496.6
16],
       ...,
       [-464.615, -464.179, -464.441, ..., -486.153, -486.153, -485.8
04],
       [-464.615, -464.964, -464.615, ..., -485.891, -485.717, -486.5
89],
       [-470.457, -470.632, -470.283, ..., -485.978, -486.065, -486.0
65]])
```

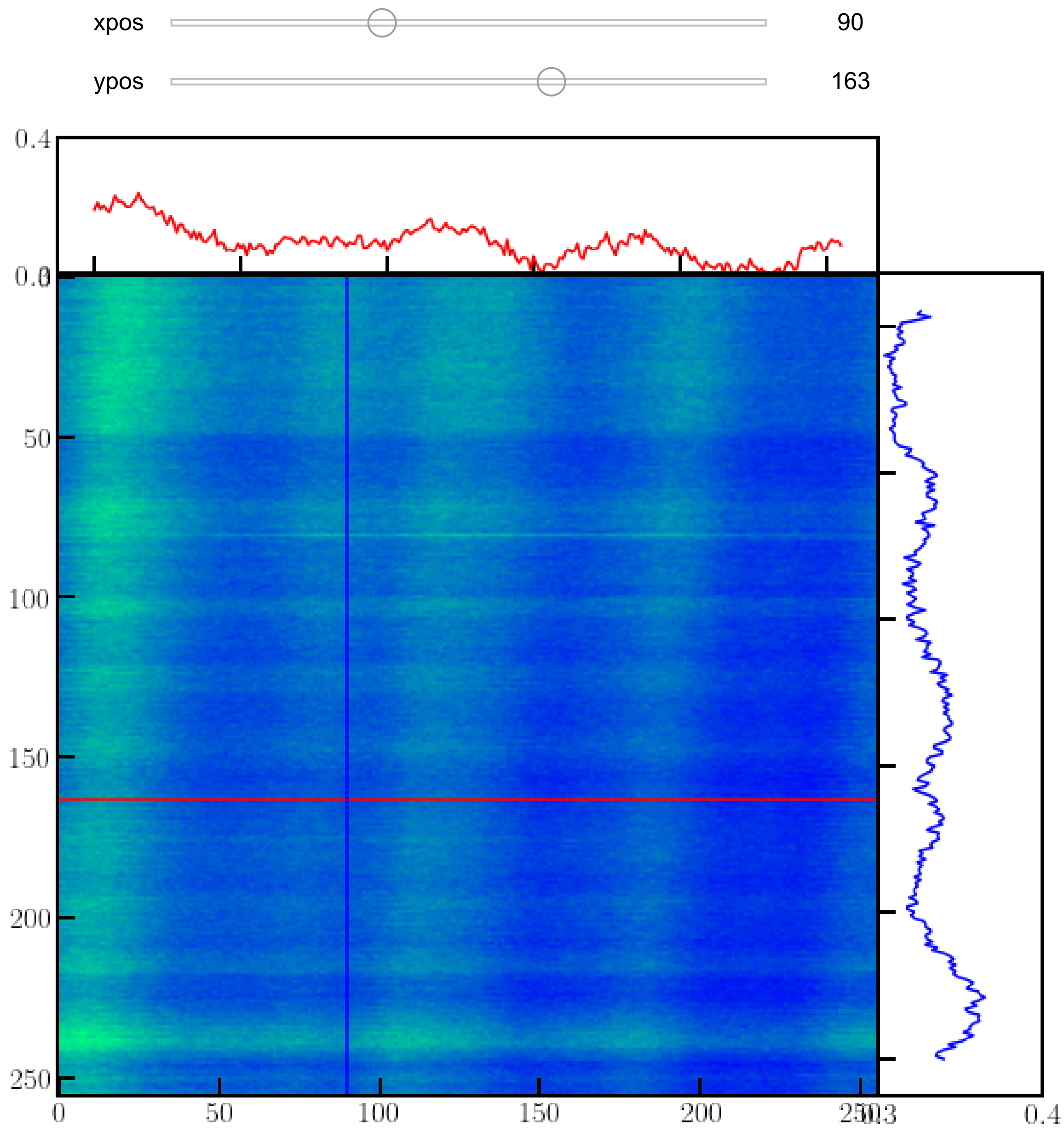
Function to map image

In [24]:

```
import scipy
import matplotlib.gridspec as gridspec
from matplotlib import cm
from scipy import ndimage
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
from ipywidgets.embed import embed_minimal_html


def mapnsom(xpos,ypos,image):
    nx, ny = image.shape[1], image.shape[0]
    X, Y = np.meshgrid(np.arange(0, nx, 1), np.arange(0, ny, 1))
    Z   = np.flip(image)


    fig = plt.figure(figsize=(10,10))
    spec = gridspec.GridSpec(3,3,width_ratios=[0.1,0.4,0.1],height_ratios=[0.1,0.4,0.2],hspace=0,wspace=0)
    ax1  = fig.add_subplot(spec[1:,0:2])
    ax2  = fig.add_subplot(spec[0,:2],xticklabels=[])
    ax3  = fig.add_subplot(spec[1:,2:],yticklabels=[])





    ax1.imshow(Z,cmap=cm.winter)
    ax1.plot([0,255],[ypos,ypos],'r')
    ax1.plot([xpos,xpos],[0,255],'b')

    #ax1.plot([x,x],[0,255],'b')



    ax2.plot(X[0],Z[ypos,:],'r')
    ax2.set_ylim([0.3,0.4])

    ax3.plot(Z[xpos,:],X[0],'b')
    ax3.set_xlim([0.3,0.4])


    plt.show()
```

In [25]:

```
%%time
%matplotlib inline
import matplotlib.animation as animation
from matplotlib.animation import PillowWriter
from mpl_toolkits.axes_grid1 import make_axes_locatable
import matplotlib.animation


#from IPython.display import HTML
plt.rcParams["animation.html"] = "jshtml"
plt.rcParams['figure.facecolor'] = 'w'

image = nsoma(files[1],tipo="T")

yy = widgets.IntSlider(min = 0, max=256-1, step=1,layout={'width': '500px'}, continuous_update=False,readout=True,)
xx = widgets.IntSlider(min = 0, max=256-1, step=1,layout={'width': '500px'}, continuous_update=False,readout=True,)

def upnsom(xpos=0,ypos=0):
    return  mapnsom(xpos,ypos,image)


nsommap =  interact(upnsom,
                               xpos=xx,
                               ypos=yy,
                              )
nsommap
```



```
CPU times: user 1.03 s, sys: 93.8 ms, total: 1.12 s
Wall time: 1.11 s
```

```
Out[25]: <function __main__.upnsom(xpos=0, ypos=0)>
```